MC68HC912BC32

Technical Summary
16-Bit Microcontroller

1 Introduction

The MC68HC912BC32 microcontroller unit (MCU) is a16-bit device composed of standard on-chip peripherals including a 16-bit central processing unit (CPU12), 32-Kbyte flash EEPROM, 1-Kbyte RAM, 768-byte EEPROM, an asynchronous serial communications interface (SCI), a serial peripheral interface (SPI), an 8-channel timer and 16-bit pulse accumulator, an 10-bit analog-to-digital converter (ADC), a four-channel pulse-width modulator (PWM), and a CAN 2.0B compatible controller (MSCAN12). System resource mapping, clock generation, interrupt control and bus interfacing are managed by the Lite integration module (LIM). The MC68HC912BC32 has full 16-bit data paths throughout, however, the multiplexed external bus can operate in an 8-bit narrow mode so single 8-bit wide memory can be interfaced for lower cost systems.

1.1 Features

- 16-Bit CPU12
  - — Upward Compatible with M68HC11 Instruction Set
  - — Interrupt Stacking and Programmer's Model Identical to M68HC11
  - — 20-Bit ALU
  - — Instruction Queue
  - — Enhanced Indexed Addressing
  - — Fuzzy Logic Instructions
- Multiplexed Bus
  - — Single Chip or Expanded
  - — 16/16 Wide or 16/8 Narrow Modes
- Memory
  - — 32-Kbyte Flash EEPROM with 2-Kbyte Erase-Protected Boot Block
  - — 768-byte EEPROM
  - — 1-Kbyte RAM with Single-Cycle Access for Aligned or Misaligned Read/Write
- 8-Channel, 10-Bit Analog-to-Digital Converter
- 8-Channel Timer
  - — Each Channel Fully Configurable as Either Input Capture or Output Compare
  - — Simple PWM Mode
  - — Modulo Reset of Timer Counter
- 16-Bit Pulse Accumulator
  - — External Event Counting
  - — Gated Time Accumulation
- Pulse-Width Modulator
  - — 8-Bit, 4-Channel or 16-Bit, 2-Channel

— Separate Control for Each Pulse Width and Duty Cycle
— Programmable Center-Aligned or Left-Aligned Outputs
• Serial Interfaces
— Asynchronous Serial Communications Interface (SCI)
— Synchronous Serial Peripheral Interface (SPI)
• Motorola Scalable CAN Controller (MSCAN12)
— CAN 2.0B compatible (standard and extended identifiers)
— Two receive and three transmit buffers
— Flexible identifier filter programmable as 2 x 32 bit, 4 x 16 bit, or 8 x 8 bit
— Four separate interrupt channels for Rx, Tx, error and wake-up
— Low-pass filter wake-up function
— Loop-back mode for self test operation
• COP Watchdog Timer, Clock Monitor, and Periodic Interrupt Timer
• 80-Pin QFP Package
— Up to 63 General-Purpose I/O Lines
— 4.5V–5.5V Operation at 8 MHz
• Single-Wire Background Debug™ Mode (BDM)
• On-Chip Hardware Breakpoints

1.2 Ordering Information

The MC68HC912BC32 is packaged in 80-pin quad flat pack (QFP) packaging and is shipped in two-piece sample packs, 50-piece trays, or 250-piece bricks. Operating temperature range and voltage requirements are specified when ordering the MC68HC912BC32 device. Refer to **Table 1** for part numbers.

Table 1 MC68HC912BC32 Device Ordering Information

Order Number	Temperature		Voltage	Frequency	Package
	Range	Designator			
MC68HC912BC32FU8	0 to +70 °C		4.5V–5.5V	8 MHz	80-Pin QFP Single Tray 50 Pcs
MC68HC912BC32CFU8	–40 to +85 °C	C			

NOTE: This part is also available in 2-piece sample packs and 250-piece bricks.

Evaluation boards, assemblers, compilers, and debuggers are available from Motorola and from third-party suppliers. An up-to-date list of products that support the M68HC12 family of microcontrollers can be found on the World Wide Web at the following URL:

`http://www.mcu.motsps.com`

Documents to assist in product selection are available from the Motorola Literature Distribution Center or your local Motorola Sales Office:

AMCU Device Selection Guide (SG166/D)

AMCU Software and Development Tool Selector Guide (SG176/D)

1.3 MC68HC912BC32 Block Diagram

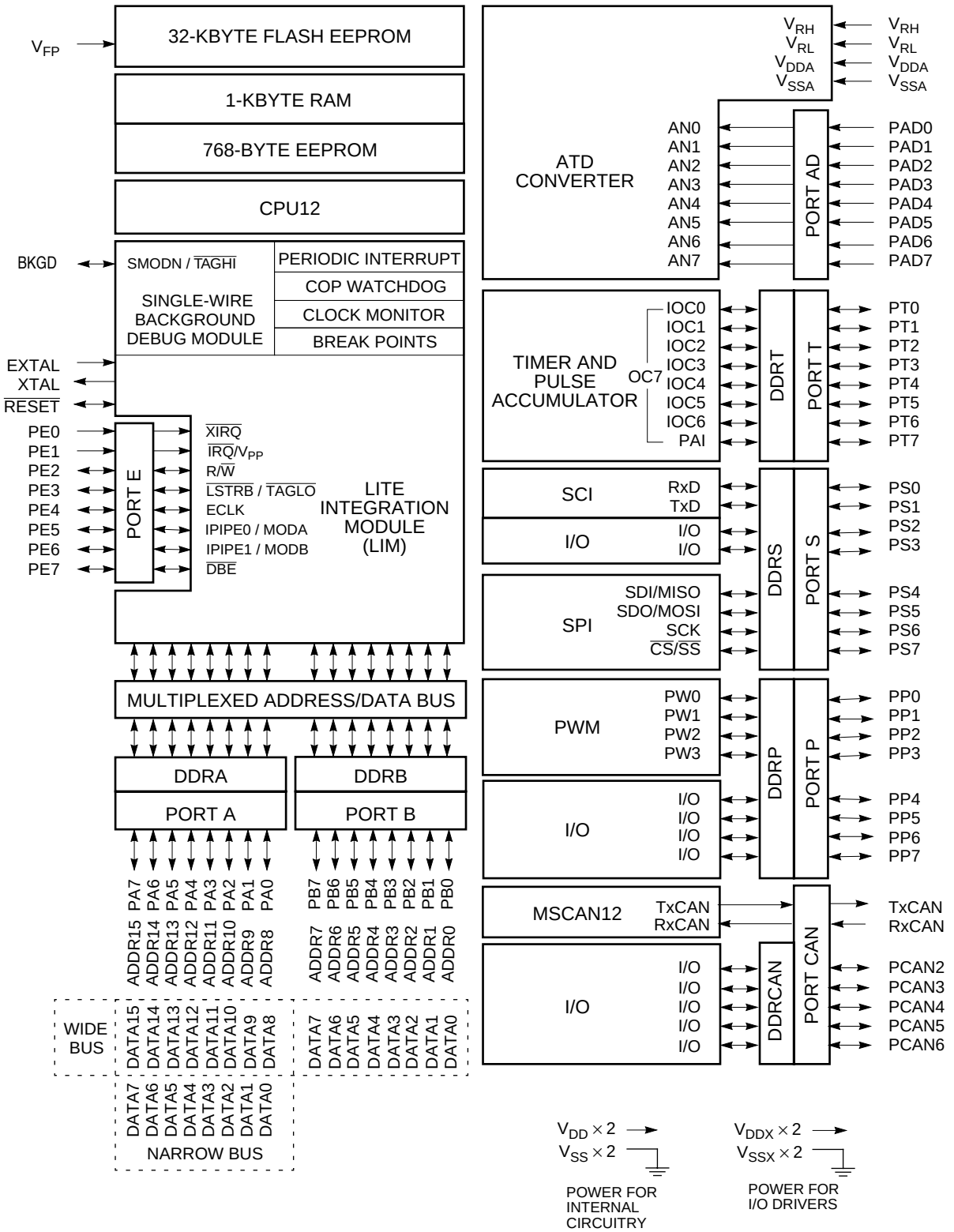


Figure 1 MC68HC912BC32 Block Diagram

2 Central Processing Unit

The CPU12 is a high-speed, 16-bit processing unit. It has full 16-bit data paths and wider internal registers (up to 20 bits) for high-speed extended math instructions. The instruction set is a proper superset of the M68HC11instruction set. The CPU12 allows instructions with odd byte counts, including many single-byte instructions. This provides efficient use of ROM space. An instruction queue buffers program information so the CPU always has immediate access to at least three bytes of machine code at the start of every instruction. The CPU12 also offers an extensive set of indexed addressing capabilities.

2.1 Programming Model

CPU12 registers are an integral part of the CPU and are not addressed as if they were memory locations.

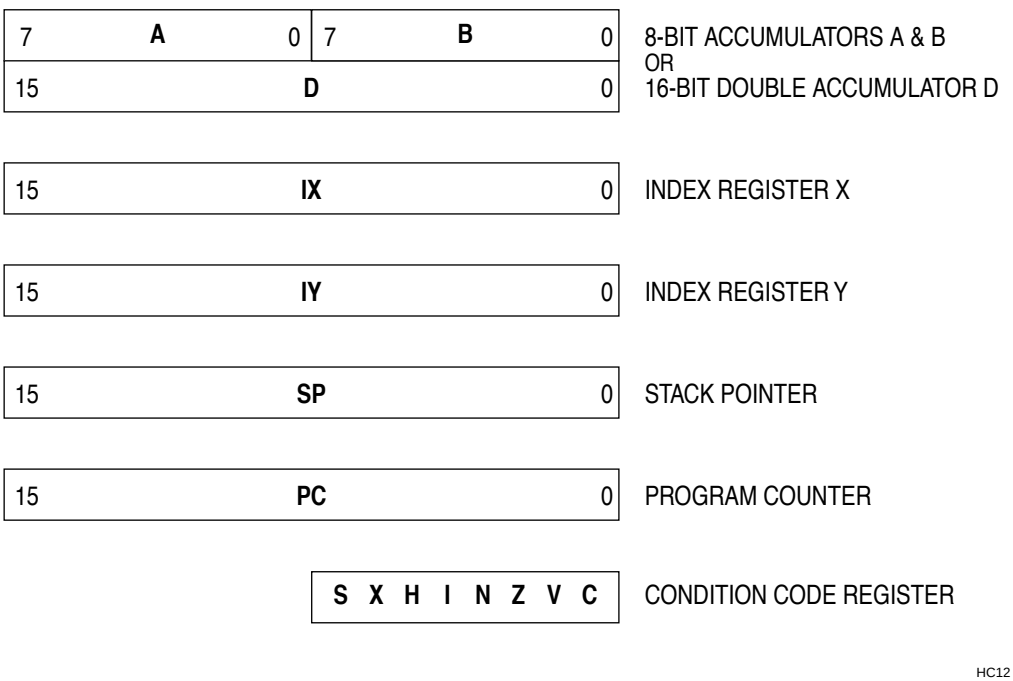


Figure 2 Programming Model

Accumulators A and B are general-purpose 8-bit accumulators used to hold operands and results of arithmetic calculations or data manipulations. Some instructions treat the combination of these two 8-bit accumulators as a 16-bit double accumulator (accumulator D).

Index registers X and Y are used for indexed addressing mode. In the indexed addressing mode, the contents of a 16-bit index register are added to 5-bit, 9-bit, or 16-bit constants or the content of an accumulator to form the effective address of the operand to be used in the instruction.

Stack pointer (SP) points to the last stack location used. The CPU12 supports an automatic program stack that is used to save system context during subroutine calls and interrupts, and can also be used for temporary storage of data. The stack pointer can also be used in all indexed addressing modes.

Program counter is a 16-bit register that holds the address of the next instruction to be executed. The program counter can be used in all indexed addressing modes except autoincrement/decrement.

Condition Code Register (CCR) contains five status indicators, two interrupt masking bits, and a STOP disable bit. The five flags are half carry (H), negative (N), zero (Z), overflow (V), and carry/borrow

(C). The half-carry flag is used only for BCD arithmetic operations. The N, Z, V, and C status bits allow for branching based on the results of a previous operation.

2.2 Data Types

The CPU12 supports the following data types:

- Bit data
- 8-bit and 16-bit signed and unsigned integers
- 16-bit unsigned fractions
- 16-bit addresses

A byte is eight bits wide and can be accessed at any byte location. A word is composed of two consecutive bytes with the most significant byte at the lower value address. There are no special requirements for alignment of instructions or operands.

2.3 Addressing Modes

Addressing modes determine how the CPU accesses memory locations to be operated upon. The CPU12 includes all of the addressing modes of the M68HC11 CPU as well as several new forms of indexed addressing. **Table 2** is a summary of the available addressing modes.

Table 2 M68HC12 Addressing Mode Summary

Addressing Mode	Source Format	Abbreviation	Description
Inherent	**INST** (no externally supplied operands)	INH	Operands (if any) are in CPU registers
Immediate	**INST** #*opr8i* or **INST** #*opr16i*	IMM	Operand is included in instruction stream 8- or 16-bit size implied by context
Direct	**INST** *opr8a*	DIR	Operand is the lower 8-bits of an address in the range $0000 – $00FF
Extended	**INST** *opr16a*	EXT	Operand is a 16-bit address
Relative	**INST** *rel8* or **INST** *rel16*	REL	An 8-bit or 16-bit relative offset from the current pc is supplied in the instruction
Indexed (5-bit offset)	**INST** *oprx5*,*xysp*	IDX	5-bit signed constant offset from x, y, sp, or pc
Indexed (auto pre-decrement)	**INST** *oprx3*,–*xys*	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexed (auto pre-increment)	**INST** *oprx3*,+*xys*	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexed (auto post-decrement)	**INST** *oprx3*,*xys*–	IDX	Auto post-decrement x, y, or sp by 1 ~ 8
Indexed (auto post-increment)	**INST** *oprx3*,*xys*+	IDX	Auto post-increment x, y, or sp by 1 ~ 8
Indexed (accumulator offset)	**INST** *abd*,*xysp*	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from x, y, sp, or pc
Indexed (9-bit offset)	**INST** *oprx9*,*xysp*	IDX1	9-bit signed constant offset from x, y, sp, or pc (lower 8-bits of offset in one extension byte)
Indexed (16-bit offset)	**INST** *oprx16*,*xysp*	IDX2	16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)
Indexed-Indirect (16-bit offset)	**INST [***oprx16*,*xysp***]**	[IDX2]	Pointer to operand is found at... 16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)
Indexed-Indirect (D accumulator offset)	**INST [D,***xysp***]**	[D,IDX]	Pointer to operand is found at... x, y, sp, or pc plus the value in D

2.4 Indexed Addressing Modes

The CPU12 indexed modes reduce execution time and eliminate code size penalties for using the Y index register. CPU12 indexed addressing uses a postbyte plus zero, one, or two extension bytes after the instruction opcode. The postbyte and extensions do the following tasks:

- Specify which index register is used.
- Determine whether a value in an accumulator is used as an offset.
- Enable automatic pre- or post-increment or decrement
- Specify use of 5-bit, 9-bit, or 16-bit signed offsets.

Table 3 Summary of Indexed Operations

Postbyte Code (xb)	Source Code Syntax	Comments rr; 00 = X, 01 = Y, 10 = SP, 11 = PC
rr0nnnnn	,r n,r –n,r	**5-bit constant offset** n = –16 to +15 r can specify X, Y, SP, or PC
111rr0zs	n,r –n,r	**Constant offset** (9- or 16-bit signed) z- 0 = 9-bit with sign in LSB of postbyte(s) -256 < n < 255 1 = 16-bit 0 < n < 65,535 if z = s = 1, 16-bit offset indexed-indirect (see below) r can specify X, Y, SP, or PC
111rr011	[n,r]	**16-bit offset indexed-indirect** rr can specify X, Y, SP, or PC 0 < n < 65,535
rr1pnnnn	n,–r n,+r n,r– n,r+	**Auto pre-decrement/increment** or **Auto post-decrement/increment**; p = pre-(0) or post-(1), n = –8 to –1, +1 to +8 r can specify X, Y, or SP (PC not a valid choice) +8 = 0111 … +1 = 0000 -1 = 1111 … -8 = 1000
111rr1aa	A,r B,r D,r	**Accumulator offset** (unsigned 8-bit or 16-bit) aa- 00 = A 01 = B 10 = D (16-bit) 11 = see accumulator D offset indexed-indirect r can specify X, Y, SP, or PC
111rr111	[D,r]	**Accumulator D offset indexed-indirect** r can specify X, Y, SP, or PC

2.5 Opcodes and Operands

The CPU12 uses 8-bit opcodes. Each opcode identifies a particular instruction and associated addressing mode to the CPU. Several opcodes are required to provide each instruction with a range of addressing capabilities.

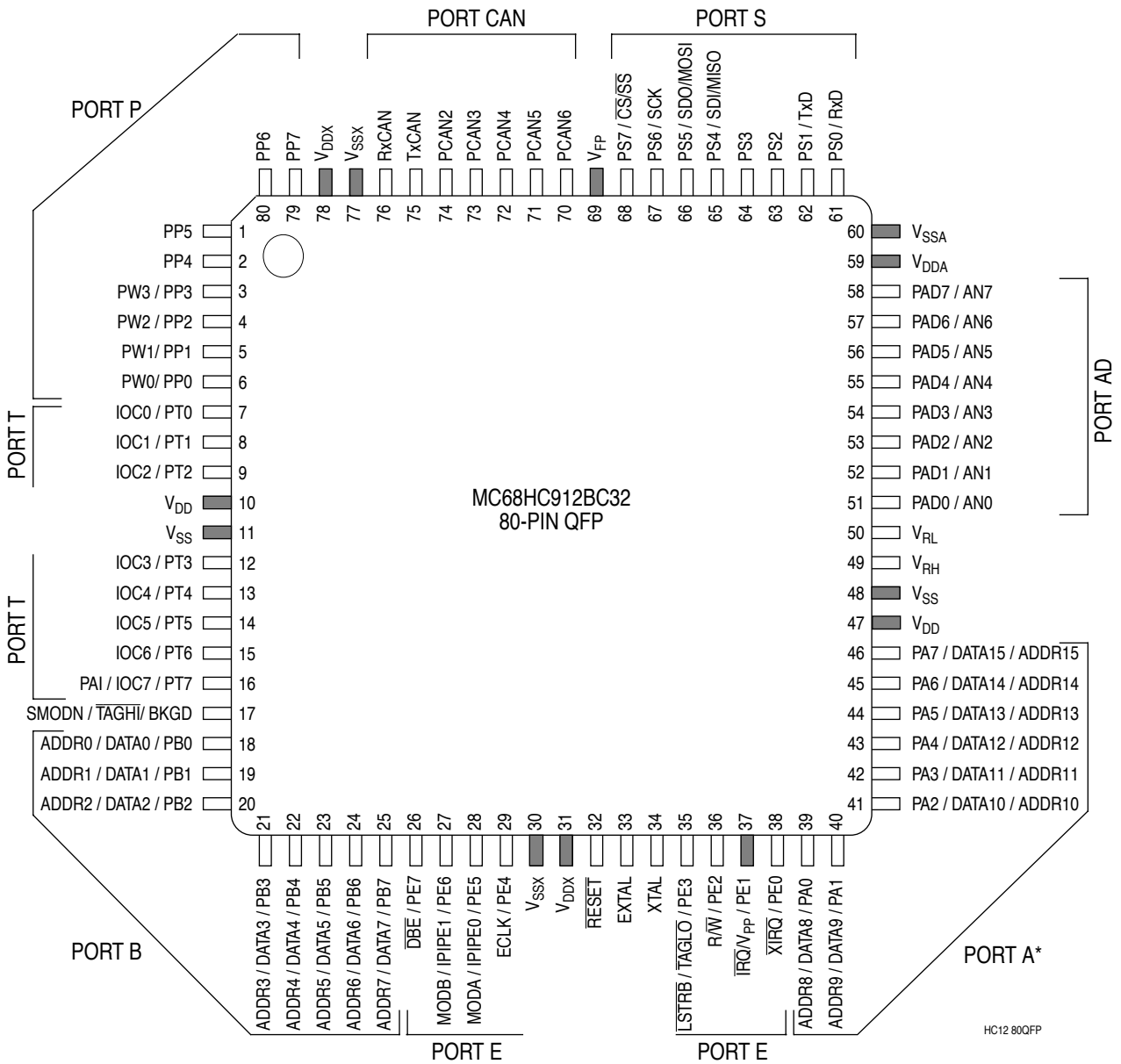
Only 256 opcodes would be available if the range of values were restricted to the number that can be represented by 8-bit binary numbers. To expand the number of opcodes, a second page is added to the opcode map. Opcodes on the second page are preceded by an additional byte with the value $18.

To provide additional addressing flexibility, opcodes can also be followed by a postbyte or extension bytes. Postbytes implement certain forms of indexed addressing, transfers, exchanges, and loop primitives. Extension bytes contain additional program information such as addresses, offsets, and immediate data.

3 Pinout and Signal Descriptions

3.1 MC68HC912BC32 Pin Assignments

The MC68HC912BC32 is available in a 80-pin quad flat pack (QFP). Most pins perform two or more functions, as described in the **3.3 Signal Descriptions**. **Figure 3** shows pin assignments. Shaded pins are power and ground.



* In narrow mode, high and low data bytes are multiplexed in alternate bus cycles on port A.

Figure 3 Pin Assignments for MC68HC912BC32

3.2 Power Supply Pins

MC68HC912BC32 power and ground pins are described below and summarized in **Table 4**.

3.2.1 Internal Power (V_{DD}) and Ground (V_{SS})

Power is supplied to the MCU through V_{DD} and V_{SS}. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded.

3.2.2 External Power (V_{DDX}) and Ground (V_{SSX})

External power and ground for I/O drivers. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded.

3.2.3 V_{DDA}, V_{SSA}

Provides operating voltage and ground for the analog-to-digital converter. This allows the supply voltage to the A/D to be bypassed independently.

3.2.4 Analog to Digital Reference Voltages (V_{RH}, V_{RL})

3.2.5 V_{FP}

Flash EEPROM programming voltage and supply voltage during normal operation.

3.2.6 V_{PP}

High voltage supply to EEPROM. Used to monitor charge pump output and testing. Not intended for general applications use.

Table 4 MC68HC912BC32 Power and Ground Connection Summary

Mnemonic	Pin Number	Description
V_{DD}	10, 47	Internal power and ground.
V_{SS}	11, 48	
V_{DDX}	31, 78	External power and ground, supply to pin drivers.
V_{SSX}	30, 77	
V_{DDA}	59	Operating voltage and ground for the analog-to-digital converter, allows the supply voltage to the A/D to be bypassed independently.
V_{SSA}	60	
V_{RH}	49	Reference voltages for the analog-to-digital converter.
V_{RL}	50	
V_{FP}	69	Programming voltage for the Flash EEPROM and required supply for normal operation.
V_{PP}	37	High voltage supply to EEPROM used for test purposes only in Special modes.

3.3 Signal Descriptions

3.3.1 Crystal Driver and External Clock Input (XTAL, EXTAL)

These pins provide the interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuitry. Out of reset the frequency applied to EXTAL is twice the desired E–clock rate. All the device clocks are derived from the EXTAL input frequency.

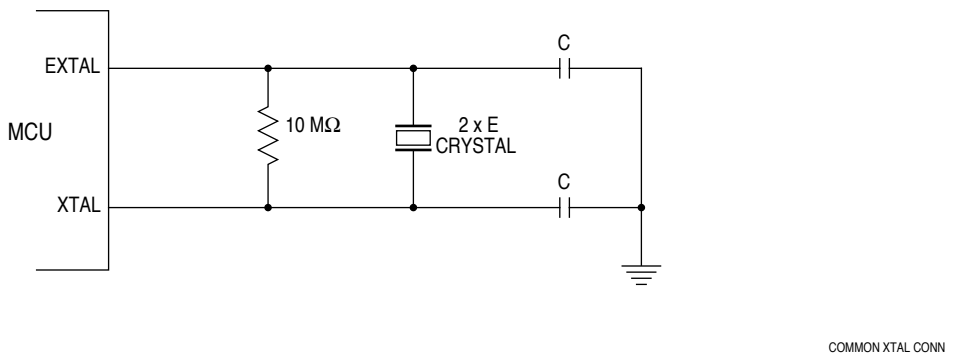


Figure 4 Common Crystal Connections

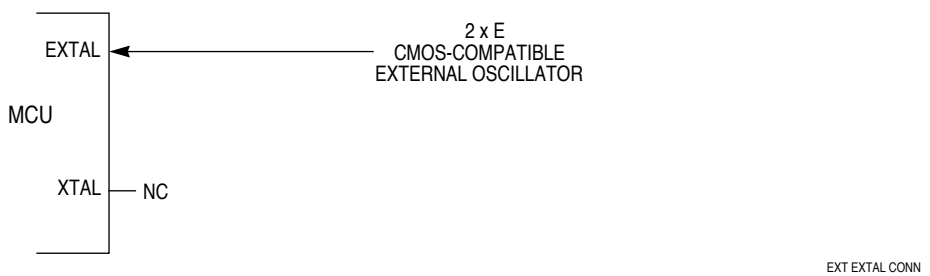


Figure 5 External Oscillator Connections

XTAL is the crystal output.The XTAL pin must be left unterminated when an external CMOS compatible clock input is connected to the EXTAL pin. The XTAL output is normally intended to drive only a crystal. The XTAL output can be buffered with a high-impedance buffer to drive the EXTAL input of another device.

In all cases take extra care in the circuit board layout around the oscillator pins. Load capacitances shown in the oscillator circuits include all stray layout capacitances. Refer to **Figure 4** and **Figure 5** for diagrams of oscillator circuits.

3.3.2 E-Clock Output (ECLK)

ECLK is the output connection for the internal bus clock and is used to demultiplex the address and data and is used as a timing reference. ECLK frequency is equal to 1/2 the crystal frequency out of reset.

In normal single-chip mode the E-clock output is off at reset to reduce the effects of RFI, but can be turned on if necessary.

In special single-chip mode the E-clock output is on at reset but can be turned off.

In special peripheral mode the E clock is an input to the MCU.

All clocks, including the E clock, are halted when the MCU is in STOP mode. It is possible to configure the MCU to interface to slow external memory. ECLK can be stretched for such accesses.

3.3.3 Reset ($\overline{\text{RESET}}$)

An active low bidirectional control signal, $\overline{\text{RESET}}$, acts as an input to initialize the MCU to a known start-up state. It also acts as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or COP watchdog circuit. The MCU goes into reset asynchronously and comes out of reset synchronously. This allows the part to reach a proper reset state even if the clocks have failed, while allowing synchronized operation when starting out of reset.

It is possible to determine whether a reset was caused by an internal source or an external source. An internal source drives the pin low for 16 cycles; eight cycles later the pin is sampled. If the pin has returned high, either the COP watchdog vector or clock monitor vector will be taken. If the pin is still low, the external reset is determined to be active and the reset vector is taken. Hold reset low for at least 32 cycles to assure that the reset vector is taken in the event that an internal COP watchdog time-out or clock monitor fail occurs.

3.3.4 Maskable Interrupt Request ($\overline{\text{IRQ}}$)

The $\overline{\text{IRQ}}$ input provides a means of applying asynchronous interrupt requests to the MCU. Either falling edge-sensitive triggering or level-sensitive triggering is program selectable (INTCR register). $\overline{\text{IRQ}}$ is always configured to level-sensitive triggering at reset. When the MCU is reset the $\overline{\text{IRQ}}$ function is masked in the condition code register.

This pin is always an input and can always be read. In special modes it can be used to apply external EEPROM V_{PP} in support of EEPROM testing. External V_{PP} is not needed for normal EEPROM program and erase cycles. Because the $\overline{\text{IRQ}}$ pin is also used as an EEPROM programming voltage pin, there is an internal resistive pull-up on the pin.

3.3.5 Nonmaskable Interrupt ($\overline{\text{XIRQ}}$)

The $\overline{\text{XIRQ}}$ input provides a means of requesting a nonmaskable interrupt after reset initialization. During reset, the X bit in the condition code register (CCR) is set and any interrupt is masked until MCU software enables it. Because the $\overline{\text{XIRQ}}$ input is level sensitive, it can be connected to a multiple-source wired-OR network. This pin is always an input and can always be read. There is an active pull-up on this pin while in reset and immediately out of reset. The pullup can be turned off by clearing PUPE in the PUCR register. $\overline{\text{XIRQ}}$ is often used as a power loss detect interrupt.

Whenever $\overline{\text{XIRQ}}$ or $\overline{\text{IRQ}}$ are used with multiple interrupt sources ($\overline{\text{IRQ}}$ must be configured for level-sensitive operation if there is more than one source of $\overline{\text{IRQ}}$ interrupt), each source must drive the interrupt input with an open-drain type of driver to avoid contention between outputs. There must also be an interlock mechanism at each interrupt source so that the source holds the interrupt line low until the MCU recognizes and acknowledges the interrupt request. If the interrupt line is held low, the MCU will recognize another interrupt as soon as the interrupt mask bit in the MCU is cleared (normally upon return from an interrupt).

3.3.6 Mode Select (SMODN, MODA, and MODB)

The state of these pins during reset determine the MCU operating mode. After reset, MODA and MODB can be configured as instruction queue tracking signals IPIPE0 and IPIPE1. MODA and MODB have active pulldowns during reset.

The SMODN pin can be used as BKGD or $\overline{\text{TAGHI}}$ after reset.

3.3.7 Single-Wire Background Mode Pin (BKGD)

The BKGD pin receives and transmits serial background debugging commands. A special self-timing protocol is used. The BKGD pin has an active pullup when configured as input; BKGD has no pullup control. Refer to **16 Development Support**.

3.3.8 External Address and Data Buses (ADDR[15:0] and DATA[15:0])

External bus pins share function with general-purpose I/O ports A and B. In single-chip operating modes, the pins can be used for I/O; in expanded modes, the pins are used for the external buses.

In expanded wide mode, ports A and B are used for multiplexed 16-bit data and address buses. PA[7:0] correspond to ADDR[15:8]/DATA[15:8]; PB[7:0] correspond to ADDR[7:0]/DATA[7:0].

In expanded narrow mode, ports A and B are used for the16-bit address bus, and an 8-bit data bus is multiplexed with the most significant half of the address bus on port A. In this mode, 16-bit data is handled as two back-to-back bus cycles, one for the high byte followed by one for the low byte. PA[7:0] correspond to ADDR[15:8] and to DATA[15:8] or DATA[7:0], depending on the bus cycle. The state of the address pin should be latched at the rising edge of E. To allow for maximum address setup time at external devices, a transparent latch should be used.

3.3.9 Read/Write (R/$\overline{W}$)

In all modes this pin can be used as I/O and is a general-purpose input with an active pull-up out of reset. If the read/write function is required it should be enabled by setting the RDWE bit in the PEAR register. External writes will not be possible until enabled.

3.3.10 Low-Byte Strobe ($\overline{LSTRB}$)

In all modes this pin can be used as I/O and is a general-purpose input with an active pull-up out of reset. If the strobe function is required, it should be enabled by setting the LSTRE bit in the PEAR register. This signal is used in write operations and so external low byte writes will not be possible until this function is enabled. This pin is also used as $\overline{TAGLO}$ in Special Expanded modes and is multiplexed with the $\overline{LSTRB}$ function.

3.3.11 Instruction Queue Tracking Signals (IPIPE1 and IPIPE0)

These signals are used to track the state of the internal instruction execution queue. Execution state is time-multiplexed on the two signals. Refer to **16 Development Support**.

3.3.12 Data Bus Enable ($\overline{DBE}$)

The $\overline{DBE}$ pin (PE7) is an active low signal that will be asserted low during E-clock high time. $\overline{DBE}$ provides separation between output of a multiplexed address and the input of data. When an external address is stretched, $\overline{DBE}$ is asserted during what would be the last quarter cycle of the last E-clock cycle of stretch. In expanded modes this pin is used to enable the drive control of external buses during external reads. Use of the $\overline{DBE}$ is controlled by the NDBE bit in the PEAR register.$\overline{DBE}$ is enabled out of reset in expanded modes. This pin has an active pullup during and after reset in single chip modes.

Table 5 MC68HC912BC32 Signal Description Summary

Pin Name	Pin Number	Description
PW[3:0]	3–6	Pulse Width Modulator channel outputs.
ADDR[7:0] DATA[7:0]	25–18	External bus pins share function with general-purpose I/O ports A and B. In single chip modes, the pins can be used for I/O. In expanded modes, the pins are used for the external buses.
ADDR[15:8] DATA[15:8]	46–39	
IOC[7:0]	16–12, 9–7	Pins used for input capture and output compare in the timer and pulse accumulator subsystem.
PAI	16	Pulse accumulator input
AN[7:0]	58–51	Analog inputs for the analog-to-digital conversion module
$\overline{DBE}$	26	Data bus control and, in expanded mode, enables the drive control of external buses during external reads.
MODB, MODA	27, 28	State of mode select pins during reset determine the initial operating mode of the MCU. After reset, MODB and MODA can be configured as instruction queue tracking signals IPIPE1 and IPIPE0 or as general-purpose I/O pins.
IPIPE1, IPIPE0	27, 28	
ECLK	29	E Clock is the output connection for the external bus clock. ECLK is used as a timing reference and for address demultiplexing.
$\overline{RESET}$	32	An active low bidirectional control signal, $\overline{RESET}$ acts as an input to initialize the MCU to a known start-up state, and an output when COP or clock monitor causes a reset.
EXTAL	33	Crystal driver and external clock input pins. On reset all the device clocks are derived from the EXTAL input frequency. XTAL is the crystal output.
XTAL	34	
$\overline{LSTRB}$	35	Low byte strobe (0 = low byte valid), in all modes this pin can be used as I/O. The low strobe function is the exclusive-NOR of A0 and the internal $\overline{SZ8}$ signal. (The $\overline{SZ8}$ internal signal indicates the size 16/8 access.)
$\overline{TAGLO}$	35	Pin used in instruction tagging. See **16 Development Support**.
R/$\overline{W}$	36	Indicates direction of data on expansion bus. Shares function with general-purpose I/O. Read/write in expanded modes.
$\overline{IRQ}$	37	Maskable interrupt request input provides a means of applying asynchronous interrupt requests to the MCU. Either falling edge-sensitive triggering or level-sensitive triggering is program selectable (INTCR register).
$\overline{XIRQ}$	38	Provides a means of requesting asynchronous nonmaskable interrupt requests after reset initialization
BKGD	17	Single-wire background interface pin is dedicated to the background debug function. During reset, this pin determines special or normal operating mode.
$\overline{TAGHI}$	17	Pin used in instruction tagging. See **16 Development Support**.
RxCAN	76	MSCAN12 receive pin
TxCAN	75	MSCAN12 transmit pin
$\overline{CS}$/$\overline{SS}$	68	Slave select output for SPI master mode, input for slave mode or master mode.
SCK	67	Serial clock for SPI system.
SDO/MOSI	66	Master out/slave in pin for serial peripheral interface
SDI/MISO	65	Master in/slave out pin for serial peripheral interface
TxD0	62	SCI transmit pin
RxD0	61	SCI receive pin

3.4 Port Signals

The MC68HC912BC32 incorporates eight ports which are used to control and access the various device subsystems. When not used for these purposes, port pins may be used for general-purpose I/O. In addition to the pins described below, each port consists of a data register which can be read and written at any time, and, with the exception of port AD and PE[1:0], a data direction register which controls the direction of each pin. After reset all port pins are configured as input.

3.4.1 Port A

Port A pins are used for address and data in expanded modes. The port data register is not in the address map during expanded and peripheral mode operation. When it is in the map, port A can be read or written at anytime.

Register DDRA determines whether each port A pin is an input or output. DDRA is not in the address map during expanded and peripheral mode operation. Setting a bit in DDRA makes the corresponding bit in port A an output; clearing a bit in DDRA makes the corresponding bit in port A an input. The default reset state of DDRA is all zeroes.

When the PUPA bit in the PUCR register is set, all port A input pins are pulled-up internally by an active pull-up device. This bit has no effect if the port is being used in expanded modes as the pull-ups are inactive.

Setting the RDPA bit in register RDRIV causes all port A outputs to have reduced drive level. RDRIV can be written once after reset. RDRIV is not in the address map in peripheral mode. Refer to **6 Bus Control and Input/Output**.

3.4.2 Port B

Port B pins are used for address and data in expanded modes. The port data register is not in the address map during expanded and peripheral mode operation. When it is in the map, port B can be read or written at anytime.

Register DDRB determines whether each port B pin is an input or output. DDRB is not in the address map during expanded and peripheral mode operation. Setting a bit in DDRB makes the corresponding bit in port B an output; clearing a bit in DDRB makes the corresponding bit in port B an input. The default reset state of DDRB is all zeroes.

When the PUPB bit in the PUCR register is set, all port B input pins are pulled-up internally by an active pull-up device. This bit has no effect if the port is being used in expanded modes as the pull-ups are inactive.

Setting the RDPB bit in register RDRIV causes all port B outputs to have reduced drive level. RDRIV can be written once after reset. RDRIV is not in the address map in peripheral mode. Refer to **6 Bus Control and Input/Output**.

3.4.3 Port E

Port E pins operate differently from port A and B pins. Port E pins are used for bus control signals and interrupt service request signals. When a pin is not used for one of these specific functions, it can be used as general-purpose I/O. However, two of the pins (PE[1:0]) can only be used for input, and the states of these pins can be read in the port data register even when they are used for $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$.

The PEAR register determines pin function, and register DDRE determines whether each pin is an input or output when it is used for general-purpose I/O. PEAR settings override DDRE settings. Because PE[1:0] are input-only pins, only DDRE[7:2] have effect. Setting a bit in the DDRE register makes the corresponding bit in port E an output; clearing a bit in the DDRE register makes the corresponding bit in port E an input. The default reset state of DDRE is all zeroes.

When the PUPE bit in the PUCR register is set, PE[7,3,2,0] are pulled up. PE[7,3,2,0] are pulled up active devices, while PE1 is always pulled up by means of an internal resistor.

Neither port E nor DDRE is in the map in peripheral mode; neither is in the internal map in expanded modes with EME set.

Setting the RDPE bit in register RDRIV causes all port E outputs to have reduced drive level. RDRIV can be written once after reset. RDRIV is not in the address map in peripheral mode. Refer to **6 Bus Control and Input/Output**.

3.4.4 Port CAN

The port CAN has five general-purpose I/O pins, PCAN[6:2]. The MSCAN12 receive pin, RxCAN, and transmit pin, TxCAN, cannot be configured as general-purpose I/O on port CAN.

Register DDRCAN determines whether each port CAN pin PCAN[6:2] is an input or output. Setting a bit in DDRCAN makes the corresponding pin in port CAN an output; clearing a bit makes the corresponding pin an input. After reset port CAN pins PCAN[6:2] are configured as inputs.

When a read to the port CAN is performed, the value read for the MSB depends on the MSB, PCAN7, of the port CAN data register, PORTCAN, and the MSB of DDRCAN: it is 0 if DDRCAN7=0 and is PCAN7 if DDRCAN7=1.

When the PUECAN bit in the PCTLCAN register is set, port CAN input pins PCAN[6:2] are pulled up internally by an active pull-up device.

Setting the RDRCAN bit in register PCTLCAN causes the port CAN outputs PCAN[6:2] to have reduced drive level. Levels are at normal drive capability after reset. RDRCAN can be written anytime after reset. Refer to **14 MSCAN12 Controller**.

3.4.5 Port AD

Input to the analog-to-digital subsystem and general-purpose input. When analog-to-digital functions are not enabled, the port has eight general-purpose input pins, PAD[7:0]. The ADPU bit in the ATDCTL2 register enables the A/D function.

Port AD pins are inputs; no data direction register is associated with this port. The port has no resistive input loads and no reduced drive controls. Refer to **15 Analog-To-Digital Converter**.

3.4.6 Port P

The four pulse-width modulation channel outputs share general-purpose port P pins. The PWM function is enabled with the PWEN register. Enabling PWM pins takes precedence over the general-purpose port. When pulse-width modulation is not in use, the port pins may be used for general-purpose I/O.

Register DDRP determines pin direction of port P when used for general-purpose I/O. When DDRP bits are set, the corresponding pin is configured for output. On reset the DDRP bits are cleared and the corresponding pin is configured for input.

When the PUPP bit in the PWCTL register is set, all input pins are pulled up internally by an active pull-up device. Pullups are disabled after reset.

Setting the RDPP bit in the PWCTL register configures all port P outputs to have reduced drive levels. Levels are at normal drive capability after reset. The PWCTL register can be read or written anytime after reset. Refer to **11 Pulse Width Modulator**.

3.4.7 Port T

This port provides eight general-purpose I/O pins when not enabled for input capture and output compare in the timer and pulse accumulator subsystem. The TEN bit in the TSCR register enables the timer function. The pulse accumulator subsystem is enabled with the PAEN bit in the PACTL register.

Register DDRT determines pin direction of port T when used for general-purpose I/O. When DDRT bits are set, the corresponding pin is configured for output. On reset the DDRT bits are cleared and the corresponding pin is configured for input.

When the PUPT bit in the TMSK2 register is set, all input pins are pulled up internally by an active pull-up device. Pullups are disabled after reset.

Setting the RDPT bit in the TMSK2 register configures all port T outputs to have reduced drive levels. Levels are at normal drive capability after reset. The TMSK2 register can be read or written anytime after reset Refer to **12 Standard Timer Module**.

3.4.8 Port S

Port S is the 8-bit interface to the standard serial interface consisting of the serial communications interface (SCI) and serial peripheral interface (SPI) subsystems. Port S pins are available for general-purpose parallel I/O when standard serial functions are not enabled.

Port S pins serve several functions depending on the various internal control registers. If WOMS bit in the SC0CR1 register is set, the P-channel drivers of the output buffers are disabled for bits 0 through 1 (2 through 3). If SWOM bit in the SP0CR1 register is set, the P-channel drivers of the output buffers are disabled for bits 4 through 7. (wired-OR mode). The open drain control effects to both the serial and the general-purpose outputs. If the RDPSx bits in the PURDS register are set, the appropriate Port S pin drive capabilities are reduced. If PUPSx bits in the PURDS register are set, the appropriate pull-up device is connected to each port S pin which is programmed as a general-purpose input. If the pin is programmed as a general-purpose output, the pull-up is disconnected from the pin regardless of the state of the individual PUPSx bits. See **13 Serial Interface**.

Table 6 MC68HC912BC32 Port Description Summary

Port Name	Pin Numbers	Data Direction DD Register (Address)	Description
Port A PA[7:0]	46–39	In/Out DDRA ($0002)	Port A and port B pins are used for **address** and **data** in expanded modes. The port data registers are not in the address map during expanded and peripheral mode operation. When in the map, port A and port B can be read or written any time. DDRA and DDRB are not in the address map in expanded or peripheral modes.
Port B PB[7:0]	25–18	In/Out DDRB ($0003)	
Port AD PAD[7:0]	58–51	In	**Analog-to-digital converter** and general-purpose I/O.
Port CAN PCAN[6:2] TxCAN RxCAN	70–76	In/Out DDRCAN ($013F) for PCAN[6:2] TxCAN Out RxCAN In	**CAN Controller** (MSCAN12) subsystem with TxCAN output and RxCAN input and general-purpose I/O on PCAN[6:2].
Port E PE[7:0]	26–29, 35–38	PE[1:0] In PE[7:2] In/Out DDRE ($0009)	**Mode selection**, **bus control** signals and **interrupt service request** signals; or general-purpose I/O.
Port P PP[7:0]	79, 80, 1–6	In/Out DDRP ($0057)	General-purpose I/O. PP[3:0] are use with the **pulse-width modulator** when enabled.
Port S PS[7:0]	68–61	In/Out DDRS ($00D7)	**Serial communications interface** and **serial peripheral interface** subsystems and general-purpose I/O.
Port T PT[7:0]	16–12, 9–7	In/Out DDRT ($00AF)	General-purpose I/O when not enabled for input capture and output compare in the **timer** and **pulse accumulator** subsystem.

3.5 Port Pull-Up, Pull-Down and Reduced Drive

MCU ports can be configured for internal pull-up. To reduce power consumption and RFI, the pin output drivers can be configured to operate at a reduced drive level. Reduced drive causes a slight increase in transition time depending on loading and should be used only for ports which have a light loading. **Table 7** summarizes the port pull-up default status and controls.

Table 7 Port Pull-Up, Pull-Down and Reduced Drive Summary

Port Name	Resistive Input Loads	Enable Bit			Reduced Drive Control Bit		
		Register (Address)	Bit Name	Reset State	Register (Address)	Bit Name	Reset State
Port A	Pull-up	PUCR ($000C)	PUPA	Disabled	RDRIV ($000D)	RDPA	Full Drive
Port B	Pull-up	PUCR ($000C)	PUPB	Disabled	RDRIV ($000D)	RDPB	Full Drive
Port E:							
PE7, PE3, PE2, PE0	Pull-up	PUCR ($000C)	PUPE	Enabled	RDRIV ($000D)	RDPE	Full Drive
PE1	Pull-up	Always Enabled			RDRIV ($000D)	RDPE	Full Drive
PE[6:4]	None	—			RDRIV ($000D)	RDPE	Full Drive
PE[6:5]	Pull-down	Enabled During Reset			—	—	—
Port P	Pull-up	PWCTL ($0054)	PUPP	Disabled	PWCTL ($0054)	RDPP	Full Drive
Port S	Pull-up	PURDS ($00DB)	PUPS0	Disabled	PURDS ($00DB)	RDPS0	Full Drive
PS[3:2]	Pull-up	PURDS ($00DB)	PUPS1	Disabled	PURDS ($00DB)	RDPS1	Full Drive
PS[7:4]	Pull-up	PURDS ($00DB)	PUPS2	Disabled	PURDS ($00DB)	RDPS2	Full Drive
Port T	Pull-up	TMSK2 ($008D)	PUPT	Disabled	TMSK2 ($008D)	RDPT	Full Drive
Port CAN:							
PCAN[6:2]	Pull-up	PCTLCAN ($013D)	PUECAN	Disabled	PCTLCAN ($013D)	RDRCAN	Full Drive
TxCAN	None	—			—	—	Full Drive
RxCAN	Pull-up	—	—	Enabled	—		
Port AD	None	—			—		
BKGD	Pull-up	—	—	Enabled	—	—	Full Drive

4 Register Block

The register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space by manipulating bits REG[15:11] in the INITRG register. INITRG establishes the upper five bits of the register block's 16-bit address. The register block occupies the first 512 bytes of the 2-Kbyte block. Default addressing (after reset) is indicated in the table below. For additional information refer to **5 Operating Modes and Resource Mapping**.

Table 8 MC68HC912BC32 Register Map (Sheet 1 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA[1]
$0001	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PORTB[1]
$0002	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA[1]
$0003	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB[1]
$0004	0	0	0	0	0	0	0	0	Reserved
$0005	0	0	0	0	0	0	0	0	Reserved
$0006	0	0	0	0	0	0	0	0	Reserved
$0007	0	0	0	0	0	0	0	0	Reserved
$0008	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE[2]
$0009	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	0	0	DDRE[2]
$000A	NDBE	0	PIPOE	NECLK	LSTRE	RDWE	0	0	PEAR[2]
$000B	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	0	EME	MODE[3]
$000C	0	0	0	PUPE	0	0	PUPB	PUPA	PUCR[3]
$000D	0	0	0	0	RDPE	0	RDPB	RDPA	RDRIV[3]
$000E	0	0	0	0	0	0	0	0	Reserved
$000F	0	0	0	0	0	0	0	0	Reserved
$0010	RAM15	RAM14	RAM13	RAM12	RAM11	0	0	0	INITRM
$0011	REG15	REG14	REG13	REG12	REG11	0	0	MMSWAI	INITRG
$0012	EE15	EE14	EE13	EE12	0	0	0	EEON	INITEE
$0013	0	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	MAPROM	ROMON	MISC
$0014	RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0	RTICTL
$0015	RTIF	0	0	0	0	0	0	0	RTIFLG
$0016	CME	FCME	FCM	FCOP	DISR	CR2	CR1	CR0	COPCTL
$0017	Bit 7	6	5	4	3	2	1	Bit 0	COPRST
$0018	ITE6	ITE8	ITEA	ITEC	ITEE	ITF0	ITF2	ITF4	ITST0
$0019	ITD6	ITD8	ITDA	ITDC	ITDE	ITE0	ITE2	ITE4	ITST1
$001A	ITC6	ITC8	ITCA	ITCC	ITCE	ITD0	ITD2	ITD4	ITST2
$001B	0	0	0	0	0	ITC0	ITC2	ITC4	ITST3
$001C–$001D	0	0	0	0	0	0	0	0	Reserved
$001E	IRQE	IRQEN	DLY	0	0	0	0	0	INTCR
$001F	1	1	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	0	HPRIO
$0020	BKEN1	BKEN0	BKPM	0	BK1ALE	BK0ALE	0	0	BRKCT0
$0021	0	BKDBE	BKMBH	BKMBL	BK1RWE	BK1RW	BK0RWE	BK0RW	BRKCT1
$0022	Bit 15	14	13	12	11	10	9	Bit 8	BRKAH

Table 8 MC68HC912BC32 Register Map (Sheet 2 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0023	Bit 7	6	5	4	3	2	1	Bit 0	BRKAL
$0024	Bit 15	14	13	12	11	10	9	Bit 8	BRKDH
$0025	Bit 7	6	5	4	3	2	1	Bit 0	BRKDL
$0026–$003F	0	0	0	0	0	0	0	0	Reserved
$0040	CON23	CON01	PCKA2	PCKA1	PCKA0	PCKB2	PCKB1	PCKB0	PWCLK
$0041	PCLK3	PCLK2	PCLK1	PCLK0	PPOL3	PPOL2	PPOL1	PPOL0	PWPOL
$0042	0	0	0	0	PWEN3	PWEN2	PWEN1	PWEN0	PWEN
$0043	0	6	5	4	3	2	1	Bit 0	PWPRES
$0044	Bit 7	6	5	4	3	2	1	Bit 0	PWSCAL0
$0045	Bit 7	6	5	4	3	2	1	Bit 0	PWSCNT0
$0046	Bit 7	6	5	4	3	2	1	Bit 0	PWSCAL1
$0047	Bit 7	6	5	4	3	2	1	Bit 0	PWSCNT1
$0048	Bit 7	6	5	4	3	2	1	Bit 0	PWCNT0
$0049	Bit 7	6	5	4	3	2	1	Bit 0	PWCNT1
$004A	Bit 7	6	5	4	3	2	1	Bit 0	PWCNT2
$004B	Bit 7	6	5	4	3	2	1	Bit 0	PWCNT3
$004C	Bit 7	6	5	4	3	2	1	Bit 0	PWPER0
$004D	Bit 7	6	5	4	3	2	1	Bit 0	PWPER1
$004E	Bit 7	6	5	4	3	2	1	Bit 0	PWPER2
$004F	Bit 7	6	5	4	3	2	1	Bit 0	PWPER3
$0050	Bit 7	6	5	4	3	2	1	Bit 0	PWDTY0
$0051	Bit 7	6	5	4	3	2	1	Bit 0	PWDTY1
$0052	Bit 7	6	5	4	3	2	1	Bit 0	PWDTY2
$0053	Bit 7	6	5	4	3	2	1	Bit 0	PWDTY3
$0054	0	0	0	PSWAI	CENTR	RDP	PUPP	PSBCK	PWCTL
$0055	DISCR	DISCP	DISCAL	0	0	0	0	0	PWTST
$0056	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0	PORTPP
$0057	DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0	DDRP
$0058–$005F	0	0	0	0	0	0	0	0	Reserved
$0060	0	0	0	0	0	0	0	0	ATDCTL0
$0061	0	0	0	0	0	0	0	0	ATDCTL1
$0062	ADPU	AFFC	AWAI	0	0	0	ASCIE	ASCIF	ATDCTL2
$0063	0	0	0	0	0	0	FRZ1	FRZ0	ATDCTL3
$0064	S10BM	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0	ATDCTL4
$0065	0	S8CM	SCAN	MULT	CD	CC	CB	CA	ATDCTL5
$0066	SCF	0	0	0	0	CC2	CC1	CC0	ATDSTAT
$0067	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	ATDSTAT
$0068	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	ATDTSTH
$0069	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0	ATDTSTL
$006A–$006E	0	0	0	0	0	0	0	0	Reserved

Table 8 MC68HC912BC32 Register Map (Sheet 3 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$006F	PAD7	PAD6	PAD5	PAD4	PAD3	PAD2	PAD1	PAD0	PORTAD
$0070	Bit 15	14	13	12	11	10	9	Bit 8	ADR0H
$0071	Bit 7	6	5	4	3	2	1	Bit 0	ADR0L
$0072	Bit 15	14	13	12	11	10	9	Bit 8	ADR1H
$0073	Bit 7	6	5	4	3	2	1	Bit 0	ADR1L
$0074	Bit 15	14	13	12	11	10	9	Bit 8	ADR2H
$0075	Bit 7	6	5	4	3	2	1	Bit 0	ADR2L
$0076	Bit 15	14	13	12	11	10	9	Bit 8	ADR3H
$0077	Bit 7	6	5	4	3	2	1	Bit 0	ADR3L
$0078	Bit 15	14	13	12	11	10	9	Bit 8	ADR4H
$0079	Bit 7	6	5	4	3	2	1	Bit 0	ADR4L
$007A	Bit 15	14	13	12	11	10	9	Bit 8	ADR5H
$007B	Bit 7	6	5	4	3	2	1	Bit 0	ADR5L
$007C	Bit 15	14	13	12	11	10	9	Bit 8	ADR6H
$007D	Bit 7	6	5	4	3	2	1	Bit 0	ADR6L
$007E	Bit 15	14	13	12	11	10	9	Bit 0	ADR7H
$007F	Bit 7	6	5	4	3	2	1	Bit 0	ADR7L
$0080	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0	TIOS
$0081	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0	CFORC
$0082	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0	OC7M
$0083	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0	OC7D
$0084	Bit 15	14	13	12	11	10	9	Bit 8	TCNT (H)
$0085	Bit 7	6	5	4	3	2	1	Bit 0	TCNT (L)
$0086	TEN	TSWAI	TSBCK	TFFCA	0	0	0	0	TSCR
$0087	0	0	0	0	0	0	0	0	TQCR
$0088	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4	TCTL1
$0089	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0	TCTL2
$008A	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A	TCTL3
$008B	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A	TCTL4
$008C	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I	TMSK1
$008D	TOI	0	PUPT	RDPT	TCRE	PR2	PR1	PR0	TMSK2
$008E	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F	TFLG1
$008F	TOF	0	0	0	0	0	0	0	TFLG2
$0090	Bit 15	14	13	12	11	10	9	Bit 8	TC0 (H)
$0091	Bit 7	6	5	4	3	2	1	Bit 0	TC0 (L)
$0092	Bit 15	14	13	12	11	10	9	Bit 8	TC1 (H)
$0093	Bit 7	6	5	4	3	2	1	Bit 0	TC1 (L)
$0094	Bit 15	14	13	12	11	10	9	Bit 8	TC2 (H)
$0095	Bit 7	6	5	4	3	2	1	Bit 0	TC2 (L)
$0096	Bit 15	14	13	12	11	10	9	Bit 8	TC3 (H)
$0097	Bit 7	6	5	4	3	2	1	Bit 0	TC3 (L)
$0098	Bit 15	14	13	12	11	10	9	Bit 8	TC4 (H)

Table 8 MC68HC912BC32 Register Map (Sheet 4 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0099	Bit 7	6	5	4	3	2	1	Bit 0	TC4 (L)
$009A	Bit 15	14	13	12	11	10	9	Bit 8	TC5 (H)
$009B	Bit 7	6	5	4	3	2	1	Bit 0	TC5 (L)
$009C	Bit 15	14	13	12	11	10	9	Bit 8	TC6 (H)
$009D	Bit 7	6	5	4	3	2	1	Bit 0	TC6 (L)
$009E	Bit 15	14	13	12	11	10	9	Bit 8	TC7 (H)
$009F	Bit 7	6	5	4	3	2	1	Bit 0	TC7 (L)
$00A0	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI	PACTL
$00A1	0	0	0	0	0	0	PAOVF	PAIF	PAFLG
$00A2	Bit 15	14	13	12	11	10	9	Bit 8	PACNT
$00A3	Bit 7	6	5	4	3	2	1	Bit 0	PACNT
$00A4–$00AC	0	0	0	0	0	0	0	0	Reserved
$00AD	0	0	0	0	0	0	TCBYP	PCBYP	TIMTST
$00AE	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0	PORTT
$00AF	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0	DDRT
$00B0–$00BF	0	0	0	0	0	0	0	0	Reserved
$00C0	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8	SC0BDH
$00C1	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0	SC0BDL
$00C2	LOOPS	WOMS	RSRC	M	WAKE	ILT	PE	PT	SC0CR1
$00C3	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SC0CR2
$00C4	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	SC0SR1
$00C5	0	0	0	0	0	0	0	RAF	SC0SR2
$00C6	R8	T8	0	0	0	0	0	0	SC0DRH
$00C7	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SC0DRL
$00C8–$00CF	0	0	0	0	0	0	0	0	Reserved
$00D0	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF	SP0CR1
$00D1	0	0	0	0	0	0	SSWAI	SPC0	SP0CR2
$00D2	0	0	0	0	0	SPR2	SPR1	SPR0	SP0BR
$00D3	SPIF	WCOL	0	MODF	0	0	0	0	SP0SR
$00D4	0	0	0	0	0	0	0	0	Reserved
$00D5	Bit 7	6	5	4	3	2	1	Bit 0	SP0DR
$00D6	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0	PORTS
$00D7	DDS7	DDS6	DDS5	DDS4	DDS3	DDS2	DDS1	DDS0	DDRS
$00D8–$00DA	0	0	0	0	0	0	0	0	Reserved
$00DB	0	RDPS2	RDPS1	RDPS0	0	PUPS2	PUPS1	PUPS0	PURDS
$00DC–$00EF	0	0	0	0	0	0	0	0	Reserved
$00F0	1	1	1	1	1	EESWAI	PROTLCK	EERC	EEMCR
$00F1	1	1	1	BPROT4	BPROT3	BPROT2	BPROT1	BPROT0	EEPROT

Table 8 MC68HC912BC32 Register Map (Sheet 5 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$00F2	EEODD	EEVEN	MARG	EECPD	EECPRD	0	EECPM	0	EETST
$00F3	BULKP	0	0	BYTE	ROW	ERASE	EELAT	EEPGM	EEPROG
$00F4	0	0	0	0	0	0	0	LOCK	FEELCK
$00F5	0	0	0	0	0	0	0	BOOTP	FEEMCR
$00F6	FSTE	GADR	HVT	FENLV	FDISVFP	VTCK	STRE	MWPR	FEETST
$00F7	0	0	0	FEESWAI	SVFP	ERAS	LAT	ENPE	FEECTL
$00F8-$00FF	-	-	-	-	-	-	-	-	Reserved
$0100	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	CMCR0
$0101	0	0	0	0	0	LOOPB	WUPM	CLKSRC	CMCR1
$0102	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	CBTR0
$0103	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	CBTR1
$0104	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	CRFLG
$0105	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE	CRIER
$0106	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	CTFLG
$0107	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	CTCR
$0108	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	CIDAC
$0109-$010D	-	-	-	-	-	-	-	-	Reserved
$010E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	CRXERR
$010F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	CTXERR
$0110	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR0
$0111	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR1
$0112	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR2
$0113	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR3
$0114	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR0
$0115	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR1
$0116	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR2
$0117	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR3
$0118	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR4
$0119	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR5
$011A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR6
$011B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR7
$011C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR4
$011D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR5
$011E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR6
$011F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR7
$0120-$013C	-	-	-	-	-	-	-	-	Reserved
$013D	0	0	0	0	0	0	PUECAN	RDRCAN	PCTLCAN
$013E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCan	RxCan	PORTCAN
$013F	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0	DDRCAN

Table 8 MC68HC912BC32 Register Map (Sheet 6 of 6)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0140-$014F	u[4]	u	u	u	u	u	u	u	RECEIVE BUFFER
$0150-$015F	u	u	u	u	u	u	u	u	TRANSMIT BUFFER 0
$0160-$016F	u	u	u	u	u	u	u	u	TRANSMIT BUFFER 1
$0170-$017F	u	u	u	u	u	u	u	u	TRANSMIT BUFFER 2

NOTES:

1. Port A, port B, and data direction registers DDRA and DDRB are not in map in expanded and peripheral modes.
2. Port E and DDRE not in map in peripheral mode; also not in map in expanded modes with EME set.
3. Not in map in peripheral mode.
4. u means, the register bit is undefined out of reset.

5 Operating Modes and Resource Mapping

Eight possible operating modes determine the operating configuration of the MC68HC912BC32. Each mode has an associated default memory map and external bus configuration. After reset, most system resources can be mapped to other addresses by writing to the appropriate control registers.

5.1 Operating Modes

The operating mode out of reset is determined by the states of the BKGD, MODB, and MODA pins during reset.

The SMODN, MODB, and MODA bits in the MODE register show current operating mode and provide limited mode switching during operation. The states of the BKGD, MODB, and MODA pins are latched into these bits on the rising edge of the reset signal. During reset an active pullup is connected to the BKGD pin (as input) and active pulldowns are connected to the MODB and MODA pins. If an open occurs on any of these pins, the device will operate in normal single-chip mode.

Table 9 Mode Selection

BKGD	MODB	MODA	Mode	Port A	Port B
0	0	0	Special Single Chip	General Purpose I/O	General Purpose I/O
0	0	1	Special Expanded Narrow	ADDR[15:8]/DATA[7:0]	ADDR[7:0]
0	1	0	Special Peripheral	ADDR/DATA	ADDR/DATA
0	1	1	Special Expanded Wide	ADDR/DATA	ADDR/DATA
1	0	0	Normal Single Chip	General Purpose I/O	General Purpose I/O
1	0	1	Normal Expanded Narrow	ADDR[15:8]/DATA[7:0]	ADDR[7:0]
1	1	0	Reserved (Forced to Peripheral)	—	—
1	1	1	Normal Expanded Wide	ADDR/DATA	ADDR/DATA

There are two basic types of operating modes:

Normal modes — some registers and bits are protected against accidental changes.

Special modes — allow greater access to protected control registers and bits for special purposes such as testing and emulation.

A system development and debug feature, background debug mode (BDM), is available in all modes. In special single-chip mode, BDM is active immediately after reset.

5.1.1 Normal Operating Modes

These modes provide three operating configurations. Background debugging is available in all three modes, but must first be enabled for some operations by means of a BDM command. BDM can then be made active by another BDM command.

Normal Expanded Wide Mode — This is a normal mode of operation in which the address and data are multiplexed onto ports A and B. ADDR[15:8] and DATA[15:8] are present on port A. ADDR[7:0] and DATA[7:0] are present on port B.

Normal Expanded Narrow Mode — Port A is configured as the high byte of address multiplexed with the 8-bit data bus. Port B is configured as the lower 8-bit address bus. This mode is used for lower cost production systems that use 8-bit wide external EEPROMs or RAMs. Such systems take extra bus cycles to access 16-bit locations but this may be preferred over the extra cost of additional external memory devices.

Normal Single-Chip Mode — There are no external address and data buses in this mode. All pins of ports A, B and E are configured as general-purpose I/O pins. Port E bits 1 and 0 are input-only with internal pullups and the other 22 pins are bidirectional I/O pins that are initially configured as high-impedance inputs. Port E pullups are enabled upon reset; port A and B pullups are disabled upon reset.

5.1.2 Special Operating Modes

There are three special operating modes that correspond to normal operating modes. These operating modes are commonly used in factory testing and system development. In addition, there is a special peripheral mode, in which an external master, such as an I.C. tester, can control the on-chip peripherals.

Special Expanded Wide Mode — This mode can be used for emulation of normal expanded wide mode and emulation of normal single-chip mode and 16-bit data bus. The bus control related pins in PORTE are all configured to serve their bus control output functions rather than general-purpose I/O.

Special Expanded Narrow Mode — This mode can be used for emulation of normal expanded narrow mode. In this mode external 16-bit data is handled as two back-to-back bus cycles, one for the high byte followed by one for the low byte. Internal operations continue to use full 16-bit data paths.

Special Single-Chip Mode — This mode can be used to force the MCU to active BDM mode to allow system debug through the BKGD pin. The MCU does not fetch the reset vector and execute application code as it would in other modes. Instead, the active background mode is in control of CPU execution and BDM firmware is waiting for additional serial commands through the BKGD pin. There are no external address and data buses in this mode. The MCU operates as a stand-alone device and all program and data space are on-chip. External port pins can be used for general-purpose I/O.

Special Peripheral Mode — The CPU is not active in this mode. An external master can control on-chip peripherals for testing purposes. It is not possible to change to or from this mode without going through reset. Background debugging should not be used while the MCU is in special peripheral mode as internal bus conflicts between BDM and the external master can cause improper operation of both modes.

5.2 Background Debug Mode

Background debug mode (BDM) is an auxiliary operating mode that is used for system development. BDM is implemented in on-chip hardware and provides a full set of debug operations. Some BDM commands can be executed while the CPU is operating normally. Other BDM commands are firmware based, and require the BDM firmware to be enabled and active for execution.

In special single-chip mode, BDM is enabled and active immediately out of reset. BDM is available in all other operating modes, but must be enabled before it can be activated. BDM should not be used in special peripheral mode because of potential bus conflicts.

Once enabled, background mode can be made active by a serial command sent via the BKGD pin or execution of a CPU12 BGND instruction. While background mode is active, the CPU can interpret special debugging commands, and read and write CPU registers, peripheral registers, and locations in memory.

While BDM is active, the CPU executes code located in a small on-chip ROM mapped to addresses $FF00 to $FFFF; BDM control registers are accessible at addresses $FF00 to $FF06. The BDM ROM replaces the regular system vectors while BDM is active. While BDM is active, the user memory from $FF00 to $FFFF is not in the map except through serial BDM commands.

BDM allows read and write access to internal memory-mapped registers and RAM, and read access to EEPROM and Flash EEPROM without interrupting the application code executing in the CPU. This non-intrusive mode uses dead bus cycles to access the memory and in most cases will remain cycle deterministic. Refer to **16 Development Support** for more details on BDM.

MODE — Mode Register **$000B**

	Bit 7	6	5	4	3	2	1	Bit 0	
	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	0	EME	
RESET:	1	0	1	1	0	0	–	0	Normal Exp Narrow
RESET:	1	1	1	1	0	0	–	0	Normal Exp Wide
RESET:	0	0	1	1	1	0	–	1	Special Exp Narrow
RESET:	0	1	1	1	1	0	–	1	Special Exp Wide
RESET:	0	1	0	1	1	0	–	1	Peripheral
RESET:	1	0	0	1	0	0	–	0	Normal Single Chip
RESET:	0	0	0	1	1	0	–	1	Special Single Chip

MODE controls the MCU operating mode and various configuration options. This register is not in the map in peripheral mode

SMODN, MODB, MODB — Mode Select Special, B and A

These bits show the current operating mode and reflect the status of the BKGD, MODB and MODA input pins at the rising edge of reset.

Read anytime. SMODN may only be written if SMODN = 0 (in special modes) but the first write is ignored; MODB, MODA may be written once if SMODN = 1; anytime if SMODN = 0, except that special peripheral and reserved modes cannot be selected.

ESTR — E Clock Stretch Enable

Determines if the E Clock behaves as a simple free-running clock or as a bus control signal that is active only for external bus cycles. ESTR is always one in expanded modes since it is required for address demultiplexing and must follow stretched cycles.

0 = E never stretches (always free running).

1 = E stretches high during external access cycles and low during non-visible internal accesses.

Normal modes: write once; Special modes: write anytime, read anytime.

IVIS — Internal Visibility

This bit determines whether internal ADDR/DATA, R/$\overline{W}$, and $\overline{LSTRB}$ signals can be seen on the bus during accesses to internal locations. In special expanded narrow mode, it is possible to configure the MCU to show internal accesses on an external 16-bit bus. The IVIS control bit must be set to 1. When the system is configured this way, visible internal accesses are shown as if the MCU was configured for expanded wide mode but normal external accesses operate as if the bus was in narrow mode. In normal expanded narrow mode, internal visibility is not allowed and IVIS is ignored.

0 = No visibility of internal bus operations on external bus

1 = Internal bus operations are visible on external bus

Normal modes: write once; Special modes: write anytime EXCEPT the first time. Read anytime.

EBSWAI — External Bus Module Stop in Wait Control

This bit controls access to the external bus interface when in Wait mode. The module will delay before shutting down in Wait mode to allow for final bus activity to complete.

0 = External bus and registers continue functioning during Wait mode.

1 = External bus is shut down during Wait mode.

EME — Emulate Port E

Removing the registers from the map allows the user to emulate the function of these registers externally. In single-chip mode PORTE and DDRE are always in the map regardless of the state of this bit.

0 = PORTE and DDRE are in the memory map.

1 = PORTE and DDRE are removed from the internal memory map (expanded mode).

Normal modes: write once; special modes: write anytime EXCEPT the first time. Read anytime.

5.3 Internal Resource Mapping

The internal register block, RAM, Flash EEPROM and EEPROM have default locations within the 64-Kbyte standard address space but may be reassigned to other locations during program execution by setting bits in mapping registers INITRG, INITRM, and INITEE. During normal operating modes these registers can be written once. It is advisable to explicitly establish these resource locations during the initialization phase of program execution, even if default values are chosen, in order to protect the registers from inadvertent modification later.

Writes to the mapping registers go into effect between the cycle that follows the write and the cycle after that. To assure that there are no unintended operations, a write to one of these registers should be followed with a NOP instruction.

If conflicts occur when mapping resources, the register block will take precedence over the other resources; RAM, Flash EEPROM, or EEPROM addresses occupied by the register block will not be available for storage. When active, BDM ROM takes precedence over other resources although a conflict between BDM ROM and register space is not possible. **Table 10** shows resource mapping precedence.

All address space not utilized by internal resources is by default external memory.

Table 10 Mapping Precedence

Precedence	Resource
1	BDM ROM (if active)
2	Register Space
3	RAM
4	EEPROM
5	Flash EEPROM
6	External Memory

5.3.1 Register Block Mapping

After reset the 512 byte register block resides at location $0000 but can be reassigned to any 2-Kbyte boundary within the standard 64-Kbyte address space. Mapping of internal registers is controlled by five bits in the INITRG register. The register block occupies the first 512 bytes of the 2-Kbyte block.

INITRG — Initialization of Internal Register Position Register **$0011**

	Bit 7	6	5	4	3	2	1	Bit 0
	REG15	REG14	REG13	REG12	REG11	0	0	MMSWAI
RESET:	0	0	0	0	0	0	0	0

REG[15:11] — Internal register map position

These bits specify the upper five bits of the 16-bit registers address.

Write once in normal modes or anytime in special modes. Read anytime.

MMSWAI — Memory Mapping Interface Stop in Wait Control

This bit controls access to the memory mapping interface when in Wait mode.

0 = Memory mapping interface continues to function during Wait mode.

1 = Memory mapping interface access is shut down during Wait mode.

RAM Mapping

The MC68HC912BC32 has 1 Kbyte of fully static RAM that is used for storing instructions, variables, and temporary data during program execution. After reset, RAM addressing begins at location $0800 but can be assigned to any 2-Kbyte boundary within the standard 64-Kbyte address space. Mapping of internal RAM is controlled by five bits in the INITRM register. The RAM array occupies the first 1 Kbyte of the 2-Kbyte block.

NITRM — Initialization of Internal RAM Position Register **$0010**

	Bit 7	6	5	4	3	2	1	Bit 0
	RAM15	RAM14	RAM13	RAM12	RAM11	0	0	0
RESET:	0	0	0	0	1	0	0	0

RAM[15:11] — Internal RAM map position

These bits specify the upper five bits of the 16-bit RAM address.

Write once in normal modes or anytime in special modes. Read anytime.

5.3.2 EEPROM Mapping

The MC68HC912BC32 has 768 bytes of EEPROM which is activated by the EEON bit in the INITEE register.

Mapping of internal EEPROM is controlled by four bits in the INITEE register. After reset EEPROM address space begins at location $0D00 but can be mapped to any 4-Kbyte boundary within the standard 64-Kbyte address space.

INITEE— Initialization of Internal EEPROM Position Register **$0012**

	Bit 7	6	5	4	3	2	1	Bit 0
	EE15	EE14	EE13	EE12	0	0	0	EEON
RESET:	0	0	0	0	0	0	0	1

EE[15:12] — Internal EEPROM map position

These bits specify the upper four bits of the 16-bit EEPROM address.

Write once in normal modes or anytime in special modes. Read anytime.

EEON — Internal EEPROM On (Enabled)

The EEON bit allows read access to the EEPROM array. EEPROM control registers can be accessed and EEPROM locations may be programmed or erased regardless of the state of EEON.

This bit is forced to one in single-chip modes. Read or write anytime.

0 = Removes the EEPROM from the map

1 = Places the on-chip EEPROM in the memory map

5.3.3 Flash EEPROM and Expansion Address Mapping

Additional mapping controls are available that can be used in conjunction with Flash EEPROM and memory expansion.

The 32-Kbyte Flash EEPROM can be mapped to either the upper or lower half of the 64-Kbyte address space. When mapping conflicts occur, registers, RAM and EEPROM have priority over Flash EEPROM. To use memory expansion the part must be operated in one of the expanded modes.

MISC — Miscellaneous Mapping Control Register **$0013**

	Bit 7	6	5	4	3	2	1	Bit 0	
	0	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	MAPROM	ROMON	
RESET:	0	0	0	0	1	1	0	0	Ex. mode
	0	0	0	0	1	1	1	1	SIngle-chip mode

This register can be read anytime. In Normal modes MISC can be written once; in Special modes it can be written anytime.

NDRF — Narrow Data Bus for Register-Following Map

This bit enables a narrow bus feature for the 512-byte register-following map. In Expanded Narrow (eight bit) modes, Single Chip modes, and Peripheral mode, NDRF has no effect. The register-following map always begins at the byte following the 512-byte register map. If the registers are moved this space will also move.

0 = Register-following map space acts as a full 16-bit data bus

1 = Register-following map space acts the same as an 8-bit external data bus

RFSTR1, RFSTR0 — Register-Following Stretch Bit 1 and Bit 0

These bits determine the amount of clock stretch on accesses to the 512-byte register-following map. It is valid regardless of the state of the NDRF bit. In Single Chip and Peripheral modes this bit has no meaning or effect.

Table 11 Register-Following Stretch-Bit Definition

Stretch bit RFSTR1	Stretch bit RFSTR0	E Clocks Stretched
0	0	0
0	1	1
1	0	2
1	1	3

EXSTR1, EXSTR0 — External Access Stretch Bit1 and Bit0

These bits determine the amount of clock stretch on accesses to the external address space. In Single Chip and Peripheral modes this bit has no meaning or effect.

Table 12 Expanded Stretch-Bit Definition

Stretch bit EXSTR1	Stretch bit EXSTR0	E Clocks Stretched
0	0	0
0	1	1
1	0	2
1	1	3

MAPROM — Map Location of Flash EEPROM

This bit determines the location of the on-chip Flash EEPROM. In Expanded modes it is reset to zero. In Single Chip modes it is reset to one. If ROMON is zero, this bit has no meaning or effect.

0 = Flash EEPROM is located from $0000 to $7FFF

1 = Flash EEPROM is located from $8000 to $FFFF

ROMON — Enable Flash EEPROM

In Expanded modes ROMON is reset to zero. In Single Chip modes it is reset to one. If the internal RAM, registers, EEPROM, or BDM ROM (if active) are mapped to the same space as the Flash EEPROM, they will have priority over the Flash EEPROM.

0 = Disables the Flash EEPROM in the memory map

1 = Enables the Flash EEPROM in the memory map

5.4 Memory Maps

The following diagrams illustrate the memory map for each mode of operation immediately after reset.

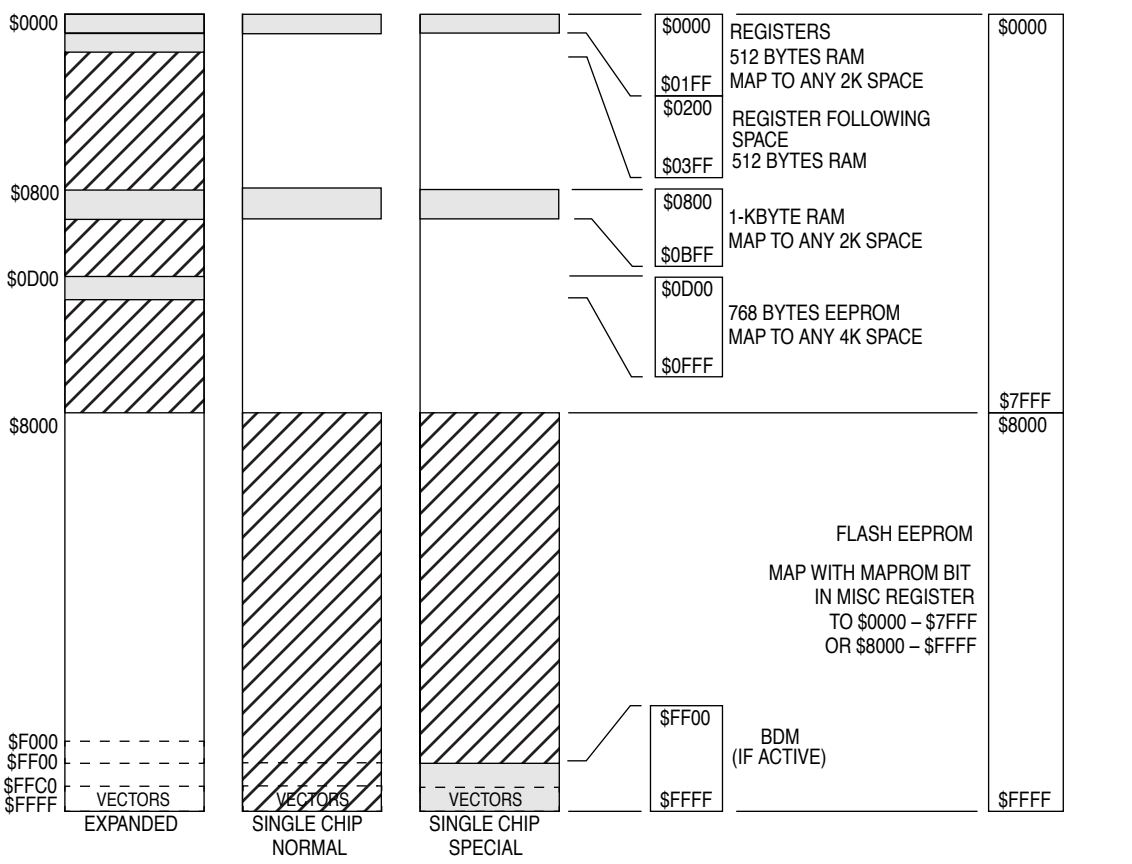


Figure 6 MC68HC912BC32 Memory Map

6 Bus Control and Input/Output

Internally the MC68HC912BC32 has full 16-bit data paths, but depending upon the operating mode and control registers, the external bus may be 8 or 16 bits. There are cases where 8-bit and 16-bit accesses can appear on adjacent cycles using the $\overline{\text{LSTRB}}$ signal to indicate 8- or 16-bit data.

6.1 Detecting Access Type from External Signals

The external signals $\overline{\text{LSTRB}}$, R/$\overline{\text{W}}$, and A0 can be used to determine the type of bus access that is taking place. Accesses to the internal RAM module are the only type of access that produce $\overline{\text{LSTRB}}$=A0=1, because the internal RAM is specifically designed to allow misaligned 16-bit accesses in a single cycle. In these cases the data for the address that was accessed is on the low half of the data bus and the data for address + 1 is on the high half of the data bus (data order is swapped).

Table 13 Access Type vs. Bus Control Pins

$\overline{\text{LSTRB}}$	A0	R/$\overline{\text{W}}$	Type of Access
1	0	1	8-bit read of an even address
0	1	1	8-bit read of an odd address
1	0	0	8-bit write of an even address
0	1	0	8-bit write of an odd address
0	0	1	16-bit read of an even address
1	1	1	16-bit read of an odd address (low/high data swapped)
0	0	0	16-bit write to an even address
1	1	0	16-bit write to an even address (low/high data swapped)

6.2 Registers

Not all registers are visible in the MC68HC912BC32 memory map under certain conditions. In special peripheral mode the first 16 registers associated with bus expansion are removed from the memory map.

In expanded modes, some or all of port A, port B, and port E are used for expansion buses and control signals. In order to allow emulation of the single-chip functions of these ports, some of these registers must be rebuilt in an external port replacement unit. In any expanded mode port A and port B are used for address and data lines so registers for these ports, as well as the data direction registers for these ports, are removed from the on-chip memory map and become external accesses.

In any expanded mode, port E pins may be needed for bus control (e.g., ECLK, R/$\overline{\text{W}}$). To regain the single-chip functions of port E, the emulate port E (EME) control bit in the MODE register may be set. In this special case of expanded mode and EME set, PORTE and DDRE registers are removed from the on-chip memory map and become external accesses so port E may be rebuilt externally.

PORTA — Port A Register **$0000**

	Bit 7	6	5	4	3	2	1	Bit 0
Single Chip	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
RESET:	–	–	–	–	–	–	–	–
Exp Wide & Periph:	ADDR15 DATA15	ADDR14 DATA14	ADDR13 DATA13	ADDR12 DATA12	ADDR11 DATA11	ADDR10 DATA10	ADDR9 DATA9	ADDR8 DATA8
Expanded Narrow	ADDR15 DATA15/7	ADDR14 DATA14/6	ADDR13 DATA13/5	ADDR12 DATA12/4	ADDR11 DATA11/3	ADDR10 DATA10/2	ADDR9 DATA9/1	ADDR8 DATA8/0

Bits PA[7:0] are associated with addresses ADDR[15:8] and DATA[15:8]. When this port is not used for external addresses and data, such as in single-chip mode, these pins can be used as general-purpose I/O. DDRA determines the primary direction of each pin. This register is not in the on-chip map in expanded and peripheral modes. Read and write anytime.

DDRA — Port A Data Direction Register **$0002**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
RESET:	0	0	0	0	0	0	0	0

This register determines the primary direction for each port A pin when functioning as a general-purpose I/O port. DDRA is not in the on-chip map in expanded and peripheral modes. Read and write anytime.

0 = Associated pin is a high-impedance input
1 = Associated pin is an output

PORTB — Port B Register **$0001**

	Bit 7	6	5	4	3	2	1	Bit 0
Single Chip	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:	–	–	–	–	–	–	–	–
Exp Wide & Periph:	ADDR7 DATA7	ADDR6 DATA6	ADDR5 DATA5	ADDR4 DATA4	ADDR3 DATA3	ADDR2 DATA2	ADDR1 DATA1	ADDR0 DATA0
Expanded Narrow	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0

Bits PB[7:0] are associated with addresses ADDR[7:0] and DATA[7:0]. When this port is not used for external addresses and data such as in single-chip mode, these pins can be used as general-purpose I/O. DDRB determines the primary direction of each pin. This register is not in the on-chip map in expanded and peripheral modes. Read and write anytime.

DDRB — Port B Data Direction Register **$0003**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
RESET:	0	0	0	0	0	0	0	0

This register determines the primary direction for each port B pin when functioning as a general-purpose I/O port. DDRB is not in the on-chip map in expanded and peripheral modes. Read and write anytime.

0 = Associated pin is a high-impedance input
1 = Associated pin is an output

PORTE — Port E Register **$0008**

	Bit 7	6	5	4	3	2	1	Bit 0
Single Chip	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:	–	–	–	–	–	–	–	–
Alt. Pin Function	$\overline{DBE}$	MODB or IPIPE1	MODA or IPIPE0	ECLK	$\overline{LSTRB}$ or $\overline{TAGLO}$	R/$\overline{W}$	$\overline{IRQ}$	$\overline{XIRQ}$

This register is associated with external bus control signals and interrupt inputs including data bus enable ($\overline{DBE}$), mode select (MODB/IPIPE1, MODA/IPIPE0), E clock, data size ($\overline{LSTRB}$/$\overline{TAGLO}$), read/write (R/$\overline{W}$), $\overline{IRQ}$, and $\overline{XIRQ}$. When the associated pin is not used for one of these specific functions, the pin can be used as general-purpose I/O. The port E assignment register (PEAR) selects the function of each pin. DDRE determines the primary direction of each port E pin when configured to be general-purpose I/O.

Some of these pins have software selectable pull-ups ($\overline{DBE}$, $\overline{LSTRB}$, R/$\overline{W}$, and $\overline{XIRQ}$). A single control bit enables the pull-ups for all these pins which are configured as inputs. $\overline{IRQ}$ always has a pull-up.

This register is not in the map in peripheral mode or expanded modes when the EME bit is set.

Read and write anytime.

DDRE — Port E Data Direction Register **$0009**

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

This register determines the primary direction for each port E pin configured as general-purpose I/O.

0 = Associated pin is a high-impedance input
1 = Associated pin is an output

PE[1:0] are associated with $\overline{XIRQ}$ and $\overline{IRQ}$ and cannot be configured as outputs. These pins can be read regardless of whether the alternate interrupt functions are enabled.

This register is not in the map in peripheral mode and expanded modes while the EME control bit is set.

Read and write anytime.

PEAR — Port E Assignment Register **$000A**

	Bit 7	6	5	4	3	2	1	Bit 0	
	NDBE	0	PIPOE	NECLK	LSTRE	RDWE	0	0	
RESET:	0	–	0	0	0	0	–	–	Normal Expanded
RESET:	0	–	1	0	1	1	–	–	Special Expanded
RESET:	1	–	0	1	0	0	–	–	Peripheral
RESET:	1	–	0	1	0	0	–	–	Normal Single Chip
RESET:	0	–	1	0	1	1	–	–	Special Single Chip

The PEAR register is used to choose between the general-purpose I/O functions and the alternate bus control functions of port E. When an alternate control function is selected, the associated DDRE bits are overridden.

The reset condition of this register depends on the mode of operation because bus-control signals are needed immediately after reset in some modes.

In normal single-chip mode, no external bus control signals are needed so all of port E is configured for general-purpose I/O.

In special single-chip mode, the E clock is enabled as a timing reference and the other bits of port E are configured for general-purpose I/O.

In normal expanded modes, the reset vector is located in external memory. The E clock may be required for this access but R/$\overline{W}$ is only needed by the system when there are external writable resources. Therefore in normal expanded modes, only the E clock is configured for its alternate bus control function and the other bits of port E are configured for general-purpose I/O. If the normal expanded system needs any other bus-control signals, PEAR would need to be written before any access that needed the additional signals.

In special expanded modes, IPIPE1, IPIPE0, E, R/$\overline{W}$, and $\overline{LSTRB}$ are configured as bus-control signals.

In peripheral mode, the PEAR register is not accessible for reads or writes.

NDBE — No Data Bus Enable

Read and write anytime.

0 = PE7 is used for external control of data enables on memories.

1 = PE7 is used for general-purpose I/O.

PIPOE — Pipe Signal Output Enable

Normal: write once; Special: write anytime except the first time. Read anytime.This bit has no effect in single chip modes.

0 = PE[6:5] are general-purpose I/O.

1 = PE[6:5] are outputs and indicate the state of the instruction queue.

NECLK — No External E Clock

In expanded modes, writes to this bit have no effect. E clock is required for de-multiplexing the external address; NECLK will remain zero in expanded modes. NECLK can be written once in normal single chip mode and can be written anytime in special single chip mode. The bit can be read anytime.

0 = PE4 is the external E-clock pin subject to the following limitation: In single-chip modes, PE4 is general-purpose I/O unless NECLK = 0 and either IVIS = 1 or ESTR = 0. A 16-bit write to PEAR:MODE can configure all three bits in one operation.

1 = PE4 is a general-purpose I/O pin.

LSTRE — Low Strobe ($\overline{\text{LSTRB}}$) Enable

Normal: write once; Special: write anytime except the first time. Read anytime. This bit has no effect in single-chip modes or normal expanded narrow mode.

0 = PE3 is a general-purpose I/O pin.

1 = PE3 is configured as the $\overline{\text{LSTRB}}$ bus-control output, provided the MCU is not in single chip or normal expanded narrow modes.

$\overline{\text{LSTRB}}$ is used during external writes. After reset in normal expanded mode, $\overline{\text{LSTRB}}$ is disabled. If needed, it should be enabled before external writes. External reads do not normally need $\overline{\text{LSTRB}}$ because all 16 data bits can be driven even if the MCU only needs 8 bits of data.

$\overline{\text{TAGLO}}$ is a shared function of the PE3/$\overline{\text{LSTRB}}$ pin. In special expanded modes with LSTRE set and the BDM instruction tagging on, a zero at the falling edge of E tags the instruction word low byte being read into the instruction queue.

RDWE — Read/Write Enable

Normal: write once; Special: write anytime except the first time. Read anytime. This bit has no effect in single-chip modes.

0 = PE2 is a general-purpose I/O pin.

1 = PE2 is configured as the R/$\overline{\text{W}}$ pin. In single chip modes, RDWE has no effect and PE2 is a general-purpose I/O pin.

R/$\overline{\text{W}}$ is used for external writes. After reset in normal expanded mode, it is disabled. If needed it should be enabled before any external writes.

PUCR — Pull Up Control Register **$000C**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	PUPE	0	0	PUPB	PUPA
RESET:	0	0	0	1	0	0	0	0

These bits select pull-up resistors for any pin in the corresponding port that is currently configured as an input. This register is not in the map in peripheral mode.

Read and write anytime.

PUPE — Pull-Up Port E Enable

Pin PE1 always has a pull-up. Pins PE6, PE5, and PE4 never have pull-ups.

0 = Port E pull-ups on PE7, PE3, PE2, and PE0 are disabled.

1 = Enable pull-up devices for port E input pins PE7, PE3, PE2, and PE0.

PUPB — Pull-Up Port B Enable

0 = Port B pull-ups are disabled.

1 = Enable pull-up devices for all port B input pins.

This bit has no effect if port B is being used as part of the address/data bus (the pull-ups are inactive).

PUPA — Pull-Up Port A Enable

0 = Port A pull-ups are disabled.

1 = Enable pull-up devices for all port A input pins.

This bit has no effect if port A is being used as part of the address/data bus (the pull-ups are inactive).

RDRIV — Reduced Drive of I/O Lines **$000D**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	RDPE	0	RDPB	RDPA
RESET:	0	0	0	0	0	0	0	0

These bits select reduced drive for the associated port pins. This gives reduced power consumption and reduced RFI with a slight increase in transition time (depending on loading). The reduced drive function is independent of which function is being used on a particular port. This register is not in the map in peripheral mode.
Normal: write once; Special: write anytime except the first time. Read anytime.

RDPE — Reduced Drive of Port E
0 = All port E output pins have full drive enabled.
1 = All port E output pins have reduced drive capability.

RDPB — Reduced Drive of Port B
0 = All port B output pins have full drive enabled.
1 = All port B output pins have reduced drive capability.

RDPA — Reduced Drive of Port A
0 = All port A output pins have full drive enabled.
1 = All port A output pins have reduced drive capability.

7 Flash EEPROM

The 32-Kbyte Flash EEPROM module for the MC68HC912BC32 serves as electrically erasable and programmable, non-volatile ROM emulation memory. The module can be used for program code that must either execute at high speed or is frequently executed, such as operating system kernels and standard subroutines, or it can be used for static data which is read frequently. The Flash EEPROM is ideal for program storage for single-chip applications allowing for field reprogramming.

7.1 Overview

The Flash EEPROM array is arranged in a 16-bit configuration and may be read as either bytes, aligned words or misaligned words. Access time is one bus cycle for byte and aligned word access and two bus cycles for misaligned word operations.

The Flash EEPROM module requires an external program/erase voltage (V_{FP}) to program or erase the Flash EEPROM array. The external program/erase voltage is provided to the Flash EEPROM module via an external V_{FP} pin. To prevent damage to the flash array, V_{FP} should always be greater than or equal to V_{DD}–0.5V. Programming is by byte or aligned word. The Flash EEPROM module supports bulk erase only.

The Flash EEPROM module has hardware interlocks which protect stored data from accidental corruption. An erase- and program-protected 2-Kbyte block for boot routines is located at $7800–$7FFF or $F800–$FFFF depending upon the mapped location of the Flash EEPROM array. (The protected boot block on the initial mask set 0H18J is 1-Kbyte and is located at $7C00–$7FFF or $FC00–$FFFF.)

7.2 Flash EEPROM Control Block

A 4-byte register block controls the Flash EEPROM module operation. Configuration information is specified and programmed independently from the contents of the Flash EEPROM array. At reset, the 4-byte register section starts at address $00F4.

7.3 Flash EEPROM Array

After reset, the Flash EEPROM array is located from addresses $8000 to $FFFF in single-chip mode. In Expanded modes the Flash EEPROM array is located from address $0000 to $7FFF, however, it is turned off. The Flash EEPROM can be mapped to an alternate address range. See **5 Operating Modes and Resource Mapping**.

7.4 Flash EEPROM Registers

FEELCK — Flash EEPROM Lock Control Register **$00F4**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	0	LOCK
RESET:	0	0	0	0	0	0	0	0

In normal modes the LOCK bit can only be written once after reset.

LOCK — Lock Register Bit
0 = Enable write to FEEMCR register
1 = Disable write to FEEMCR register

FEEMCR — Flash EEPROM Module Configuration Register **$00F5**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	0	BOOTP
RESET:	0	0	0	0	0	0	0	1

This register controls the operation of the Flash EEPROM array. BOOTP cannot be changed when the LOCK control bit in the FEELCK register is set or if ENPE in the FEECTL register is set.

BOOTP — Boot Protect

The boot block is located at $7800–$7FFF or $F800–$FFFF depending upon the mapped location of the Flash EEPROM array and mask set ($7C00–$7FFF or $FC00–$FFFF for 1-Kbyte block).

0 = Enable erase and program of 1-Kbyte or 2-Kbyte boot block
1 = Disable erase and program of 1-Kbyte or 2-Kbyte boot block

FEETST — Flash EEPROM Module Test Register **$00F6**

	Bit 7	6	5	4	3	2	1	Bit 0
	FSTE	GADR	HVT	FENLV	FDISVFP	VTCK	STRE	MWPR
RESET:	0	0	0	0	0	0	0	0

In normal mode, writes to FEETST control bits have no effect and always read zero. The Flash EEPROM module cannot be placed in test mode inadvertently during normal operation.

FSTE — Stress Test Enable

0 = Disables the gate/drain stress circuitry
1 = Enables the gate/drain stress circuitry

GADR — Gate/Drain Stress Test Select

0 = Selects the drain stress circuitry
1 = Selects the gate stress circuitry

HVT — Stress Test High Voltage Status

0 = High voltage not present during stress test
1 = High voltage present during stress test

FENLV — Enable Low Voltage

0 = Disables low voltage transistor in current reference circuit
1 = Enables low voltage transistor in current reference circuit

FDISVFP — Disable Status V_{FP} Voltage Lock

When the V_{FP} pin is below normal programming voltage the Flash module will not allow writing to the LAT bit; the user cannot erase or program the Flash module. The FDISVFP control bit enables writing to the LAT bit regardless of the voltage on the V_{FP} pin.

0 = Enable the automatic lock mechanism if V_{FP} is low
1 = Disable the automatic lock mechanism if V_{FP} is low

VTCK — V_T Check Test Enable

When VTCK is set, the Flash EEPROM module uses the V_{FP} pin to control the control gate voltage; the sense amp time-out path is disabled. This allows for indirect measurements of the bit cells program and erase threshold. If $V_{FP} < V_{ZBRK}$ (breakdown voltage) the control gate will equal the V_{FP} voltage.

If $V_{FP} > V_{ZBRK}$ the control gate will be regulated by the following equation:

$$\text{Vcontrol gate} = V_{ZBRK} + 0.44 \times (V_{FP} - V_{ZBRK})$$

0 = V_T test disable

1 = V_T test enable

STRE — Spare Test Row Enable

The spare test row consists of one Flash EEPROM array row. The reserved word at location 31 contains production test information which must be maintained through several erase cycles. When STRE is set, the decoding for the spare test row overrides the address lines which normally select the other rows in the array.

0 = LIB accesses are to the Flash EEPROM array

1 = Spare test row in array enabled if SMOD is active

MWPR — Multiple Word Programming

Used primarily for testing, if MPWR = 1, the two least-significant address lines ADDR[1:0] will be ignored when programming a Flash EEPROM location. The word location addressed if ADDR[1:0] = 00, along with the word location addressed if ADDR[1:0] = 10, will both be programmed with the same word data from the programming latches. This bit should not be changed during programming.

0 = Multiple word programming disabled

1 = Program 32 bits of data

FEECTL — Flash EEPROM Control Register **$00F7**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	FEESWAI	SVFP	ERAS	LAT	ENPE
RESET:	0	0	0	0	0	0	0	0

This register controls the programming and erasure of the Flash EEPROM.

FEESWAI — Flash EEPROM Stop in Wait Control

0 = Do not halt Flash EEPROM clock when the part is in wait mode.

1 = Halt Flash EEPROM clock when the part is in wait mode.

NOTE

The FEESWAI bit cannot be asserted if the interrupt vector resides in the Flash EEPROM array.

SVFP — Status V_{FP} Voltage

SVFP is a read only bit.

0 = Voltage of V_{FP} pin is below normal programming voltage levels

1 = Voltage of V_{FP} pin is above normal programming voltage levels

ERAS — Erase Control

This bit can be read anytime or written when ENPE = 0. When set, all locations in the array will be erased at the same time. The boot block will be erased only if BOOTP = 0. This bit also affects the result of attempted array reads. See **Table 14** for more information. Status of ERAS cannot change if ENPE is set.

0 = Flash EEPROM configured for programming

1 = Flash EEPROM configured for erasure

LAT — Latch Control

This bit can be read anytime or written when ENPE = 0. When set, the Flash EEPROM is configured for programming or erasure and, upon the next valid write to the array, the address and data will be latched for the programming sequence. See **Table 14** for the effects of LAT on array reads. A high voltage detect circuit on the V_{FP} pin will prevent assertion of the LAT bit when the programming voltage is at normal levels.

0 = Programming latches disabled
1 = Programming latches enabled

ENPE — Enable Programming/Erase

0 = Disables program/erase voltage to Flash EEPROM
1 = Applies program/erase voltage to Flash EEPROM

ENPE can be asserted only after LAT has been asserted and a write to the data and address latches has occurred. If an attempt is made to assert ENPE when LAT is negated, or if the latches have not been written to after LAT was asserted, ENPE will remain negated after the write cycle is complete.
The LAT, ERAS and BOOTP bits cannot be changed when ENPE is asserted. A write to FEECTL may only affect the state of ENPE. Attempts to read a Flash EEPROM array location in the Flash EEPROM module while ENPE is asserted will not return the data addressed. See **Table 14** for more information. Flash EEPROM module control registers may be read or written while ENPE is asserted. If ENPE is asserted and LAT is negated on the same write access, no programming or erasure will be performed.

Table 14 Effects of ENPE, LAT and ERAS on Array Reads

ENPE	LAT	ERAS	Result of Read
0	0	–	Normal read of location addressed
0	1	0	Read of location being programmed
0	1	1	Normal read of location addressed
1	–	–	Read cycle is ignored

7.5 Operation

The Flash EEPROM can contain program and data. On reset, it can operate as a bootstrap memory to provide the CPU with internal initialization information during the reset sequence.

7.5.1 Bootstrap Operation Single-Chip Mode

After reset, the CPU controlling the system will begin booting up by fetching the first program address from address $FFFE.

7.5.2 Normal Operation

The Flash EEPROM allows a byte or aligned word read/write in one bus cycle. Misaligned word read/write require an additional bus cycle. The Flash EEPROM array responds to read operations only. Write operations are ignored.

7.5.3 Program/Erase Operation

An unprogrammed Flash EEPROM bit has a logic state of one. A bit must be programmed to change its state from one to zero. Erasing a bit returns it to a logic one. The Flash EEPROM has a minimum program/erase life of 100 cycles. Programming or erasing the Flash EEPROM is accomplished by a series of control register writes and a write to a set of programming latches.

Programming is restricted to a single byte or aligned word at a time as determined by internal signal SZ8 and ADDR[0]. The Flash EEPROM must first be completely erased prior to programming final data values. It is possible to program a location in the Flash EEPROM without erasing the entire array if the new value does not require the changing of bit values from zero to one.

Read/Write Accesses During Program/Erase — During program or erase operations, read and write accesses may be different from those during normal operation and are affected by the state of the control bits in the Flash EEPROM control register (FEECTL). The next write to any valid address to the array after LAT is set will cause the address and data to be latched into the programming latches. Once the address and data are latched, write accesses to the array will be ignored while LAT is set. Writes to the control registers will occur normally.

Program/Erase Verification — When programming or erasing the Flash EEPROM array, a special verification method is required to ensure that the program/erase process is reliable, and also to provide the longest possible life expectancy. This method requires stopping the program/erase sequence at periods of t_{PPULSE} (t_{EPULSE} for erasing) to determine if the Flash EEPROM is programmed/erased. After the location reaches the proper value, it must continue to be programmed/erased with additional margin pulses to ensure that it will remain programmed/erased. Failure to provide the margin pulses could lead to corrupted or unreliable data.

Program/Erase Sequence — To begin a program or erase sequence the external V_{FP} voltage must be applied and stabilized. The ERAS bit must be set or cleared, depending on whether a program sequence or an erase sequence is to occur. The LAT bit will be set to cause any subsequent data written to a valid address within the Flash EEPROM to be latched into the programming address and data latches. The next Flash array write cycle must be either to the location that is to be programmed if a programming sequence is being performed, or, if erasing, to any valid Flash EEPROM array location. Writing the new address and data information to the Flash EEPROM is followed by assertion of ENPE to turn on the program/erase voltage to program/erase the new location(s). The LAT bit must be asserted and the address and data latched to allow the setting of the ENPE control bit. If the data and address have not been latched, an attempt to assert ENPE will be ignored and ENPE will remain negated after the write cycle to FEECTL is completed. The LAT bit must remain asserted and the ERAS bit must remain in its current state as long as ENPE is asserted. A write to the LAT bit to clear it while ENPE is set will be ignored. That is, after the write cycle, LAT will remain asserted. Likewise, an attempt to change the state of ERAS will be ignored and the state of the ERAS bit will remain unchanged.

The programming software is responsible for all timing during a program sequence. This includes the total number of program pulses (n_{PP}), the length of the program pulse (t_{PPULSE}), the program margin pulses (p_m) and the delay between turning off the high voltage and verifying the operation (t_{VPROG}).

The erase software is responsible for all timing during an erase sequence. This includes the total number of erase pulses (e_m), the length of the erase pulse (t_{EPULSE}), the erase margin pulse or pulses, and the delay between turning off the high voltage and verifying the operation (t_{VERASE}).

Software also controls the supply of the proper program/erase voltage to the V_{FP} pin, and should be at the proper level before ENPE is set during a program/erase sequence.

A program/erase cycle should not be in progress when starting another program/erase, or while attempting to read from the array.

NOTE

Although clearing ENPE disables the program/erase voltage (V_{FP}) from the V_{FP} pin to the array, care must be taken to ensure that V_{FP} is at V_{DD} whenever programming/erasing is not in progress. Not doing so could damage the part. Ensuring that V_{FP} is always greater or equal to V_{DD} can be accomplished by controlling the V_{FP} power supply with the programming software via an output pin. Alternatively, all programming and erasing can be done prior to installing the device on an application circuit board which can always connect V_{FP} to V_{DD}. Programming can also be accomplished by plugging the board into a special programming fixture which provides program/erase voltage to the V_{FP} pin.

7.6 Programming the Flash EEPROM

Programming the Flash EEPROM is accomplished by the following sequence. The V_{FP} pin voltage must be at the proper level prior to executing step 4 the first time.

1. Apply program/erase voltage to the V_{FP} pin.
2. Clear ERAS and set the LAT bit in the FEECTL register to establish program mode and enable programming address and data latches.
3. Write data to a valid address. The address and data is latched. If BOOTP is asserted, an attempt to program an address in the boot block will be ignored.
4. Apply programming voltage by setting ENPE.
5. Delay for one programming pulse (t_{PPULSE}).
6. Remove programming voltage by clearing ENPE.
7. Delay while high voltage is turning off (t_{VPROG}).
8. Read the address location to verify that it has been programmed
   - If the location is not programmed, repeat steps 4 through 7 until the location is programmed or until the specified maximum number of program pulses has been reached (n_{PP})
   - If the location is programmed, repeat the same number of pulses as required to program the location. This provides 100% program margin.
9. Read the address location to verify that it remains programmed.
10. Clear LAT.
11. If there are more locations to program, repeat steps 2 through 10.
12. Turn off V_{FP} (reduce voltage on V_{FP} pin to V_{DD}).

The flowchart in **Figure 7** demonstrates the recommended programming sequence.

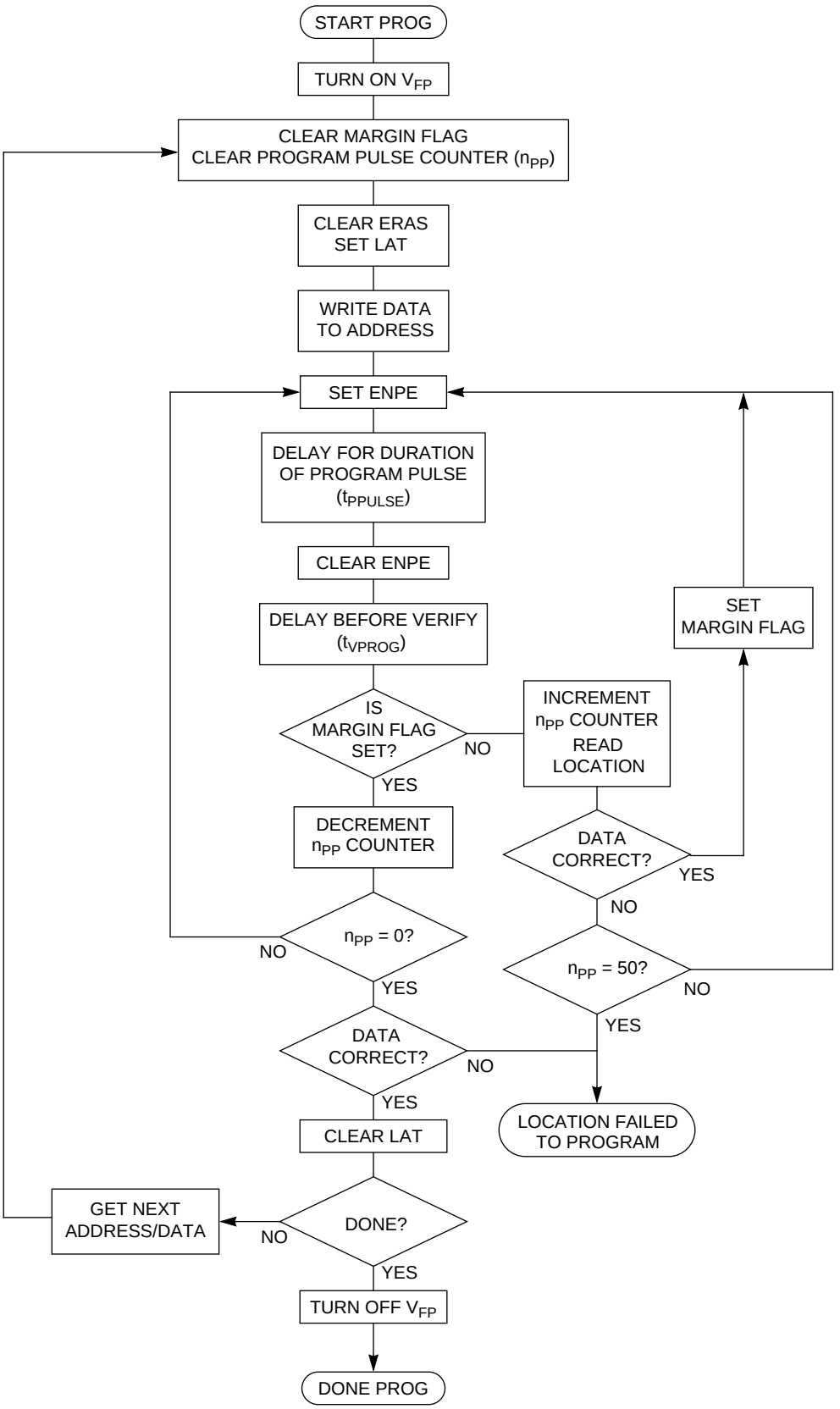


Figure 7 Program Sequence Flow

7.7 Erasing the Flash EEPROM

The following sequence demonstrates the recommended procedure for erasing the Flash EEPROM. The V_{FP} pin voltage must be at the proper level prior to executing step 4 the first time.

1. Turn on V_{FP} (apply program/erase voltage to the V_{FP} pin).
2. Set the LAT bit and ERAS bit to configure the Flash EEPROM for erasing.
3. Write to any valid address in the Flash array. This allows the erase voltage to be turned on; the data written and the address written are not important. The boot block will be erased only if the control bit BOOTP is negated.
4. Apply erase voltage by setting ENPE.
5. Delay for a single erase pulse (t_{EPULSE}).
6. Remove erase voltage by clearing ENPE.
7. Delay while high voltage is turning off (t_{VERASE}).
8. Read the entire array to ensure that the Flash EEPROM is erased.
   - If all of the Flash EEPROM locations are not erased, repeat steps 4 through 7 until either the remaining locations are erased, or until the maximum erase pulses have been applied (n_{EP})
   - If all of the Flash EEPROM locations are erased, repeat the same number of pulses as required to erase the array. This provides 100% erase margin.
9. Read the entire array to ensure that the Flash EEPROM is erased.
10. Clear LAT.
11. Turn off V_{FP} (reduce voltage on V_{FP} pin to V_{DD}).

The flowchart in **Figure 8** demonstrates the recommended erase sequence.

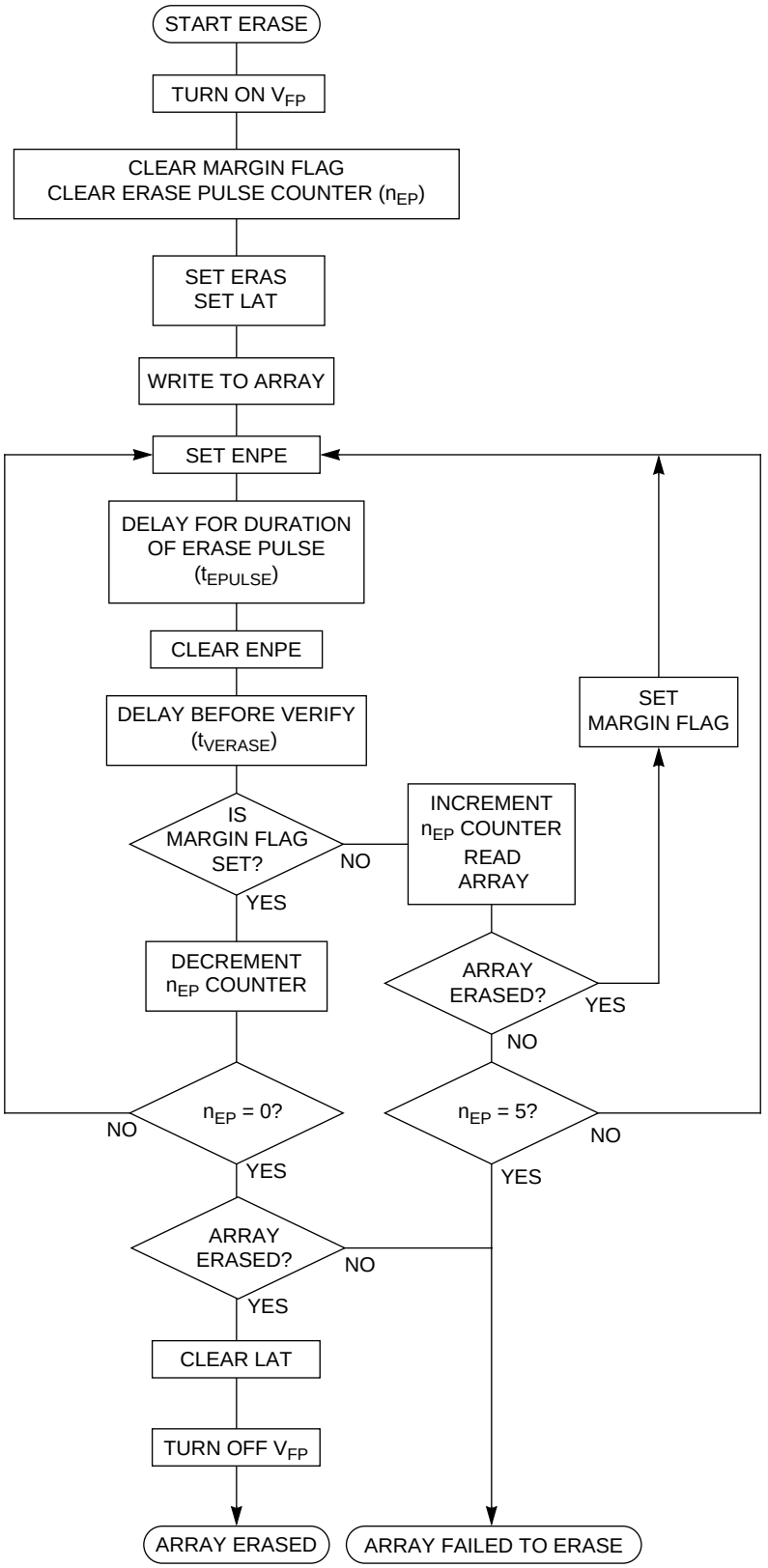


Figure 8 Erase Sequence Flow

7.8 Program/Erase Protection Interlocks

The Flash EEPROM program and erase mechanisms provide maximum protection from accidental programming or erasure.

The voltage required to program/erase the Flash EEPROM (V_{FP}) is supplied via an external pin. If V_{FP} is not present, no programming/erasing will occur. Furthermore, the program/erase voltage will not be applied to the Flash EEPROM unless turned on by setting a control bit (ENPE). The ENPE bit may not be set unless the programming address and data latches have been written previously with a valid address. The latches may not be written unless enabled by setting a control bit (LAT). The LAT and ENPE control bits must be written on separate writes to the control register (FEECTL) and must be separated by a write to the programming latches. The ERAS and LAT bits are also protected when ENPE is set. This prevents inadvertent switching between erase/program mode and also prevents the latched data and address from being changed after a program cycle has been initiated.

7.9 Stop or Wait Mode

When stop or wait commands are executed, the MCU puts the Flash EEPROM in stop or wait mode. In these modes the Flash module will cease erasure or programming immediately. It is advised not to enter stop or wait modes when programming the Flash array.

CAUTION

The Flash EEPROM module is not able to recover from STOP without a 1 microsecond delay. This cannot be controlled internal to the MCU. Therefore, do not attempt to recover from STOP with an interrupt. Use RESET to recover from a STOP mode executed from Flash EEPROM. Recovery from a STOP instruction executed from EEPROM and RAM operate normally.

7.10 Test Mode

The Flash EEPROM has some special test functions which are only accessible when the device is in test mode. Test mode is indicated to the Flash EEPROM module when the SMOD line on the LIB is asserted. When SMOD is asserted, the special test control bits may be accessed via the LIB to invoke the special test functions in the Flash EEPROM module. When SMOD is not asserted, writes to the test control bits have no effect and all bits in the test register FEETST will be cleared. This ensures that Flash EEPROM test mode cannot be invoked inadvertently during normal operation.

Note that the Flash EEPROM module will operate normally, even if SMOD is asserted, until a special test function is invoked. The test mode adds additional features over normal mode which allow the tests to be performed even after the device is installed in the final product.

8 EEPROM

The MC68HC912BC32 EEPROM serves as a 768-byte nonvolatile memory which can be used for frequently accessed static data or as fast access program code.

The MC68HC912BC32 EEPROM is arranged in a 16-bit configuration. The EEPROM array may be read as either bytes, aligned words or misaligned words. Access times is one bus cycle for byte and aligned word access and two bus cycles for misaligned word operations.

Programming is by byte or aligned word. Attempts to program or erase misaligned words will fail. Only the lower byte will be latched and programmed or erased. Programming and erasing of the user EEPROM can be done in all operating modes.

Each EEPROM byte or aligned word must be erased before programming. The EEPROM module supports byte, aligned word, row (32 bytes) or bulk erase, all using the internal charge pump. Bulk erasure of odd and even rows is also possible in test modes; the erased state is $FF. The EEPROM module has hardware interlocks which protect stored data from corruption by accidentally enabling the program/erase voltage. Programming voltage is derived from the internal V_{DD} supply with an internal charge pump. The EEPROM has a minimum program/erase life of 10,000 cycles over the complete operating temperature range.

8.1 EEPROM Programmer's Model

The EEPROM module consists of two separately addressable sections. The first is a four-byte memory mapped control register block used for control, testing and configuration of the EEPROM array. The second section is the EEPROM array itself.

At reset, the four-byte register section starts at address $00F0 and the EEPROM array is located from addresses $0D00 to $0FFF (see **Figure 9**). For information on remapping the register block and EEPROM address space, refer to **5 Operating Modes and Resource Mapping**.

Read access to the memory array section can be enabled or disabled by the EEON control bit in the INITEE register. This feature allows the access of memory mapped resources that have lower priority than the EEPROM memory array. EEPROM control registers can be accessed and EEPROM locations may be programmed or erased regardless of the state of EEON.

Using the normal EEPROG control, it is possible to continue program/erase operations during WAIT. For lowest power consumption during WAIT, stop program/erase by turning off EEPGM.

If the STOP mode is entered during programming or erasing, program/erase voltage will be automatically turned off and the RC clock (if enabled) is stopped. However, the EEPGM control bit will remain set. When STOP mode is terminated, the program/erase voltage will be automatically turned back on if EEPGM is set.

At bus frequencies below 1 MHz, the RC clock must be turned on for program/erase.

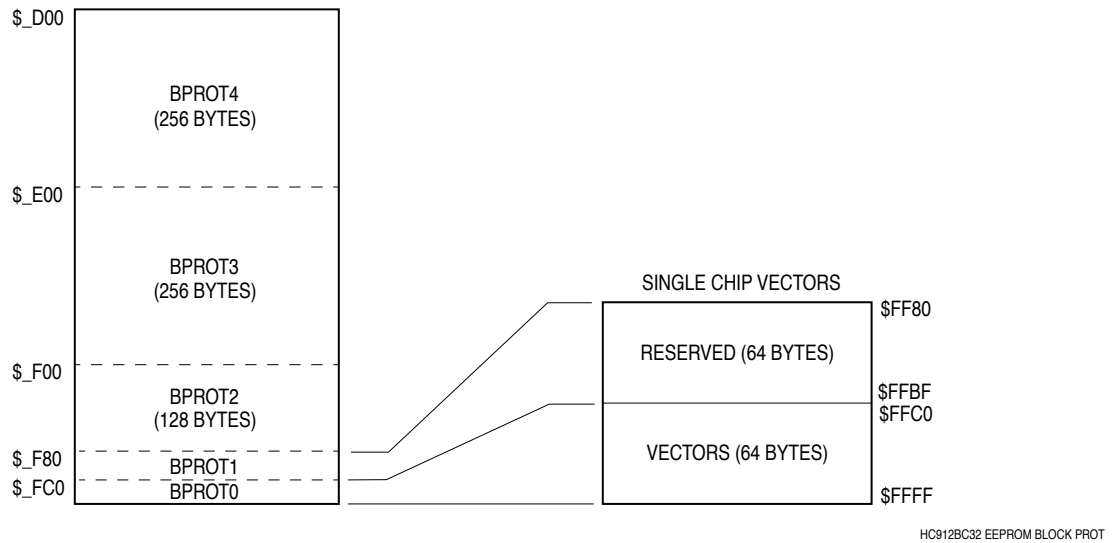


Figure 9 EEPROM Block Protect Mapping

8.2 EEPROM Control Registers

EEMCR — EEPROM Module Configuration **$00F0**

	Bit 7	6	5	4	3	2	1	Bit 0
	1	1	1	1	1	EESWAI	PROTLCK	EERC
RESET:	1	1	1	1	1	1	0	0

EESWAI — EEPROM Stops in Wait Mode
- 0 = Module is not affected during Wait mode
- 1 = Module ceases to be clocked during Wait mode

This bit should be cleared if the Wait mode vectors are mapped in the EEPROM array.

PROTLCK — Block Protect Write Lock
- 0 = Block protect bits and bulk erase protection bit can be written
- 1 = Block protect bits are locked

Read anytime. Write once in normal modes (SMODN = 1), set and clear any time in special modes (SMODN = 0).

EERC — EEPROM Charge Pump Clock
- 0 = System clock is used as clock source for the internal charge pump. Internal RC oscillator is stopped.
- 1 = Internal RC oscillator drives the charge pump. The RC oscillator is required when the system bus clock is lower than f_{PROG}.

Read and write anytime.

EEPROT — EEPROM Block Protect **$00F1**

	Bit 7	6	5	4	3	2	1	Bit 0
	STPROT	1	1	BPROT4	BPROT3	BPROT2	BPROT1	BPROT0
RESET:	1	1	1	1	1	1	1	1

Prevents accidental writes to EEPROM. Read anytime. Write anytime if EEPGM = 0 and PROTLCK = 0.

STPROT — Shadow and Test Row Protection
0 = Shadow and test rows can be programmed and erased.
1 = Shadow and test rows are protected from being programmed and erased.
This bit cannot be modified while programming is taking place (EEPGM = 1).

BPROT[4:0] — EEPROM Block Protection
0 = Associated EEPROM block can be programmed and erased.
1 = Associated EEPROM block is protected from being programmed and erased.
Cannot be modified while programming is taking place (EEPGM = 1).

Table 15 768-byte EEPROM Block Protection

Bit Name	Block Protected	Block Size
BPROT4	$0D00 to $0DFF	256 Bytes
BPROT3	$0E00 to $0EFF	256 Bytes
BPROT2	$0F00 to $0F7F	128 Bytes
BPROT1	$0F80 to $0FBF	64 Bytes
BPROT0	$0FC0 to $0FFF	64 Bytes

EETST — EEPROM Test**$00F2**

	Bit 7	6	5	4	3	2	1	Bit 0
	EEODD	EEVEN	MARG	EECPD	EECPRD	0	EECPM	0
RESET:	0	0	0	0	0	0	0	0

Read anytime. Write in special modes only (SMODN = 0). These bits are used for test purposes only. In normal modes the bits are forced to zero.

EEODD — Odd Row Programming
0 = Odd row bulk programming/erasing is disabled.
1 = Bulk program/erase all odd rows.
Refers to a physical location in the array rather than an odd byte address.

EEVEN — Even Row Programming
0 = Even row bulk programming/erasing is disabled.
1 = Bulk program/erase all even rows.
Refers to a physical location in the array rather than an even byte address.

MARG — Program and Erase Voltage Margin Test Enable
0 = Normal operation.
1 = Program and Erase Margin test.
This bit is used to evaluate the program/erase voltage margin.

EECPD — Charge Pump Disable
0 = Charge pump is turned on during program/erase.
1 = Disable charge pump.

EECPRD — Charge Pump Ramp Disable

Known to enhance write/erase endurance of EEPROM cells.

0 = Charge pump is turned on progressively during program/erase.

1 = Disable charge pump controlled ramp up.

EECPM — Charge Pump Monitor Enable

0 = Normal operation.

1 = Output the charge pump voltage on the $\overline{IRQ}/V_{PP}$ pin.

EEPROG — EEPROM Control **$00F3**

	Bit 7	6	5	4	3	2	1	Bit 0
	BULKP	0	0	BYTE	ROW	ERASE	EELAT	EEPGM
RESET:	1	0	0	0	0	0	0	0

BULKP — Bulk Erase Protection

0 = EEPROM can be bulk erased.

1 = EEPROM is protected from being bulk or row erased.

Read anytime. Write anytime if EEPGM = 0 and PROTLCK = 0.

BYTE — Byte and Aligned Word Erase

0 = Bulk or row erase is enabled.

1 = One byte or one aligned word erase only.

Read anytime. Write anytime if EEPGM = 0.

ROW — Row or Bulk Erase (when BYTE = 0)

0 = Erase entire EEPROM array.

1 = Erase only one 32-byte row.

Read anytime. Write anytime if EEPGM = 0.

BYTE and ROW have no effect when ERASE = 0

Table 16 Erase Selection

BYTE	ROW	Block size
0	0	Bulk erase entire EEPROM array
0	1	Row erase 32 bytes
1	0	Byte or aligned word erase
1	1	Byte or aligned word erase

If BYTE = 1 and test mode is not enabled, only the location specified by the address written to the programming latches will be erased. The operation will be a byte or an aligned word erase depending on the size of written data.

ERASE — Erase Control

0 = EEPROM configuration for programming or reading.

1 = EEPROM configuration for erasure.

Read anytime. Write anytime if EEPGM = 0.

Configures the EEPROM for erasure or programming.

When test mode is not enabled and unless BULKP is set, erasure is by byte, aligned word, row or bulk.

EELAT — EEPROM Latch Control

0 = EEPROM set up for normal reads.

1 = EEPROM address and data bus latches set up for programming or erasing.

Read anytime. Write anytime if EEPGM = 0.

NOTE

When EELAT is set, the entire EEPROM is unavailable for reads, therefore no program residing in the EEPROM can be executed while attempting to program unused EEPROM space. Care should be taken that no references to the EEPROM are used while programming. Interrupts should be turned off if the vectors are in the EEPROM. Timing and any serial communications must be done with polling during the programming process.

BYTE, ROW, ERASE and EELAT bits can be written simultaneously or in any sequence.

EEPGM — Program and Erase Enable

0 = Disables program/erase voltage to EEPROM.

1 = Applies program/erase voltage to EEPROM.

The EEPGM bit can be set only after EELAT has been set. When an attempt is made to set EELAT and EEPGM simultaneously, EEPGM remains clear but EELAT is set.

The BULKP, BYTE, ROW, ERASE and EELAT bits cannot be changed when EEPGM is set. To complete a program or erase, two successive writes to clear EEPGM and EELAT bits are required before reading the programmed data. A write to an EEPROM location has no effect when EEPGM is set. Latched address and data cannot be modified during program or erase.

A program or erase operation should follow the sequence below:

1. Write BYTE, ROW and ERASE to the desired value, write EELAT = 1
2. Write a byte or an aligned word to an EEPROM address
3. Write EEPGM = 1
4. Wait for programming (t_{PROG}) or erase (t_{ERASE}) delay time
5. Write EEPGM = 0
6. Write EELAT = 0

It is possible to program/erase more bytes or words without intermediate EEPROM reads, by jumping from step 5 to step 2.

9 Resets and Interrupts

CPU12 exceptions include resets and interrupts. Each exception has an associated 16-bit vector, which points to the memory location where the routine that handles the exception is located. Vectors are stored in the upper 128 bytes of the standard 64K byte address map.

The six highest vector addresses are used for resets and non-maskable interrupt sources. The remainder of the vectors are used for maskable interrupts, and all must be initialized to point to the address of the appropriate service routine.

9.1 Exception Priority

A hardware priority hierarchy determines which reset or interrupt is serviced first when simultaneous requests are made. Six sources are not maskable. The remaining sources are maskable, and any one of them can be given priority over other maskable interrupts.

The priorities of the non-maskable sources are:

1. POR or $\overline{\text{RESET}}$ pin
2. Clock monitor reset
3. COP watchdog reset
4. Unimplemented instruction trap
5. Software interrupt instruction (SWI)
6. $\overline{\text{XIRQ}}$ signal (if X bit in CCR = 0)

9.2 Maskable interrupts

Maskable interrupt sources include on-chip peripheral systems and external interrupt service requests. Interrupts from these sources are recognized when the global interrupt mask bit (I) in the CCR is cleared. The default state of the I bit out of reset is one, but it can be written at any time.

Interrupt sources are prioritized by default but any one maskable interrupt source may be assigned the highest priority by means of the HPRIO register. The relative priorities of the other sources remain the same.

An interrupt that is assigned highest priority is still subject to global masking by the I bit in the CCR, or by any associated local bits. Interrupt vectors are not affected by priority assignment. HPRIO can only be written while the I bit is set (interrupts inhibited). **Table 17** lists interrupt sources and vectors in default order of priority.

Table 17 Interrupt Vector Map

Vector Address	Interrupt Source	CCR Mask	Local Enable Register (Bit)	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	none	none	–
$FFFC, $FFFD	COP Clock Monitor Fail Reset	none	COPCTL (CME, FCME)	–
$FFFA, $FFFB	COP Failure Reset	none	COP rate selected	–
$FFF8, $FFF9	Unimplemented Instruction Trap	none	none	–
$FFF6, $FFF7	SWI	none	none	–
$FFF4, $FFF5	XIRQ	X bit	none	–
$FFF2, $FFF3	IRQ	I bit	INTCR (IRQEN)	$F2
$FFF0, $FFF1	Real Time Interrupt	I bit	RTICTL (RTIE)	$F0
$FFEE, $FFEF	Timer Channel 0	I bit	TMSK1 (C0I)	$EE
$FFEC, $FFED	Timer Channel 1	I bit	TMSK1 (C1I)	$EC
$FFEA, $FFEB	Timer Channel 2	I bit	TMSK1 (C2I)	$EA
$FFE8, $FFE9	Timer Channel 3	I bit	TMSK1 (C3I)	$E8
$FFE6, $FFE7	Timer Channel 4	I bit	TMSK1 (C4I)	$E6
$FFE4, $FFE5	Timer Channel 5	I bit	TMSK1 (C5I)	$E4
$FFE2, $FFE3	Timer Channel 6	I bit	TMSK1 (C6I)	$E2
$FFE0, $FFE1	Timer Channel 7	I bit	TMSK1 (C7I)	$E0
$FFDE, $FFDF	Timer Overflow	I bit	TMSK2 (TOI)	$DE
$FFDC, $FFDD	Pulse Accumulator Overflow	I bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse Accumulator Input Edge	I bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI Serial Transfer Complete	I bit	SP0CR1 (SPIE)	$D8
$FFD6, $FFD7	SCI 0	I bit	SC0CR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	Reserved	I bit		$D4
$FFD2, $FFD3	ATD	I bit	ATDCTL2 (ASCIE)	$D2
$FFD0, $FFD1	MSCAN Wake-Up	I bit	CRIER (WUPIE)	$D0
$FFCA-$FFCF	Reserved	I bit		$CA-$CF
$FFC8, $FFC9	MSCAN Errors	I bit	CRIER (RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE)	$C8
$FFC6, $FFC7	MSCAN Receive	I bit	CRIER (RXFIE)	$C6
$FFC4, $FFC5	MSCAN Transmit	I bit	CTCR (TXEIE[2:0])	$C4
$FF80–$FFC3	Reserved	I bit		$80–$C3

9.3 Interrupt Control and Priority Registers

INTCR — Interrupt Control Register **$001E**

	Bit 7	6	5	4	3	2	1	Bit 0
	IRQE	IRQEN	DLY	0	0	0	0	0
RESET:	0	1	1	0	0	0	0	0

IRQE — $\overline{\text{IRQ}}$ Select Edge Sensitive Only

0 = $\overline{\text{IRQ}}$ configured for low-level recognition.

1 = $\overline{\text{IRQ}}$ configured to respond only to falling edges (on pin PE1/$\overline{\text{IRQ}}$).

IRQE can be read anytime and written once in normal modes. In special modes, IRQE can be read anytime and written anytime, except the first write is ignored.

IRQEN — External $\overline{\text{IRQ}}$ Enable

The $\overline{\text{IRQ}}$ pin has an internal pull-up.

0 = External $\overline{\text{IRQ}}$ pin disconnected from interrupt logic

1 = External $\overline{\text{IRQ}}$ pin connected to interrupt logic

IRQEN can be read and written anytime in all modes.

DLY — Enable Oscillator Start-up Delay on Exit from STOP

The delay time of about 4096 cycles is based on the E clock rate.

0 = No stabilization delay imposed on exit from STOP mode. A stable external oscillator must be supplied.

1 = Stabilization delay is imposed before processing resumes after STOP.

DLY can be read anytime and written once in normal modes. In special modes, DLY can be read and written anytime.

HPRIO — Highest Priority I Interrupt **$001F**

	Bit 7	6	5	4	3	2	1	Bit 0
	1	1	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	0
RESET:	1	1	1	1	0	0	1	0

Write only if I mask in CCR = 1 (interrupts inhibited). Read anytime.

To give a maskable interrupt source highest priority, write the low byte of the vector address to the HPRIO register. For example, writing $F0 to HPRIO would assign highest maskable interrupt priority to the real-time interrupt timer ($FFF0). If an unimplemented vector address or a non-I-masked vector address (value higher than $F2) is written, then $\overline{\text{IRQ}}$ will be the default highest priority interrupt.

9.4 Resets

There are four possible sources of reset. Power-on reset (POR), and external reset on the $\overline{\text{RESET}}$ pin share the normal reset vector. The computer operating properly (COP) reset and the clock monitor reset each has a vector. Entry into reset is asynchronous and does not require a clock but the MCU cannot sequence out of reset without a system clock.

9.4.1 Power-On Reset

A positive transition on V_{DD} causes a power-on reset (POR). An external voltage level detector, or other external reset circuits, are the usual source of reset in a system. The POR circuit only initializes internal circuitry during cold starts and cannot be used to force a reset as system voltage drops.

9.4.2 External Reset

The CPU distinguishes between internal and external reset conditions by sensing whether the reset pin rises to a logic one in less than eight E-clock cycles after an internal device releases reset. When a reset condition is sensed, the $\overline{\text{RESET}}$ pin is driven low by an internal device for about 16 E-clock cycles, then released. Eight E-clock cycles later it is sampled. If the pin is still held low, the CPU assumes that an external reset has occurred. If the pin is high, it indicates that the reset was initiated internally by either the COP system or the clock monitor.

To prevent a COP or clock monitor reset from being detected during an external reset, hold the reset pin low for at least 32 cycles. An external RC power-up delay circuit on the reset pin is not recommended — circuit charge time can cause the MCU to misinterpret the type of reset that has occurred.

9.4.3 COP Reset

The MCU includes a computer operating properly (COP) system to help protect against software failures. When COP is enabled, software must write $55 and $AA (in this order) to the COPRST register in order to keep a watchdog timer from timing out. Other instructions may be executed between these writes. A write of any value other than $55 or $AA or software failing to execute the sequence properly causes a COP reset to occur.

9.4.4 Clock Monitor Reset

If clock frequency falls below a predetermined limit when the clock monitor is enabled, a reset occurs.

9.5 Effects of Reset

When a reset occurs, MCU registers and control bits are changed to known start-up states, as follows.

9.5.1 Operating Mode and Memory Map

Operating mode and default memory mapping are determined by the states of the BKGD, MODA, and MODB pins during reset. The SMODN, MODA, and MODB bits in the MODE register reflect the status of the mode-select inputs at the rising edge of reset. Operating mode and default maps can subsequently be changed according to strictly defined rules.

9.5.2 Clock and Watchdog Control Logic

The COP watchdog system is enabled, with the CR [2:0] bits set for the shortest duration time-out. The clock monitor is disabled. The RTIF flag is cleared and automatic hardware interrupts are masked. The rate control bits are cleared, and must be initialized before the RTI system is used. The DLY control bit is set to specify an oscillator start-up delay upon recovery from STOP mode.

9.5.3 Interrupts

PSEL is initialized in the HPRIO register with the value $F2, causing the external $\overline{\text{IRQ}}$ pin to have the highest I-bit interrupt priority. The $\overline{\text{IRQ}}$ pin is configured for level-sensitive operation (for wired-OR systems). However, the interrupt mask bits in the CPU12 CCR are set to mask X and I related interrupt requests.

9.5.4 Parallel I/O

If the MCU comes out of reset in an expanded mode, port A and port B are used for the multiplexed address/data bus and port E pins are normally used to control the external bus (operation of port E pins can be affected by the PEAR register). If the MCU comes out of reset in a single-chip mode, all ports are configured as general-purpose high-impedance inputs. Port S, port T, port DLC, port P, and port AD are all configured as general-purpose inputs.

9.5.5 Central Processing Unit

After reset, the CPU fetches a vector from the appropriate address, then begins executing instructions. The stack pointer and other CPU registers are indeterminate immediately after reset. The CCR X and I interrupt mask bits are set to mask any interrupt requests. The S bit is also set to inhibit the STOP instruction.

9.5.6 Memory

After reset, the internal register block is located at $0000–$01FF, the register-following space is at $0200–$03FF, and RAM is at $0800–$0BFF. EEPROM is located at $0D00–$0FFF. Flash EEPROM is located at $8000–$FFFF in single-chip modes and at $0000–$7FFF (but disabled) in expanded modes.

9.5.7 Other Resources

The timer, serial communications interface (SCI), serial peripheral interface (SPI), CAN controller (MSCAN12), pulse-width modulator (PWM), and analog-to-digital converter (ATD) are off after reset.

9.6 Register Stacking

Once enabled, an interrupt request can be recognized at any time after the I bit in the CCR is cleared. When an interrupt service request is recognized, the CPU responds at the completion of the instruction being executed. Interrupt latency varies according to the number of cycles required to complete the instruction. Some of the longer instructions can be interrupted and will resume normally after servicing the interrupt.

When the CPU begins to service an interrupt, the instruction queue is cleared, the return address is calculated, and then it and the contents of the CPU registers are stacked as shown in **Table 18**.

Table 18 Stacking Order on Entry to Interrupts

Memory Location	CPU Registers
SP – 2	RTN_H : RTN_L
SP – 4	Y_H : Y_L
SP – 6	X_H : X_L
SP – 8	B : A
SP – 9	CCR

After the CCR is stacked, the I bit (and the X bit, if an $\overline{\text{XIRQ}}$ interrupt service request is pending) is set to prevent other interrupts from disrupting the interrupt service routine. The interrupt vector for the highest priority source that was pending at the beginning of the interrupt sequence is fetched, and execution continues at the referenced location. At the end of the interrupt service routine, an RTI instruction restores the content of all registers from information on the stack, and normal program execution resumes. If another interrupt is pending at the end of an interrupt service routine, the register unstacking and restacking is bypassed and the vector of the pending interrupt is fetched.

10 Clock Functions

Clock generation circuitry generates the internal and external E-clock signals as well as internal clock signals used by the CPU and on-chip peripherals. A clock monitor circuit, a computer operating properly (COP) watchdog circuit, and a periodic interrupt circuit are also incorporated into the MC68HC912BC32.

10.1 Clock Sources

A compatible external clock signal can be applied to the EXTAL pin or the MCU can generate a clock signal using an on-chip oscillator circuit and an external crystal or ceramic resonator. The MCU uses three types of internal clock signals derived from the primary clock signal: T clocks, E clock, and P clock. The T clocks are used by the CPU. The E and P clocks are used by the bus interfaces, BDM, SPI, and ATD. The P clock also drives on-chip modules such as the timer chain, SCI, RTI, COP, and restart-from-stop delay time. **Figure 10** shows clock timing relationships.

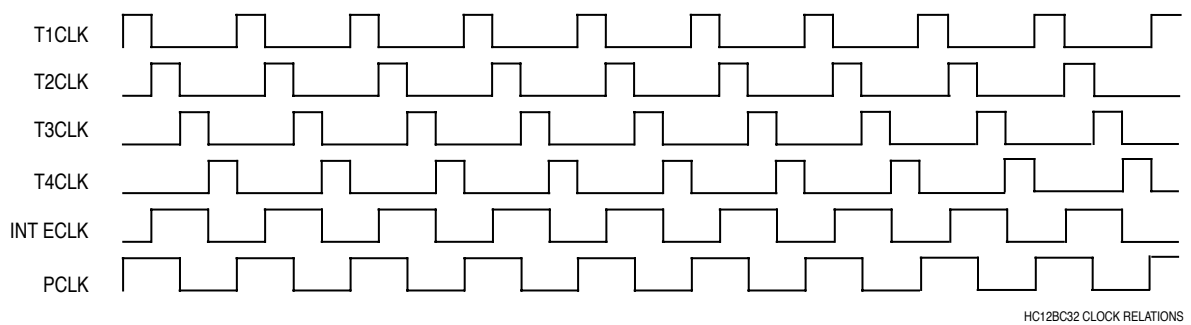


Figure 10 Internal Clock Relationships

10.2 Computer Operating Properly (COP)

The COP or watchdog timer is an added check that a program is running and sequencing properly. When the COP is being used, software is responsible for keeping a free running watchdog timer from timing out. If the watchdog timer times out it is an indication that the software is no longer being executed in the intended sequence; thus a system reset is initiated. Three control bits allow selection of seven COP time-out periods or COP disable. When COP is enabled, sometime during the selected period the program must write $55 and $AA (in this order) to the COPRST register. If the program fails to do this the part will reset. If any value other than $55 or $AA is written to COPRST, the part is reset.

10.3 Real-Time Interrupt

There is a real time (periodic) interrupt available to the user. This interrupt will occur at one of seven selected rates. An interrupt flag and an interrupt enable bit are associated with this function. There are three bits for the rate select.

10.4 Clock Monitor

The clock monitor circuit is based on an internal resistor-capacitor (RC) time delay. If no MCU clock edges are detected within this RC time delay, the clock monitor can optionally generate a system reset. The clock monitor function is enabled/disabled by the CME control bit in the COPCTL register. This time-out is based on an RC delay so that the clock monitor can operate without any MCU clocks.

Clock monitor time-outs are shown in **Table 19**.

Table 19 Clock Monitor Time-Outs

Supply	Range
5V +/– 10%	2-20 µS
3V +/– 10%	5-100 µS

10.5 Clock Function Registers

All register addresses shown reflect the reset state. Registers may be mapped to any 2-Kbyte space.

RTICTL — Real-Time Interrupt Control Register **$0014**

	Bit 7	6	5	4	3	2	1	Bit 0
	RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0
RESET:	0	0	0	0	0	0	0	0

RTIE — Real Time Interrupt Enable

Read and write anytime.

0 = Interrupt requests from RTI are disabled.

1 = Interrupt will be requested whenever RTIF is set.

RSWAI — RTI and COP Stop While in Wait

Write once in normal modes, anytime in special modes. Read anytime.

0 = Allows the RTI and COP to continue running in wait.

1 = Disables both the RTI and COP whenever the part goes into Wait.

RSBCK — RTI and COP Stop While in Background Debug Mode

Write once in normal modes, anytime in special modes. Read anytime.

0 = Allows the RTI and COP to continue running while in background mode.

1 = Disables both the RTI and COP whenever the part is in background mode. This is useful for emulation.

RTBYP — Real Time Interrupt Divider Chain Bypass

Write not allowed in normal modes, anytime in special modes. Read anytime.

0 = Divider chain functions normally.

1 = Divider chain is bypassed, allows faster testing (the divider chain is normally P divided by 2^{13}, when bypassed becomes P divided by 4).

RTR2, RTR1, RTR0 — Real-Time Interrupt Rate Select

Read and write anytime.

Rate select for real-time interrupt. The clock used for this module is the E clock.

Table 20 Real Time Interrupt Rates

RTR2	RTR1	RTR0	Divide E By:	Time-Out Period E = 4.0 MHz	Time-Out Period E = 8.0 MHz
0	0	0	OFF	OFF	OFF
0	0	1	2^{13}	2.048 ms	1.024 ms
0	1	0	2^{14}	4.096 ms	2.048 ms
0	1	1	2^{15}	8.196 ms	4.096 ms
1	0	0	2^{16}	16.384 ms	8.196 ms
1	0	1	2^{17}	32.768 ms	16.384 ms
1	1	0	2^{18}	65.536 ms	32.768 ms
1	1	1	2^{19}	131.72 ms	65.536 ms

RTIFLG — Real Time Interrupt Flag Register **$0015**

	Bit 7	6	5	4	3	2	1	Bit 0
	RTIF	0	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

RTIF — Real Time Interrupt Flag

This bit is cleared automatically by a write to this register with this bit set.

0 = Time-out has not yet occurred.

1 = Set when the time-out period is met.

COPCTL — COP Control Register **$0016**

	Bit 7	6	5	4	3	2	1	Bit 0	
	CME	FCME	FCM	FCOP	DISR	CR2	CR1	CR0	
RESET:	0	0	0	0	0	0	0	1	Normal
RESET:	0	0	0	0	1	0	0	1	Special

CME — Clock Monitor Enable

Read and write anytime.

If FCME is set, this bit has no meaning nor effect.

0 = Clock monitor is disabled. Slow clocks and stop instruction may be used.

1 = Slow or stopped clocks (including the stop instruction) will cause a clock reset sequence.

FCME — Force Clock Monitor Enable

Write once in normal modes, anytime in special modes. Read anytime.

In normal modes, when this bit is set, the clock monitor function cannot be disabled until a reset occurs.

0 = Clock monitor follows the state of the CME bit.

1 = Slow or stopped clocks will cause a clock reset sequence.

In order to use both STOP and Clock Monitor, the CME bit should be cleared prior to executing a STOP instruction and set after recovery from STOP. If you plan on using STOP always keep FCME = 0.

FCM — Force Clock Monitor Reset

Writes are not allowed in normal modes, anytime in special modes. Read anytime.

If DISR is set, this bit has no effect.

0 = Normal operation.

1 = Force a clock monitor reset (if clock monitor is enabled).

FCOP — Force COP Watchdog Reset

Writes are not allowed in normal modes; can be written anytime in special modes. Read anytime.
If DISR is set, this bit has no effect.

0 = Normal operation.
1 = Force a COP reset (if COP is enabled).

DISR — Disable Resets from COP Watchdog and Clock Monitor

Writes are not allowed in normal modes, anytime in special modes. Read anytime.

0 = Normal operation.
1 = Regardless of other control bit states, COP and clock monitor will not generate a system reset.

CR2, CR1, CR0 — COP Watchdog Timer Rate Select Bits

The COP system is driven by a constant frequency of $E/2^{13}$. (RTBYP in the RTICTL register allows all but two stages of this divider to be bypassed for testing in Special modes only.) These bits specify an additional division factor to arrive at the COP time-out rate (the clock used for this module is the E clock). Write once in normal modes, anytime in special modes. Read anytime.

Table 21 COP Watchdog Rates (RTBYP = 0)

CR2	CR1	CR0	Divide E By:	At E = 4.0 MHz Time-Out –0 to +2.048 ms	At E = 8.0 MHz Time-Out –0 to +1.024 ms
0	0	0	OFF	OFF	OFF
0	0	1	2^{13}	2.048 ms	1.024 ms
0	1	0	2^{15}	8.1920 ms	4.096 ms
0	1	1	2^{17}	32.768 ms	16.384 ms
1	0	0	2^{19}	131.072 ms	65.536 ms
1	0	1	2^{21}	524.288 ms	262.144 ms
1	1	0	2^{22}	1.048 s	524.288 ms
1	1	1	2^{23}	2.097 s	1.048576 s

COPRST — Arm/Reset COP Timer Register **$0017**

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

Always reads $00.

Writing $55 to this address is the first step of the COP watchdog sequence.

Writing $AA to this address is the second step of the COP watchdog sequence. Other instructions may be executed between these writes but both must be completed in the correct order prior to time-out to avoid a watchdog reset. Writing anything other than $55 or $AA causes a COP reset to occur.

10.6 Clock Divider Chains

Figure 11, **Figure 12**, **Figure 13**, and **Figure 14** summarize the clock divider chains for the various peripherals on the MC68HC912BC32.

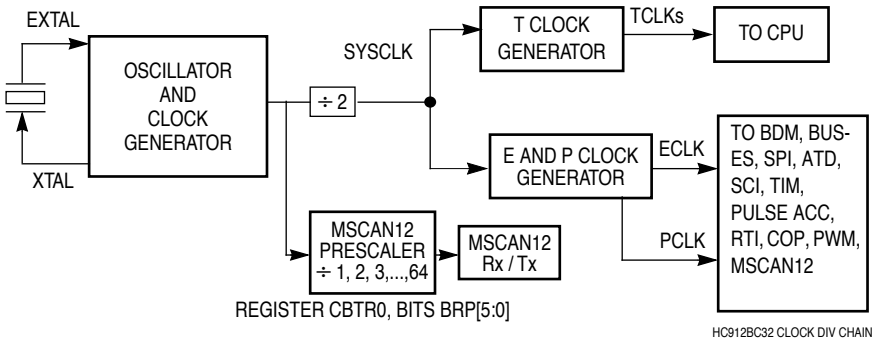


Figure 11 Clock Divider Chain

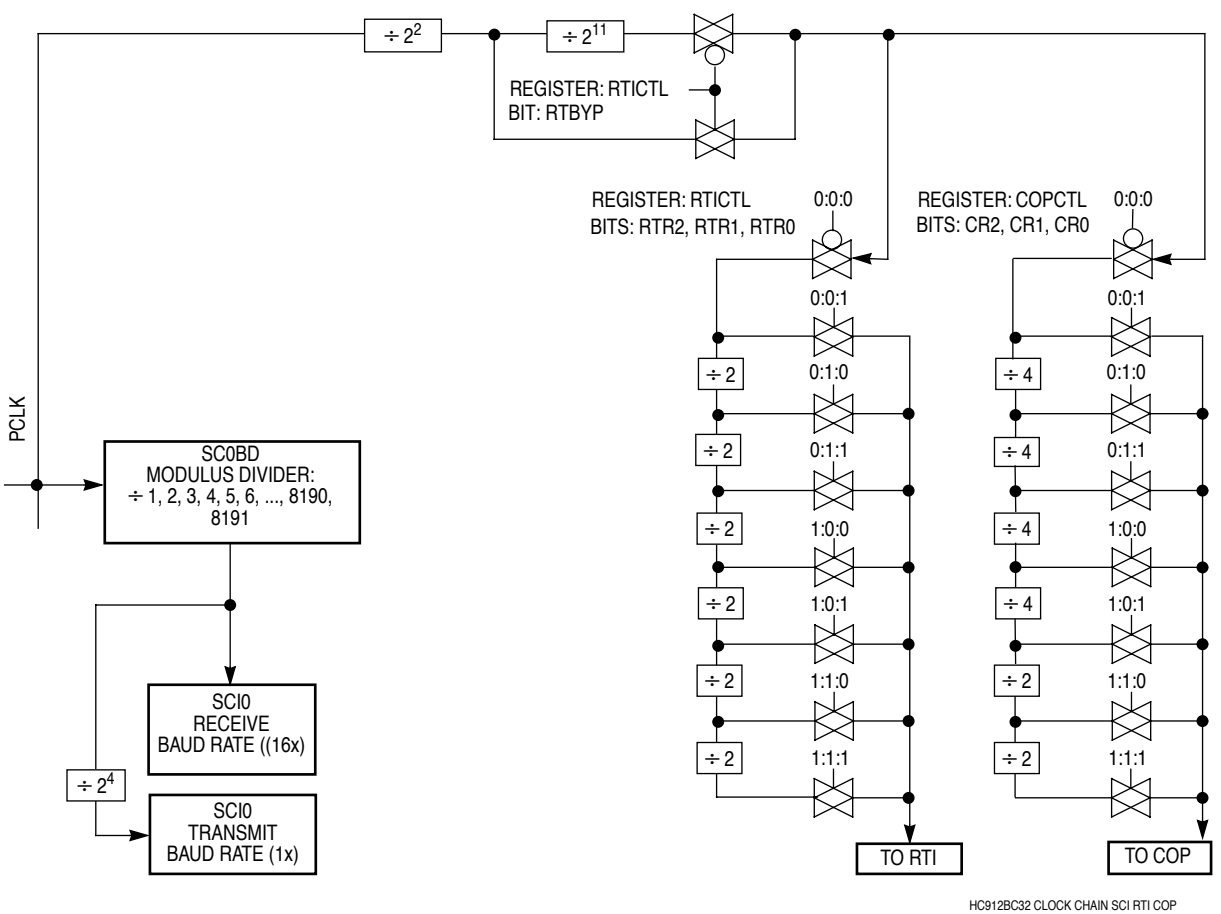


Figure 12 Clock Chain for SCI, RTI, COP

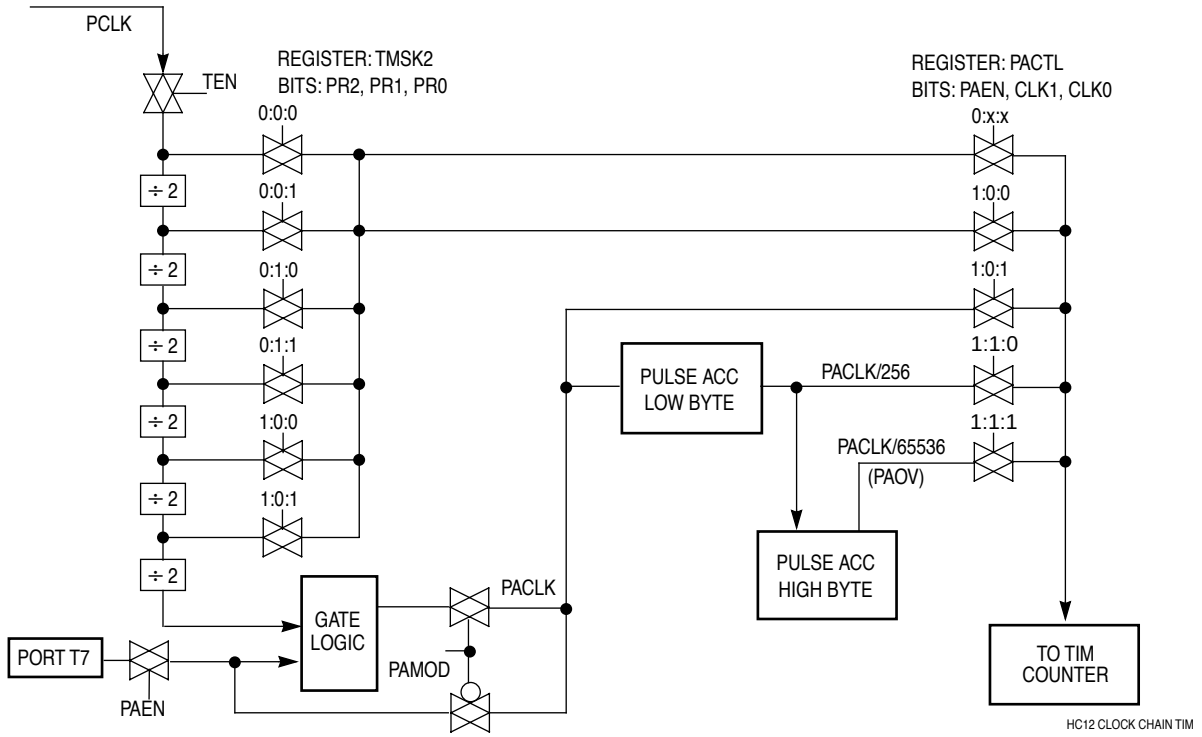


Figure 13 Clock Chain for TIM

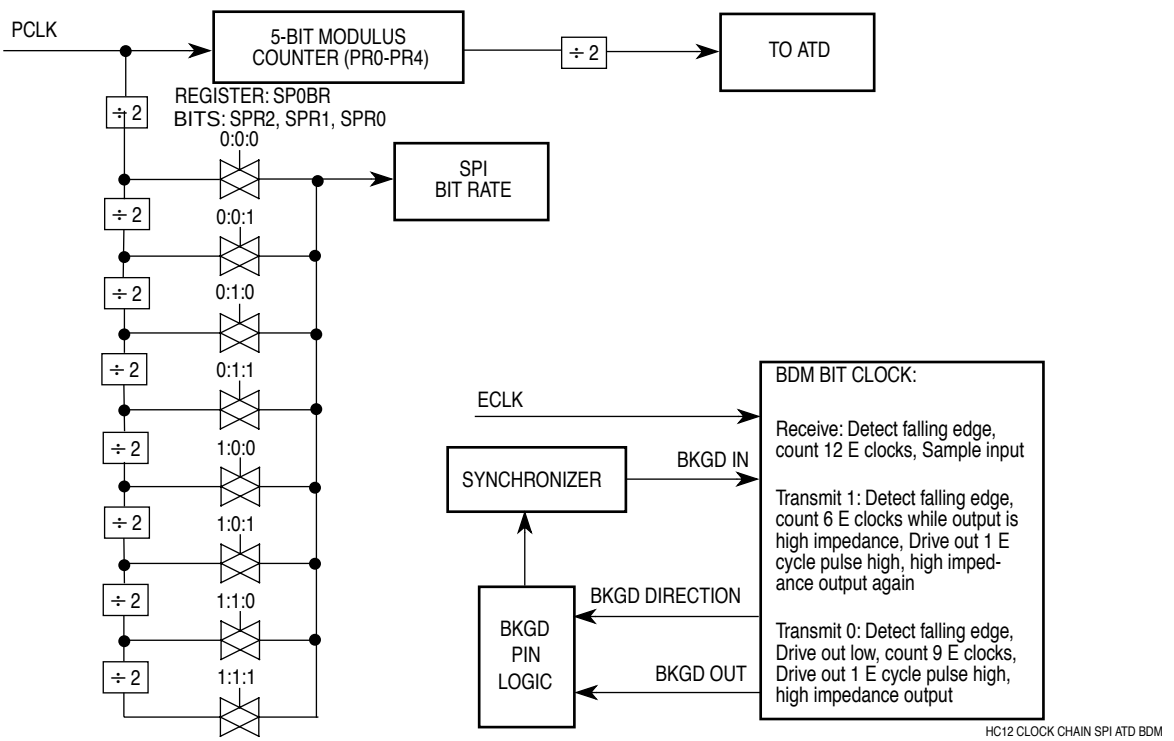


Figure 14 Clock Chain for SPI, ATD and BDM

11 Pulse Width Modulator

The pulse-width modulator (PWM) subsystem provides four independent 8-bit PWM waveforms or two 16-bit PWM waveforms or a combination of one 16-bit and two 8-bit PWM waveforms. Each waveform channel has a programmable period and a programmable duty-cycle as well as a dedicated counter. A flexible clock select scheme allows four different clock sources to be used with the counters. Each of the modulators can create independent, continuous waveforms with software-selectable duty rates from 0 percent to 100 percent. The PWM outputs can be programmed as left-aligned outputs or center-aligned outputs.

The four PWM channel outputs share general-purpose port P pins. Enabling PWM pins takes precedence over the general-purpose port. When PWM are not in use, the port pins may be used for discrete input/output.

The period and duty registers are double buffered so that if they change while the channel is enabled, the change will not take effect until the counter reaches zero or the channel is disabled and then re-enabled. If the channel is not enabled then writes to the period and/or duty register will go only to the shadow latches. When the channel is enabled (PWENx) is written from 0 to 1) the counter direction is set to UP and the Shadow to Register transfer for PWPERx and PWDTYx is done.

For Center Aligned mode two additional initialization operations occur: the PWM counter is reset to $00 and a decision for the PWM output value takes place. In this way the output of the PWM will always be; the old waveform, or the new waveform and not some variation of the two.

The pulse modulated signal becomes available at port P output when the PWM channel clock source begins its next cycle.

A change in duty or period can be forced into immediate effect by writing the new value to the duty and/or period registers and then writing to the counter. This causes the counter to reset and the new duty and/or period values to be latched (this may cause a truncated PWM period). In addition, since the counter is readable it is possible to know where the count is with respect to the duty value and software can be used to make adjustments by turning the enable bit off and on (only in Left Aligned output mode).

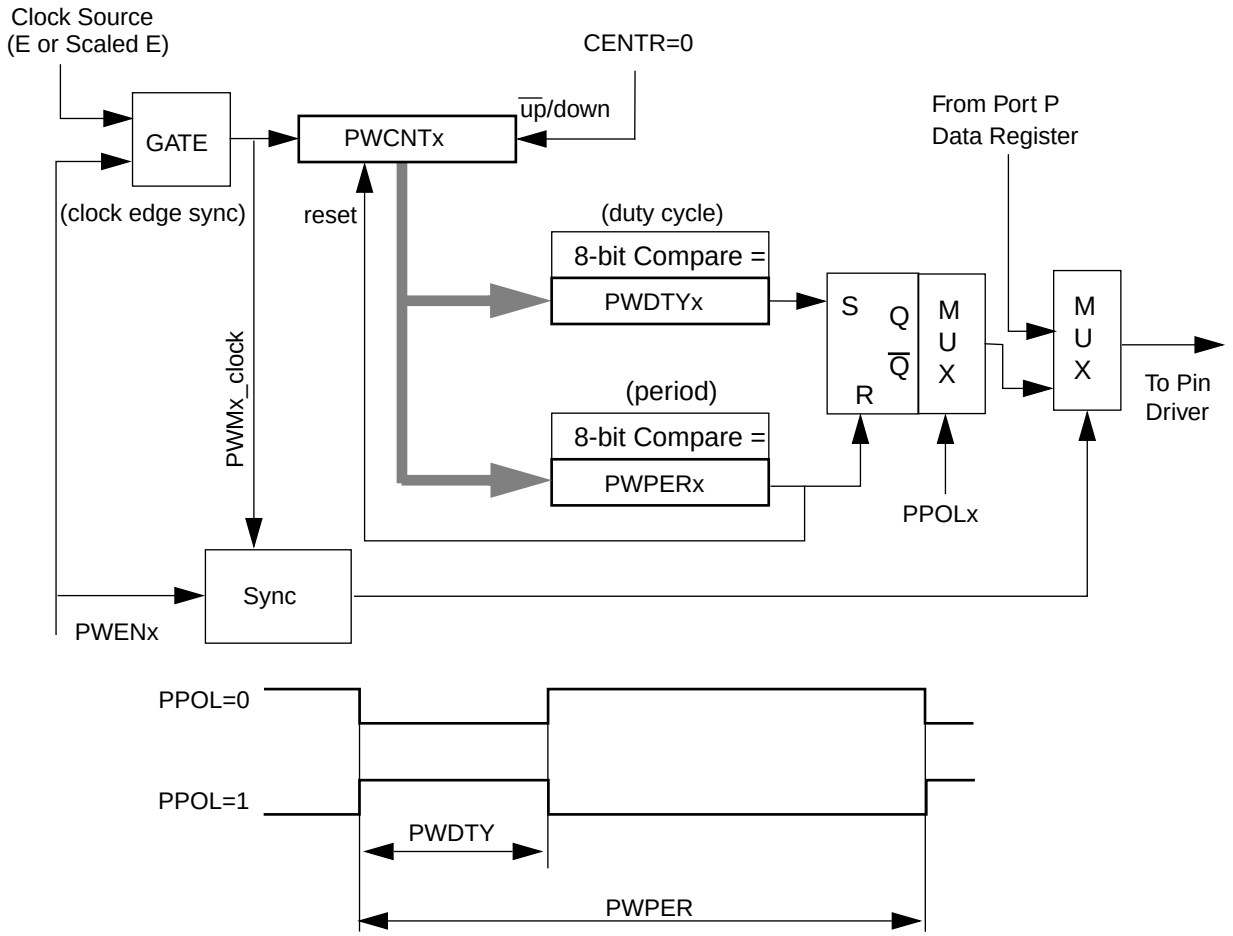


Figure 15 Block Diagram of PWM Left-Aligned Output Channel

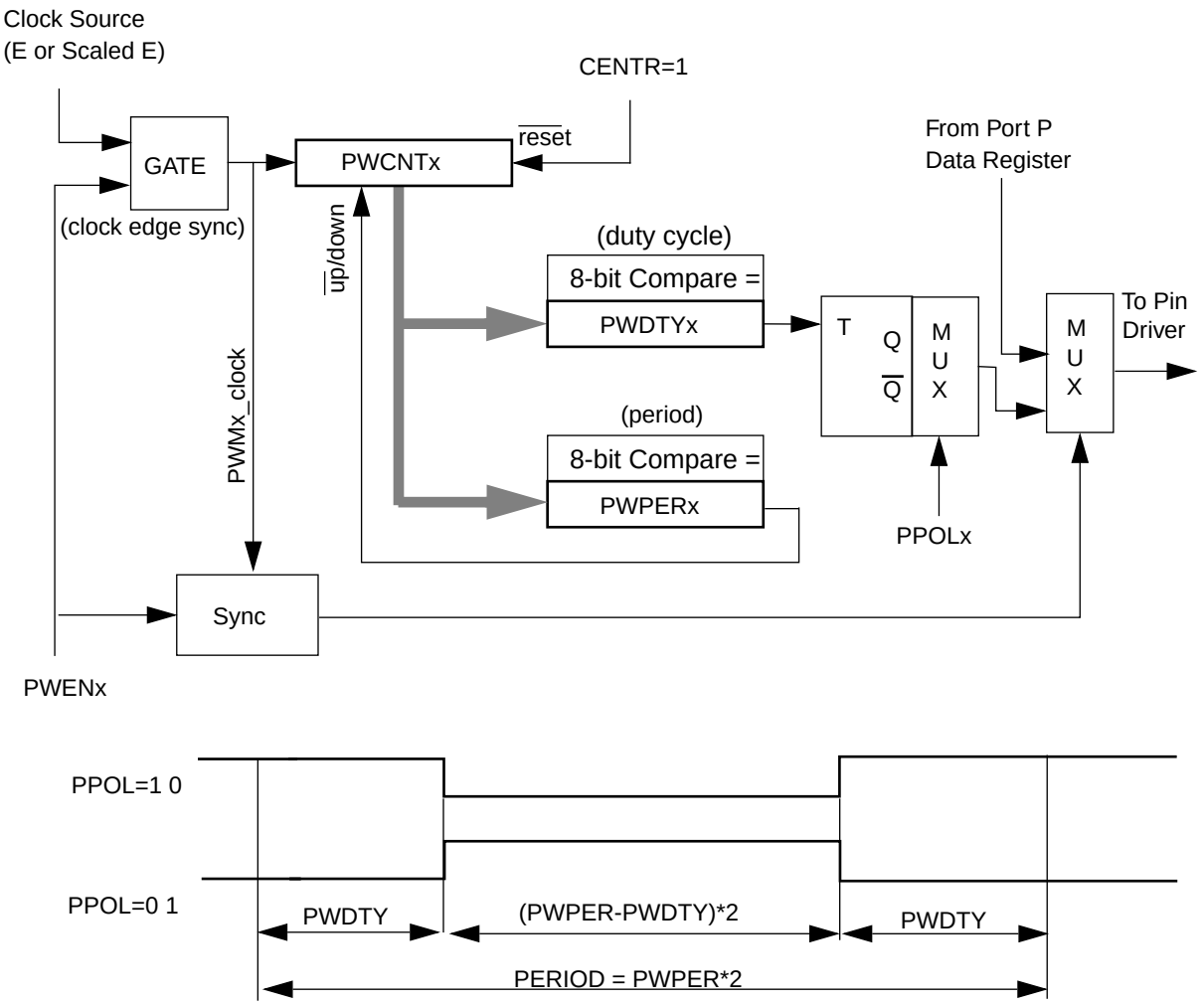


Figure 16 Block Diagram of PWM Center-Aligned Output Channel

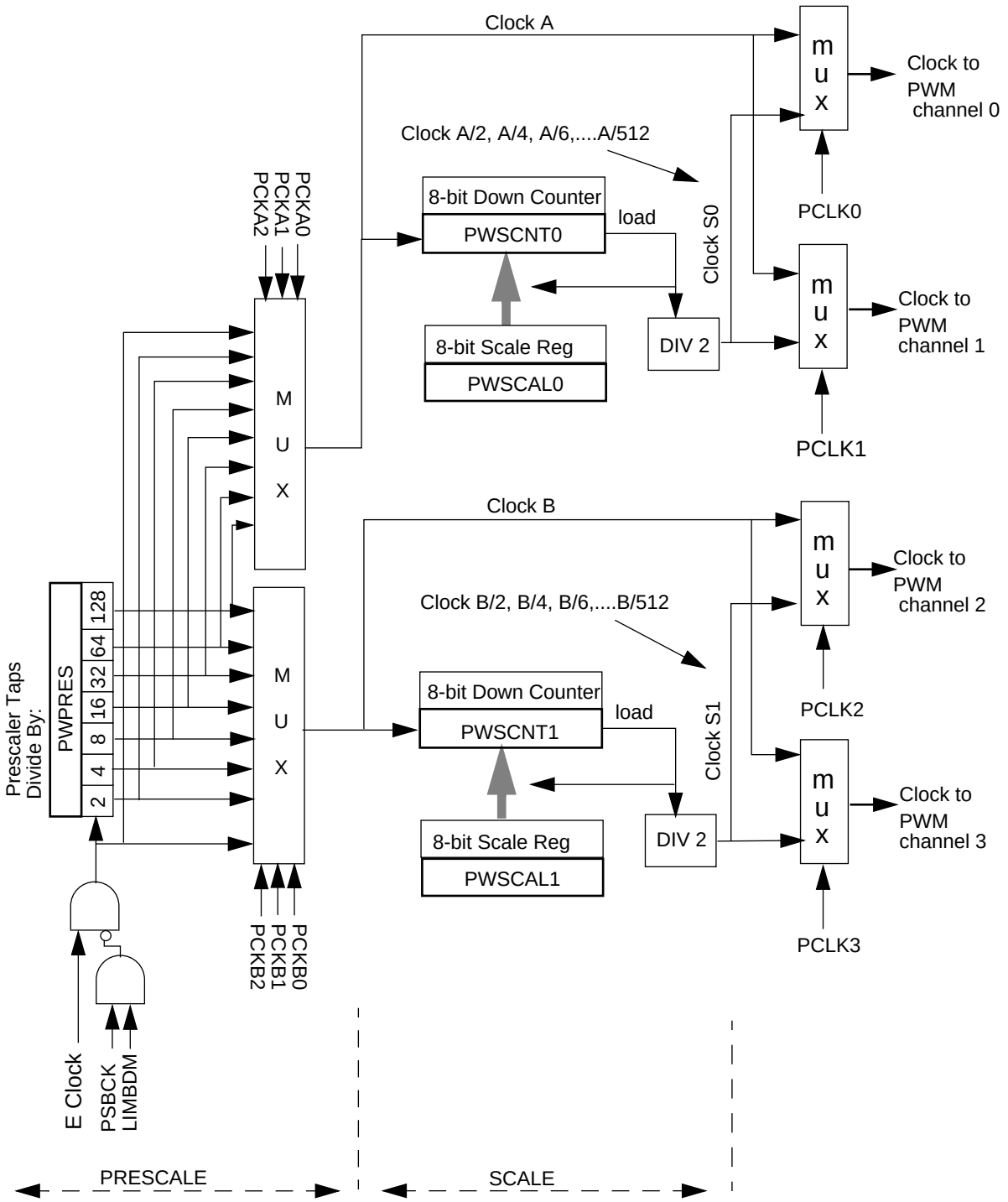


Figure 17 PWM Clock Sources

11.1 PWM Register Description

PWCLK — PWM Clock Select Register with Concatenate Bits **$0040**

	Bit 7	6	5	4	3	2	1	Bit 0
	CON23	CON01	PCKA2	PCKA1	PCKA0	PCKB2	PCKB1	PCKB0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

CON23 — Concatenate PWM Channels 2 and 3

When concatenated, channel 2 becomes the high-order byte and channel 3 becomes the low-order byte. Channel 2 output pin is used as the output for this 16-bit PWM (bit 2 of port P). Channel 3 clock-select control bits determines the clock source.

0 = Channels 2 and 3 are separate 8-bit PWMs.
1 = Channels 2 and 3 are concatenated to create one 16-bit PWM channel.

CON01 — Concatenate PWM Channels 0 and 1

When concatenated, channel 0 becomes the high-order byte and channel 1 becomes the low-order byte. Channel 0 output pin is used as the output for this 16-bit PWM (bit 0 of port P). Channel 1 clock-select control bits determines the clock source.

0 = Channels 0 and 1 are separate 8-bit PWMs.
1 = Channels 0 and 1 are concatenated to create one 16-bit PWM channel.

NOTE

These bits should be changed only when both corresponding channels are disabled. For Left Aligned output mode operation when changing these bits the user should write to the associated PWM counters as the LAST operation before enabling (setting PWENx = q) the channel(s).

PCKA2 - PCKA0 — Prescaler for Clock A

Clock A is one of two clock sources which may be used for channels 0 and 1. These three bits determine the rate of clock A, as shown in **Table 22**.

PCKB2 - PCKB0 — Prescaler for Clock B

Clock B is one of two clock sources which may be used for channels 2 and 3. These three bits determine the rate of clock B, as shown in **Table 22**.

Table 22 Clock A and Clock B Prescaler

PCKA2 (PCKB2)	PCKA1 (PCKB1)	PCKA0 (PCKB0)	Value of Clock A (B)
0	0	0	E
0	0	1	E/ 2
0	1	0	E/4
0	1	1	E/ 8
1	0	0	E/16
1	0	1	E/ 32
1	1	0	E/ 64
1	1	1	E/ 128

PWPOL— PWM Clock Select and Polarity **$0041**

	Bit 7	6	5	4	3	2	1	Bit 0
	PCLK3	PCLK2	PCLK1	PCLK0	PPOL3	PPOL2	PPOL1	PPOL0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

PCLK3 — PWM Channel 3 Clock Select
0 = Clock B is the clock source for channel 3.
1 = Clock S1 is the clock source for channel 3.

PCLK2 — PWM Channel 2 Clock Select
0 = Clock B is the clock source for channel 2.
1 = Clock S1 is the clock source for channel 2.

PCLK1 — PWM Channel 1 Clock Select
0 = Clock A is the clock source for channel 1.
1 = Clock S0 is the clock source for channel 1.

PCLK0 — PWM Channel 0 Clock Select
0 = Clock A is the clock source for channel 0.
1 = Clock S0 is the clock source for channel 0.

PPOL3 — PWM Channel 3 Polarity
0 = Channel 3 output is low at the beginning of the period; high when the duty count is reached.
1 = Channel 3 output is high at the beginning of the period; low when the duty count is reached.

PPOL2 — PWM Channel 2 Polarity
0 = Channel 2 output is low at the beginning of the period; high when the duty count is reached.
1 = Channel 2 output is high at the beginning of the period; low when the duty count is reached.

PPOL1 — PWM Channel 1 Polarity
0 = Channel 1 output is low at the beginning of the period; high when the duty count is reached.
1 = Channel 1 output is high at the beginning of the period; low when the duty count is reached.

PPOL0 — PWM Channel 0 Polarity
0 = Channel 0 output is low at the beginning of the period; high when the duty count is reached.
1 = Channel 0 output is high at the beginning of the period; low when the duty count is reached.

NOTE

Register bits PCLK0 to PCLK3 may be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse may occur during transition.

NOTE

Depending on the polarity bit, the duty registers may contain the count of either the high time or the low time. If the polarity bit is one, the output starts high and then goes low when the duty count is reached, so the duty registers contain a count of the high time. If the polarity bit is zero, the output starts low and then goes high when the duty count is reached, so the duty registers contain a count of the low time.

PWEN — PWM Enable **$0042**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	PWEN3	PWEN2	PWEN1	PWEN0
RESET:	0	0	0	0	0	0	0	0

Read and Write any time.

Setting any of the PWENx bits causes the associated port P line to become an output regardless of the state of the associated data direction register (DDRP) bit. This does not change the state of the data direction bit. When PWENx returns to zero, the data direction bit controls I/O direction. On the front end of the PWM channel, the scaler clock is enabled to the PWM circuit by the PWENx enable bit being high. When all four PWM channels are disabled, the prescaler counter shuts off to save power. There is an edge-synchronizing gate circuit to guarantee that the clock will only be enabled or disabled at an edge.

Read and write anytime.

PWEN3 — PWM Channel 3 Enable

The pulse modulated signal will be available at port P, bit 3 when its clock source begins its next cycle.

0 = Channel 3 is disabled.

1 = Channel 3 is enabled.

PWEN2 — PWM Channel 2 Enable

The pulse modulated signal will be available at port P, bit 2 when its clock source begins its next cycle.

0 = Channel 2 is disabled.

1 = Channel 2 is enabled.

PWEN1 — PWM Channel 1 Enable

The pulse modulated signal will be available at port P, bit 1 when its clock source begins its next cycle.

0 = Channel 1 is disabled.

1 = Channel 1 is enabled.

PWEN0 — PWM Channel 0 Enable

The pulse modulated signal will be available at port P, bit 0 when its clock source begins its next cycle.

0 = Channel 0 is disabled.

1 = Channel 0 is enabled.

PWPRES— PWM Prescale Counter **$0043**

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

PWPRES is a free running 7-bit counter. Read anytime. Write only in Special mode (SMOD = 1).

PWSCAL0— PWM Scale Register 0 **$0044**

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

Read and write anytime. A write will cause the scaler counter PWSCNT0 to load the PWSCAL0 value unless in Special mode with DISCAL =1 in the PWTST register.
PWM channels 0 and 1 can select clock S0 (scaled) as its input clock by setting the control bit PCLK0 and PCLK1 respectively. Clock S0 is generated by dividing clock A by the value in the PWSCAL0 register and dividing again by two. When PWSCAL0 = $FF, clock A is divided by 256 then divided by two to generate clock S0. In Special Mode when DISCAL = 1, writes to PWSCAL0 does not load Scale-Counter 0 (PWSCNT0).

PWSCNT0— PWM Scale Counter 0 Value **$0045**

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

PWSCNT0 is a down-counter that, upon reaching $00, loads the value of PWSCAL0. Read any time.

PWSCAL1— PWM Scale Register 1 **$0046**

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

Read and write anytime. A write will cause the scaler counter PWSCNT1 to load the PWSCAL1 value unless in Special mode with DISCAL =1 in the PWTST register.
PWM channels 2 and 3 can select clock S1 (scaled) as its input clock by setting the control bit PCLK2 and PCLK3 respectively. Clock S1 is generated by dividing clock B by the value in the PWSCAL1 register and dividing again by two. When PWSCAL1 = $FF, clock B is divided by 256 then divided by two to generate clock S1.

PWSCNT1— PWM Scale Counter 1 Value **$0047**

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

PWSCNT1 is a down-counter that, upon reaching $00, loads the value of PWSCAL1. Read any time.

PWCNTx — PWM Channel Counters

	Bit 7	6	5	4	3	2	1	Bit 0	
PWCNT0	Bit 7	6	5	4	3	2	1	Bit 0	**$0048**
PWCNT1	Bit 7	6	5	4	3	2	1	Bit 0	**$0049**
PWCNT2	Bit 7	6	5	4	3	2	1	Bit 0	**$004A**
PWCNT3	Bit 7	6	5	4	3	2	1	Bit 0	**$004B**
RESET	0	0	0	0	0	0	0	0	

Read and write anytime. A write will cause the PWM counter to reset to $00.
In Special mode, if DISCR = 1, a write does not reset the PWM counter.
Each channel has its own counter. Each counter may be read any time without affecting the count or the operation of the corresponding PWM channel. Writes to a counter cause the counter to be reset to $00 and force an immediate load of both duty and period registers with new values. In concatenated mode, writes to the 16-bit counter by using 16 bit access, or writes to the high order byte of the counter, will cause reset of the 16-bit counters.Writes to the lower byte of the counter have no effect. Reads of the 16-bit counter should be made only by 16 bit access to maintain data coherency.

PWPERx — PWM Channel Period Registers

	Bit 7	6	5	4	3	2	1	Bit 0	
PWPER0	Bit 7	6	5	4	3	2	1	Bit 0	**$004C**
PWPER1	Bit 7	6	5	4	3	2	1	Bit 0	**$004D**
PWPER2	Bit 7	6	5	4	3	2	1	Bit 0	**$004E**
PWPER3	Bit 7	6	5	4	3	2	1	Bit 0	**$004F**
RESET	1	1	1	1	1	1	1	1	

Read and write anytime.
There is a dedicated period register for each channel. The value in the period register determines the period of the associated PWM channel. If written while the channel is enabled, the new value will not take effect until the existing period terminates, forcing the counter to reset. The new period is then latched and is used until a new period value is written. Reading this register returns the most recent value written. When in concatenated mode, the new period should be written with a 16-bit access to both channels registers (0 & 1, or 2 & 3) for data coherency. To start a new period immediately, write the new period value and then write the counter forcing a new period to start with the new period value.

Period = [Channel-Clock-Period x (PWPER + 1)] (CENTR = 0)
Period = [Channel-Clock-Period × (PWPER x 2)] (CENTR = 1)

NOTE

For Boundary Case programming values please refer to **11.2 PWM Boundary Cases**.

PWDTYx — PWM Channel Duty Registers

	Bit 7	6	5	4	3	2	1	Bit 0	
PWDTY0	Bit 7	6	5	4	3	2	1	Bit 0	**$0050**
PWDTY1	Bit 7	6	5	4	3	2	1	Bit 0	**$0051**
PWDTY2	Bit 7	6	5	4	3	2	1	Bit 0	**$0052**
PWDTY3	Bit 7	6	5	4	3	2	1	Bit 0	**$0053**
RESET	1	1	1	1	1	1	1	1	

Read and write anytime.

The value in each duty register determines the duty of the associated PWM channel. The duty value is compared to the counter and if it is equal to the counter value a match occurs and he output changes state. If the register is written to while the channel is enabled, then the new value is held in the buffer until the counter reaches zero or the channel is disabled and re-enabled. When in concatenated mode the new period should be written to both channels registers (0 & 1, or 2 & 3) with a 16 bit access for data coherency. Reading this register returns the most recent value written.

NOTE

For Boundary Case programming values please refer to **11.2 PWM Boundary Cases**.

Left-Aligned-Output Mode (CENTR = 0):

Duty cycle = [(PWDTYx + 1) / (PWPERx + 1)] × 100% (PPOLx = 1)

Duty cycle = [(PWPERx – PWDTYx) / (PWPERx + 1)] × 100% (PPOLx = 0)

Center-Aligned-Output Mode (CENTR = 1):

Duty cycle = [(PWPERx – PWDTYx) / PWPERx] × 100% (PPOLx = 1)

Duty cycle = (PWDTYx) / (PWPERx) × 100% (PPOLx = 0)

PWCTL— PWM Control Register **$0054**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	PSWAI	CENTR	RDP	PUPP	PSBCK
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

PSWAI — PWM Halts while in Wait Mode

0 = Allows PWM main clock generator to continue while in Wait mode.

1 = Halt PWM main clock generator when the part is in Wait mode.

CENTR — Center-Aligned Output Mode

Program change of CENTR bit should be done when PWM channels are disabled. All PWM counters should be written to as the last operation before enabling the channels. Otherwise, asserting/de-asserting the CENTR bit may cause irregularities in the PWM output.

0 = PWM channels operate in Left-Aligned Output mode

1 = PWM channels operate in Center-Aligned Output mode

RDP — Reduced Drive of Port P

0 = All port P output pins have normal drive capability.

1 = All port P output pins have reduced drive capability.

PUPP — Pull-Up Port P Enable

0 = All port P pins have an active pull-up device disabled.
1 = All port P pins have an active pull-up device enabled.

PSBCK — PWM Stops while in Background Mode

0 = Allows PWM to continue while in background mode. This is useful for emulation.
1 = Disable PWM input clock when the part is in background mode.

PWTST— PWM Special Mode Register (“Test”) **$0055**

	Bit 7	6	5	4	3	2	1	Bit 0
	DISCR	DISCP	DISCAL	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

Read anytime but write only in Special mode (SMOD = 1). These bits are available only in Special mode and are reset in Normal mode.
The PWM has some special test functions which are only accessible when the device is in Special Mode. Special Mode is indicated to the PWM module when the SMOD line on the LIB is asserted.
When the SMOD line is asserted, the Special Mode register control bits are accessed via the LIB. When SMOD is not asserted, writes to the Special Mode control bits have no effect ana all bits in the Special Mode register are forced to 0. This ensures that the PWM Special Mode can not be invoked inadvertently during normal operation.

DISCR — Disable Reset of Channel Counter on Write to Channel Counter

0 = Normal operation. Write to PWM channel counter will reset channel counter.
1 = Write to PWM channel counter does not reset channel counter.

DISCP — Disable Compare Count Period

0 = Normal Operation
1 = In Left-Aligned Output mode, match of period does not reset the associated PWM counter register.

DISCAL — Disable load of Scale-counters on write to the associated scale-registers

0 = Normal Operation
1 = Write to PWSCAL0 and PWSCAL1 does not load scale counters

PORTPP — Port P Data Register **$0056**

	Bit 7	6	5	4	3	2	1	Bit 0
	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
PWM	–	–	–	–	PWM3	PWM2	PWM1	PWM0
RESET:	–	–	–	–	–	–	–	–

PWM functions share port P pins 3 to 0 and take precedence over the general-purpose port when enabled.
READ: any time (input configured pin return pin level; output configured pin return internal latch pin driver input level.
WRITE: Data stored in an internal latch (drives pins only if configured for output and corresponding PWM channel not enabled).

NOTE

Writes do not change pin state when pin is configured for PWM outputs, only after the PWM channel becomes available on port P pin, see PWEN bit description.

.

PORTPD — Port P Data Direction Register **$0057**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
RESET:	0	0	0	0	0	0	0	0

This register determines whether each pin is input or output when it is selected to be a general purpose I/O pin. Reading Port P Data Register always follows pin data associated with he Port P Data Direction bit. Read and write any time.

DDRP[7:0] — Data Direction Port P pins
0 = Configure I/O pin for input only
1 = Configure I/O for output

11.2 PWM Boundary Cases

The boundary conditions for the PWM channel duty registers and the PWM channel period registers cause these results:

Table 23 PWM Boundary Conditions

PWDTYx	PWPERx	PPOLx	Output
$FF	>$00	1	Low
$FF	>$00	0	High
≥PWPERx	–	1	High
≥PWPERx	–	0	Low
–	$00	1	High
–	$00	0	Low

12 Standard Timer Module

The standard timer module consists of a 16-bit software-programmable counter driven by a prescaler. It contains eight complete 16-bit input capture/output compare channels and one 16-bit pulse accumulator.

This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from less than a microsecond to many seconds. It can also generate PWM signals without CPU intervention.

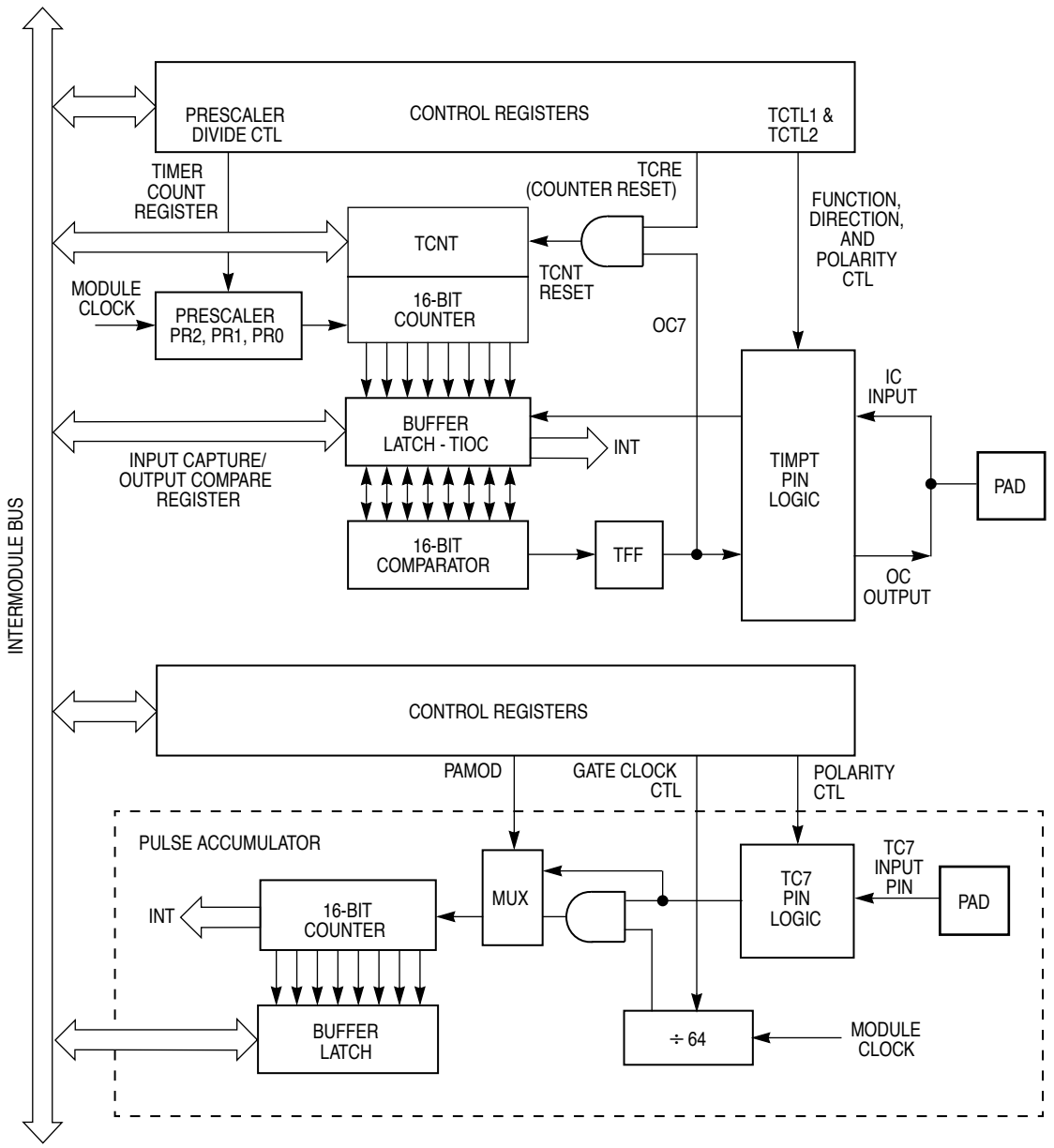


Figure 18 Timer Block Diagram: Input Capture, Output Compare, Pulse Accumulator

12.1 Timer Registers

Input/output pins default to general-purpose I/O lines until an internal function which uses that pin is specifically enabled. The timer overrides the state of the DDR to force the I/O state of each associated port line when an output compare using a port line is enabled. In these cases the data direction bits will have no affect on these lines.

When a pin is assigned to output an on-chip peripheral function, writing to this PORTTn bit does not affect the pin but the data is stored in an internal latch such that if the pin becomes available for general-purpose output the driven level will be the last value written to the PORTTn bit.

TIOS — Timer Input Capture/Output Compare Select **$0080**

	Bit 7	6	5	4	3	2	1	Bit 0
	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

IOS[7:0] — Input Capture or Output Compare Channel Configuration
0 = The corresponding channel acts as an input capture
1 = The corresponding channel acts as an output compare.

CFORC — Timer Compare Force Register **$0081**

	Bit 7	6	5	4	3	2	1	Bit 0
	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
RESET:	0	0	0	0	0	0	0	0

Read anytime but will always return $00 (1 state is transient). Write anytime.

FOC[7:0] — Force Output Compare Action for Channel 7-0
A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “n” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCn register except the interrupt flag does not get set.

OC7M — Output Compare 7 Mask Register **$0082**

	Bit 7	6	5	4	3	2	1	Bit 0
	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.
The bits of OC7M correspond bit-for-bit with the bits of timer port (PORTT). Setting the OC7Mn will set the corresponding port to be an output port regardless of the state of the DDRTn bit when the corresponding TIOSn bit is set to be an output compare. This does not change the state of the DDRT bits.

OC7D — Output Compare 7 Data Register **$0083**

	Bit 7	6	5	4	3	2	1	Bit 0
	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.
The bits of OC7D correspond bit-for-bit with the bits of timer port (PORTT). When a successful OC7 compare occurs, for each bit that is set in OC7M, the corresponding data bit in OC7D is stored to the corresponding bit of the timer port.

When the OC7Mn bit is set, a successful OC7 action will override a successful OC[6:0] compare action during the same cycle; therefore, the OCn action taken will depend on the corresponding OC7D bit.

TCNT — Timer Count Register **$0084–$0085**

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

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.
Read anytime.
Write has no meaning or effect in the normal mode; only writable in special modes (SMODN = 0).
The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

TSCR — Timer System Control Register **$0086**

	Bit 7	6	5	4	3	2	1	Bit 0
	TEN	TSWAI	TSBCK	TFFCA	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

TEN — Timer Enable
0 = Disables the timer, including the counter. Can be used for reducing power consumption.
1 = Allows the timer to function normally.
If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator since the E÷64 is generated by the timer prescaler.

TSWAI — Timer Stops While in Wait
0 = Allows the timer to continue running during wait.
1 = Disables the timer when the MCU is in the wait mode. Timer interrupts cannot be used to get the MCU out of wait.

TSBCK — Timer Stops While in Background Mode
0 = Allows the timer to continue running while in background mode.
1 = Disables the timer whenever the MCU is in background mode. This is useful for emulation.

TFFCA — Timer Fast Flag Clear All
0 = Allows the timer flag clearing to function normally.
1 = For TFLG1($8E), a read from an input capture or a write to the output compare channel ($90–$9F) causes the corresponding channel flag, CnF, to be cleared. For TFLG2 ($8F), any access to the TCNT register ($84, $85) clears the TOF flag. Any access to the PACNT register ($A2, $A3) clears the PAOVF and PAIF flags in the PAFLG register ($A1). This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses.

TQCR — Reserved **$0087**

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

TCTL1 — Timer Control Register 1 **$0088**

	Bit 7	6	5	4	3	2	1	Bit 0
	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
RESET:	0	0	0	0	0	0	0	0

TCTL2 — Timer Control Register 2 **$0089**

	Bit 7	6	5	4	3	2	1	Bit 0
	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

OMn — Output Mode

OLn — Output Level

These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCn compare. When either OMn or OLn is one, the pin associated with OCn becomes an output tied to OCn regardless of the state of the associated DDRT bit.

Table 24 Compare Result Output Action

OMn	OLn	Action
0	0	Timer disconnected from output pin logic
0	1	Toggle OCn output line
1	0	Clear OCn output line to zero
1	1	Set OCn output line to one

TCTL3 — Timer Control Register 3 **$008A**

	Bit 7	6	5	4	3	2	1	Bit 0
	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
RESET:	0	0	0	0	0	0	0	0

TCTL4 — Timer Control Register 4 **$008B**

	Bit 7	6	5	4	3	2	1	Bit 0
	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

EDGnB, EDGnA — Input Capture edge control

These eight pairs of control bits configure the input capture edge detector circuits.

Table 25 Edge Detector Circuit Configuration

EDGnB	EDGnA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge (rising or falling)

TMSK1 — Timer Interrupt Mask 1 **$008C**

	Bit 7	6	5	4	3	2	1	Bit 0
	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
RESET:	0	0	0	0	0	0	0	0

The bits in TMSK1 correspond bit-for-bit with the bits in the TFLG1 status register. If cleared, the corresponding flag is disabled from causing a hardware interrupt. If set, the corresponding flag is enabled to cause a hardware interrupt.
Read or write anytime.

C7I–C0I—Input Capture/Output Compare “x” Interrupt enable.

TMSK2 — Timer Interrupt Mask 2 **$008D**

	Bit 7	6	5	4	3	2	1	Bit 0
	TOI	0	PUPT	RDPT	TCRE	PR2	PR1	PR0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

TOI — Timer Overflow Interrupt Enable
0 = Interrupt inhibited
1 = Hardware interrupt requested when TOF flag set

PUPT — Timer Pull-Up Resistor Enable
This enable bit controls pull-up resistors on the timer port pins when the pins are configured as inputs.
1 = Enable pull-up resistor function
0 = Disable pull-up resistor function

RDPT — Timer Drive Reduction
This bit reduces the effective output driver size which can reduce power supply current and generated noise depending upon pin loading.
1 = Enable output drive reduction function
0 = Normal output drive capability

TCRE — Timer Counter Reset Enable
This bit allows the timer counter to be reset by a successful output compare 7 event.
0 = Counter reset inhibited and counter free runs
1 = Counter reset by a successful output compare 7
If TC7 = $0000 and TCRE = 1, TCNT will stay at $0000 continuously. If TC7 = $FFFF and TCRE = 1, TOF will never get set even though TCNT will count from $0000 through $FFFF.

PR2, PR1, PR0 — Timer Prescaler Select
These three bits specify the number of ÷2 stages that are to be inserted between the module clock and the timer counter.

Table 26 Prescaler Selection

PR2	PR1	PR0	Prescale Factor
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	Reserved
1	1	1	Reserved

The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero.

TFLG1 — Timer Interrupt Flag 1 **$008E**

	Bit 7	6	5	4	3	2	1	Bit 0
	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
RESET:	0	0	0	0	0	0	0	0

TFLG1 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write a one to the bit.
Read anytime. Write used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a zero will not effect current status of the bit.
When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel ($90–$9F) will cause the corresponding channel flag CnF to be cleared.

C7F–C0F — Input Capture/Output Compare Channel “n” Flag.

TFLG2 — Timer Interrupt Flag 2 **$008F**

	Bit 7	6	5	4	3	2	1	Bit 0
	TOF	0	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, set the bit to one.
Read anytime. Write used in clearing mechanism (bits set cause corresponding bits to be cleared).
Any access to TCNT will clear TFLG2 register if the TFFCA bit in TSCR register is set.

TOF — Timer Overflow Flag
Set when 16-bit free-running timer overflows from $FFFF to $0000. This bit is cleared automatically by a write to the TFLG2 register with bit 7 set. (See also TCRE control bit explanation.)

TC0 — Timer Input Capture/Output Compare Register 0 **$0090–$0091**

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

TC1 — Timer Input Capture/Output Compare Register 1 **$0092–$0093**

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

TC2 — Timer Input Capture/Output Compare Register 2 **$0094–$0095**

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

TC3 — Timer Input Capture/Output Compare Register 3 **$0096–$0097**

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

TC4 — Timer Input Capture/Output Compare Register 4 **$0098–$0099**

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

TC5 — Timer Input Capture/Output Compare Register 5 **$009A–$009B**

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

TC6 — Timer Input Capture/Output Compare Register 6 **$009C–$009D**

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

TC7 — Timer Input Capture/Output Compare Register 7 **$009E–$009F**

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

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

Read anytime. Write anytime for output compare function. Writes to these registers have no meaning or effect during input capture. All timer input capture/output compare registers are reset to $0000.

PACTL — Pulse Accumulator Control Register **$00A0**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

PAEN — Pulse Accumulator System Enable
0 = Pulse Accumulator system disabled
1 = Pulse Accumulator system enabled
PAEN is independent from TEN.

PAMOD — Pulse Accumulator Mode
0 = Event counter mode
1 = Gated time accumulation mode

PEDGE — Pulse Accumulator Edge Control
For PAMOD = 0 (event counter mode)
0 = Falling edges on the pulse accumulator input pin (PT7/PAI) cause the count to be incremented
1 = Rising edges on the pulse accumulator input pin cause the count to be incremented
For PAMOD = 1 (gated time accumulation mode)
0 = Pulse accumulator input pin high enables E÷64 clock to pulse accumulator and the trailing falling edge on the pulse accumulator input pin sets the PAIF flag.
1 = Pulse accumulator input pin low enables E÷64 clock to pulse accumulator and the trailing rising edge on the pulse accumulator input pin sets the PAIF flag.
If the timer is not active (TEN = 0 in TSCR), there is no ÷64 clock since the E÷64 clock is generated by the timer prescaler.

CLK1, CLK0 — Clock Select Register

Table 27 Clock Selection

CLK1	CLK0	Selected Clock
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65536 as timer counter clock frequency

If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

PAOVI — Pulse Accumulator Overflow Interrupt Enable
0 = Interrupt inhibited
1 = Interrupt requested if PAOVF is set

PAI — Pulse Accumulator Input Interrupt Enable
0 = Interrupt inhibited
1 = Interrupt requested if PAIF is set

PAFLG — Pulse Accumulator Flag Register **$00A1**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	PAOVF	PAIF
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.
When TFFCA bit in the TSCR register is set, any access to the PACNT register will clear all the flags in the PAFLG register.

PAOVF — Pulse Accumulator Overflow Flag
Set when the 16-bit pulse accumulator overflows from $FFFF to $0000. This bit is cleared automatically by a write to the PAFLG register with bit 1 set.

PAIF — Pulse Accumulator Input Edge Flag
Set when the selected edge is detected at the pulse accumulator input pin. In event mode, the event edge triggers PAIF. In gated time accumulation mode, the trailing edge of the gate signal at the pulse accumulator input pin triggers PAIF. This bit is cleared automatically by a write to the PAFLG register with bit 0 set.

PACNT — 16-bit Pulse Accumulator Count Register **$00A2–$00A3**

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

Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.
Read or write anytime.

TIMTST — Timer Test Register **$00AD**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	TCBYP	PCBYP
RESET:	0	0	0	0	0	0	0	0

Read anytime. Write only in special mode (SMODN = 0)

TCBYP — Timer Divider Chain Bypass
0 = Normal operation
1 = The 16-bit free-running timer counter is divided into two 8-bit halves and the prescaler is bypassed. The clock drives both halves directly.

PCBYP — Pulse Accumulator Divider Chain Bypass
0 = Normal operation
1 = The 16-bit pulse accumulator counter is divided into two 8-bit halves and the prescaler is bypassed. The clock drives both halves directly.

PORTT — Timer Port Data Register **$00AE**

	Bit 7	6	5	4	3	2	1	Bit 0
	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0
TIMER	I/OC7	I/OC6	I/OC5	I/OC4	I/OC3	I/OC2	I/OC1	I/OC0
PA	PAI							

PORTT can be read anytime. When configured as an input, a read will return the pin level. When configured as output, a read will return the latched output data.

NOTE

Writes do not change pin state when the pin is configured for timer output. The minimum pulse width for pulse accumulator input should always be greater than two module clocks due to input synchronizer circuitry. The minimum pulse width for the input capture should always be greater than the width of two module clocks due to input synchronizer circuitry.

DDRT — Data Direction Register for Timer Port **$00AF**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

0 = Configures the corresponding I/O pin for input only

1 = Configures the corresponding I/O pin for output

The timer forces the I/O state to be an output for each timer port pin associated with an enabled output compare. In these cases the data direction bits will not be changed but have no affect on the direction of these pins. The DDRT will revert to controlling the I/O direction of a pin when the associated timer output compare is disabled. Input captures do not override the DDRT settings.

12.2 Timer Operation in Modes

STOP:Timer is off since both PCLK and ECLK are stopped.
BDM:Timer keeps running, unless TSBCK = 1
WAIT:Counters keep running, unless TSWAI = 1
NORMAL:Timer keeps running, unless TEN = 0
TEN = 0:All timer operations are stopped, registers may be accessed.
Gated pulse accumulator ÷64 clock is also disabled.
PAEN = 0:All pulse accumulator operations are stopped, registers may be accessed.

13 Serial Interface

The serial interface of the MC68HC912BC32 consists of two independent serial I/O sub-systems: the serial communication interface (SCI) and the serial peripheral interface (SPI). Each serial pin shares function with the general-purpose port pins of port S. The SCI is an NRZ type system that is compatible with standard RS-232 systems. The SCI system has a single wire operation mode which allows the unused pin to be available as general-purpose I/O. The SPI subsystem, which is compatible with the M68HC11 SPI, includes new features such as $\overline{SS}$ output and bidirectional mode.

13.1 Block diagram

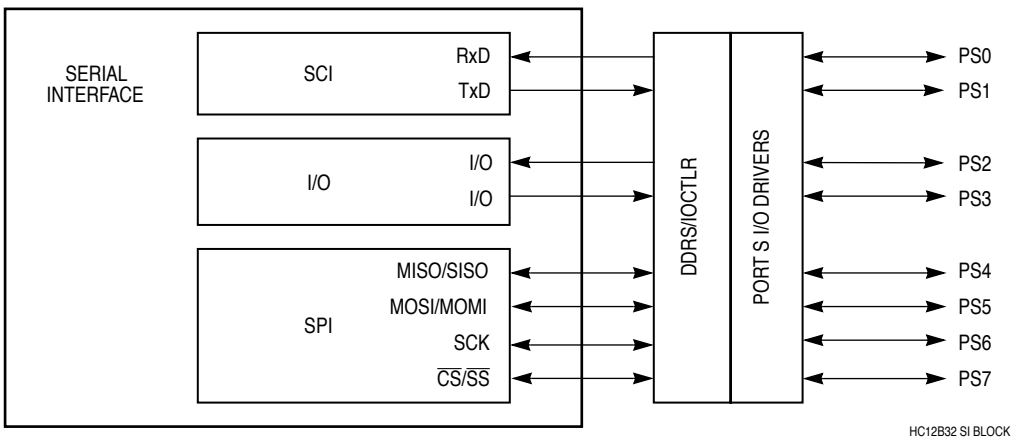


Figure 19 Serial Interface Block Diagram

13.2 Serial Communication Interface (SCI)

The serial communication interface on the MC68HC912BC32 is an NRZ format (one start, eight or nine data, and one stop bit) asynchronous communication system with independent internal baud rate generation circuitry and an SCI transmitter and receiver. It can be configured for eight or nine data bits (one of which may be designated as a parity bit, odd or even). If enabled, parity is generated in hardware for transmitted and received data. Receiver parity errors are flagged in hardware. The baud rate generator is based on a modulus counter, allowing flexibility in choosing baud rates. There is a receiver wake-up feature, an idle line detect feature, a loop-back mode, and various error detection features. Two port pins provide the external interface for the transmitted data (TXD) and the received data (RXD).

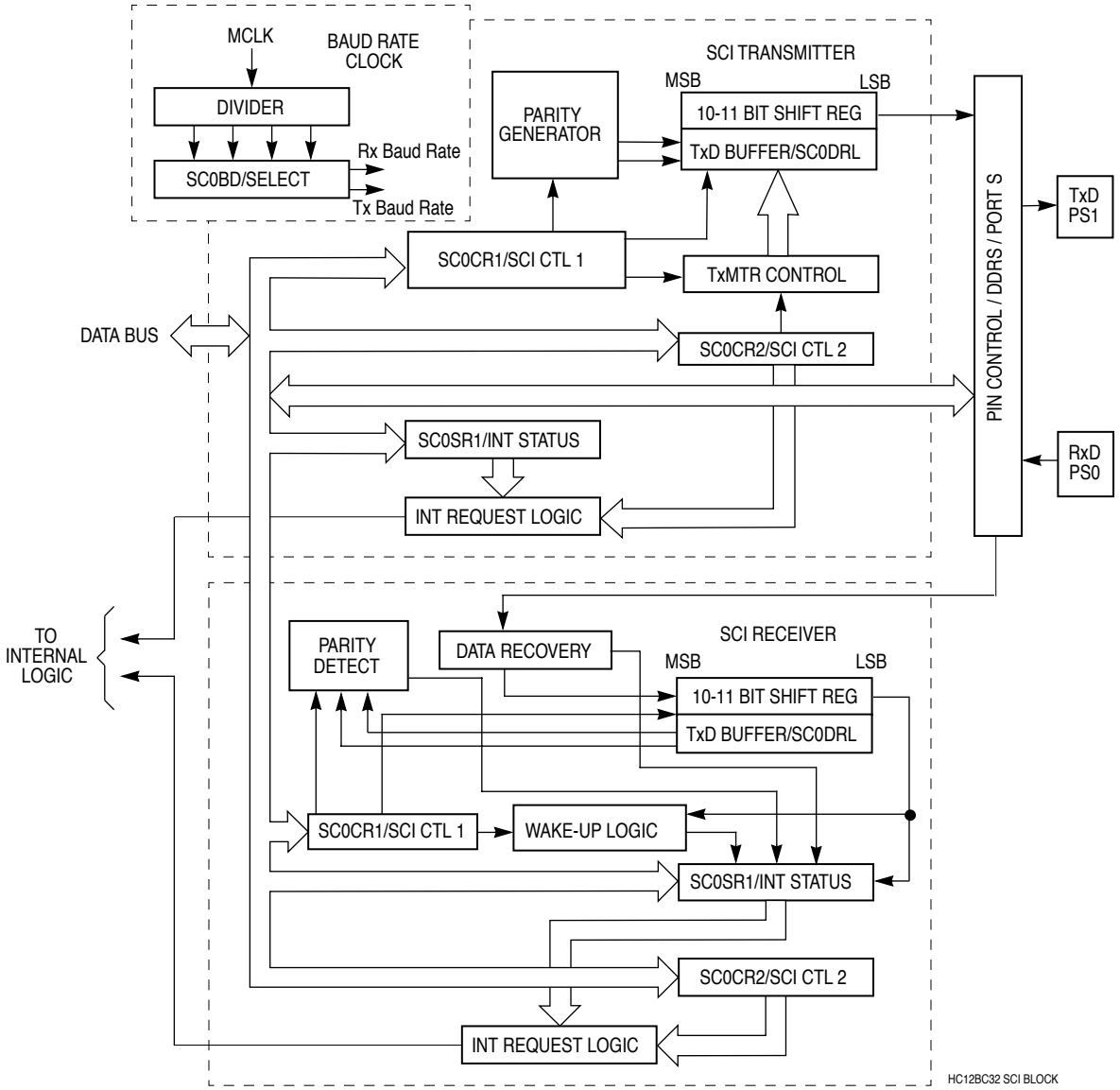


Figure 20 Serial Communications Interface Block Diagram

13.2.1 Data Format

The serial data format requires the following conditions:

- An idle-line in the high state before transmission or reception of a message.
- A start bit (logic zero), transmitted or received, that indicates the start of each character.
- Data that is transmitted or received least significant bit (LSB) first.
- A stop bit (logic one), used to indicate the end of a frame. (A frame consists of a start bit, a character of eight or nine data bits and a stop bit.)
- A BREAK is defined as the transmission or reception of a logic zero for one frame or more.
- This SCI supports hardware parity for transmit and receive.

13.2.2 SCI Baud Rate Generation

The basis of the SCI baud rate generator is a 13-bit modulus counter. This counter gives the generator the flexibility necessary to achieve a reasonable level of independence from the CPU operating frequency and still be able to produce standard baud rates with a minimal amount of error. The clock source for the generator comes from the P Clock.

Table 28 Baud Rate Generation

Desired SCI Baud Rate	BR Divisor for P = 4.0 MHz	BR Divisor for P = 8.0 MHz
110	2273	4545
300	833	1667
600	417	833
1200	208	417
2400	104	208
4800	52	104
9600	26	52
14400	17	35
19200	13	26
38400	—	13

13.2.3 Register Descriptions

Control and data registers for the SCI subsystem are described below. The memory address indicated for each register is the default address that is in use after reset. The entire 512-byte register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space.

SC0BDH — SCI Baud Rate Control Register **$00C0**

	Bit 7	6	5	4	3	2	1	Bit 0	
	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8	High
RESET:	0	0	0	0	0	0	0	0	

SC0BDL — SCI Baud Rate Control Register **$00C1**

	Bit 7	6	5	4	3	2	1	Bit 0	
	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0	Low
RESET:	0	0	0	0	0	1	0	0	

SC0BDH and SC0BDL are considered together as a 16-bit baud rate control register.
Read any time. Write SBR[12:0] anytime. Low order byte must be written for change to take effect. Write SBR[15:13] only in special modes. The value in SBR[12:0] determines the baud rate of the SCI. The desired baud rate is determined by the following formula:

$$\text{SCI Baud Rate} = \frac{\text{MCLK}}{16 \times \text{BR}}$$

which is equivalent to:

$$\text{BR} = \frac{\text{MCLK}}{16 \times \text{SCI Baud Rate}}$$

BR is the value written to bits SBR[12:0] to establish baud rate.

NOTE

The baud rate generator is disabled until the TE or RE bit in SC0CR2 register is set for the first time after reset, and/or the baud rate generator is disabled when SBR[12:0] = 0.

BTST — Baud Register Test
Reserved for test function

BSPL — Baud Rate Counter Split
Reserved for test function

BRLD — Baud Rate Reload
Reserved for test function

SC0CR1 — SCI Control Register 1 **$00C2**

	Bit 7	6	5	4	3	2	1	Bit 0
	LOOPS	WOMS	RSRC	M	WAKE	ILT	PE	PT
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

LOOPS — SCI LOOP Mode/Single Wire Mode Enable
0 = SCI transmit and receive sections operate normally.
1 = SCI receive section is disconnected from the RXD pin and the RXD pin is available as general purpose I/O. The receiver input is determined by the RSRC bit. The transmitter output is controlled by the associated DDRS bit. Both the transmitter and the receiver must be enabled to use the LOOP or the single wire mode.
If the DDRS bit associated with the TXD pin is set during the LOOPS = 1, the TXD pin outputs the SCI waveform. If the DDRS bit associated with the TXD pin is clear during the LOOPS = 1, the TXD pin becomes high (IDLE line state) for RSRC = 0 and high impedance for RSRC = 1. Refer to **Table 29**.

WOMS — Wired-Or Mode for Serial Pins
This bit controls the two pins (TXD and RXD) associated with the SCI section.
0 = Pins operate in a normal mode with both high and low drive capability. To affect the RXD bit, that bit would have to be configured as an output (via DDRS) which is the single wire case when using the SCI. WOMS bit still affects general-purpose output on TXD and RXD pins when SCI is not using these pins.
1 = Each pin operates in an open drain fashion if that pin is declared as an output.

RSRC — Receiver Source
When LOOPS = 1, the RSRC bit determines the internal feedback path for the receiver.
0 = Receiver input is connected to the transmitter internally (not TXD pin)
1 = Receiver input is connected to the TXD pin

Table 29 Loop Mode Functions

LOOPS	RSRC	DDS1(3)	WOMS	Function of Port S Bit 1/3
0	x	x	x	Normal Operations
1	0	0	0/1	LOOP mode without TXD output(TXD = High Impedance)
1	0	1	0	LOOP mode with TXD output (CMOS)
1	0	1	1	LOOP mode with TXD output (open-drain)
1	1	0	x	Single wire mode without TXD output (the pin is used as receiver input only, TXD = High Impedance)
1	1	1	0	Single wire mode with TXD output (the output is also fed back to receiver input, CMOS)
1	1	1	1	Single wire mode for the receiving and transmitting(open-drain)

M — Mode (select character format)

0 = One start, eight data, one stop bit

1 = One start, eight data, ninth data, one stop bit

WAKE — Wakeup by Address Mark/Idle

0 = Wake up by IDLE line recognition

1 = Wake up by address mark (last data bit set)

ILT — Idle Line Type

Determines which of two types of idle line detection will be used by the SCI receiver.

0 = Short idle line mode is enabled.

1 = Long idle line mode is detected.

In the short mode, the SCI circuitry begins counting ones in the search for the idle line condition immediately after the start bit. This means that the stop bit and any bits that were ones before the stop bit could be counted in that string of ones, resulting in earlier recognition of an idle line.

In the long mode, the SCI circuitry does not begin counting ones in the search for the idle line condition until a stop bit is received. Therefore, the last byte's stop bit and preceding "1" bits do not affect how quickly an idle line condition can be detected.

PE — Parity Enable

0 = Parity is disabled.

1 = Parity is enabled.

PT — Parity Type

If parity is enabled, this bit determines even or odd parity for both the receiver and the transmitter.

0 = Even parity is selected. An even number of ones in the data character causes the parity bit to be zero and an odd number of ones causes the parity bit to be one.

1 = Odd parity is selected. An odd number of ones in the data character causes the parity bit to be zero and an even number of ones causes the parity bit to be one.

SC0CR2 — SCI Control Register 2 **$00C3**

	Bit 7	6	5	4	3	2	1	Bit 0
	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

TIE — Transmit Interrupt Enable

0 = TDRE interrupts disabled

1 = SCI interrupt will be requested whenever the TDRE status flag is set.

TCIE — Transmit Complete Interrupt Enable
0 = TC interrupts disabled
1 = SCI interrupt will be requested whenever the TC status flag is set.

RIE — Receiver Interrupt Enable
0 = RDRF and OR interrupts disabled
1 = SCI interrupt will be requested whenever the RDRF status flag or the OR status flag is set.

ILIE — Idle Line Interrupt Enable
0 = IDLE interrupts disabled
1 = SCI interrupt will be requested whenever the IDLE status flag is set.

TE — Transmitter Enable
0 = Transmitter disabled
1 = SCI transmit logic is enabled and the TXD pin (Port S bit 1) is dedicated to the transmitter. The TE bit can be used to queue an idle preamble.

RE — Receiver Enable
0 = Receiver disabled
1 = Enables the SCI receive circuitry

RWU — Receiver Wake-Up Control
0 = Normal SCI Receiver
1 = Enables the wake-up function and inhibits further receiver interrupts. Normally hardware wakes the receiver by automatically clearing this bit.

SBK — Send Break
0 = Break generator off
1 = Generate a break code (at least 10 or 11 contiguous zeros)
As long as SBK remains set the transmitter will send zeros. When SBK is changed to zero, the current frame of all zeros is finished before the TxD line goes to the idle state. If SBK is toggled on and off, the transmitter will send 10 (or 11) zeros and then revert to mark idle or sending data.

SC0SR1 — SCI Status Register 1 **$00C4**

	Bit 7	6	5	4	3	2	1	Bit 0
	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
RESET:	1	1	0	0	0	0	0	0

The bits in these registers are set by various conditions in the SCI hardware and are automatically cleared by special acknowledge sequences. The receive related flag bits in SC0SR1 (RDRF, IDLE, OR, NF, FE, and PF) are all cleared by a read of the SC0SR1 register followed by a read of the transmit/receive data register L. However, only those bits which were set when SC0SR1 was read will be cleared by the subsequent read of the transmit/receive data register L. The transmit related bits in SC0SR1 (TDRE and TC) are cleared by a read of the SC0SR1 register followed by a write to the transmit/receive data register L.
Read anytime (used in auto clearing mechanism). Write has no meaning or effect.

TDRE — Transmit Data Register Empty Flag
New data will not be transmitted unless SC0SR1 is read before writing to the transmit data register. Reset sets this bit.
0 = SC0DR busy
1 = Any byte in the transmit data register is transferred to the serial shift register so new data may now be written to the transmit data register.

TC — Transmit Complete Flag

Flag is set when the transmitter is idle (no data, preamble, or break transmission in progress). Clear by reading SC0SR1 with TC set and then writing to SC0DR.

0 = Transmitter busy
1 = Transmitter is idle

RDRF — Receive Data Register Full Flag

Once cleared, IDLE is not set again until the RxD line has been active and becomes idle again. RDRF is set if a received character is ready to be read from SC0DR. Clear the RDRF flag by reading SC0SR1 with RDRF set and then reading SC0DR.

0 = SC0DR empty
1 = SC0DR full

IDLE — Idle Line Detected Flag

Receiver idle line is detected (the receipt of a minimum of 10/11 consecutive ones). This bit will not be set by the idle line condition when the RWU bit is set. Once cleared, IDLE will not be set again until after RDRF has been set (after the line has been active and becomes idle again).

0 = RxD line is idle
1 = RxD line is active

OR — Overrun Error Flag

New byte is ready to be transferred from the receive shift register to the receive data register and the receive data register is already full (RDRF bit is set). Data transfer is inhibited until this bit is cleared.

0 = No overrun
1 = Overrun detected

NF — Noise Error Flag

Set during the same cycle as the RDRF bit but not set in the case of an overrun (OR).

0 = Unanimous decision
1 = Noise on a valid start bit, any of the data bits, or on the stop bit

FE — Framing Error Flag

Set when a zero is detected where a stop bit was expected. Clear the FE flag by reading SC0SR1 with FE set and then reading SC0DR.

0 = Stop bit detected
1 = Zero detected rather than a stop bit

PF — Parity Error Flag

Indicates if received data's parity matches parity bit. This feature is active only when parity is enabled. The type of parity tested for is determined by the PT (parity type) bit in SC0CR1.

0 = Parity correct
1 = Incorrect parity detected

SC0SR2 — SCI Status Register 2 **$00C5**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	0	RAF
RESET:	0	0	0	0	0	0	0	0

Read anytime. Write has no meaning or effect.

RAF — Receiver Active Flag

This bit is controlled by the receiver front end. It is set during the RT1 time period of the start bit search. It is cleared when an idle state is detected or when the receiver circuitry detects a false start bit (generally due to noise or baud rate mismatch).

0 = A character is not being received
1 = A character is being received

SC0DRH — SCI Data Register High **$00C6**

	Bit 7	6	5	4	3	2	1	Bit 0
	R8	T8	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

SC0DRL — SCI Data Register Low **$00C7**

	Bit 7	6	5	4	3	2	1	Bit 0
	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	0	0	0	0	0	0	0	0

R8 — Receive Bit 8

Read anytime. Write has no meaning or affect.
This bit is the ninth serial data bit received when the SCI system is configured for nine-data-bit operation.

T8 — Transmit Bit 8

Read or write anytime.
This bit is the ninth serial data bit transmitted when the SCI system is configured for nine-data-bit operation. When using 9-bit data format this bit does not have to be written for each data word. The same value will be transmitted as the ninth bit until this bit is rewritten.

R7/T7–R0/T0 — Receive/Transmit Data Bits 7 to 0

Reads access the eight bits of the read-only SCI receive data register (RDR). Writes access the eight bits of the write-only SCI transmit data register (TDR). SC0DRL:SC0DRH form the 9-bit data word for the SCI. If the SCI is being used with a 7- or 8-bit data word, only SC0DRL needs to be accessed. If a 9-bit format is used, the upper register should be written first to ensure that it is transferred to the transmitter shift register with the lower register.

13.3 Serial Peripheral Interface (SPI)

The serial peripheral interface allows the MC68HC912BC32 to communicate synchronously with peripheral devices and other microprocessors. The SPI system in the MC68HC912BC32 can operate as a master or as a slave. The SPI is also capable of interprocessor communications in a multiple master system.

When the SPI is enabled, all pins that are defined by the configuration as inputs will be inputs regardless of the state of the DDRS bits for those pins. All pins that are defined as SPI outputs will be outputs only if the DDRS bits for those pins are set. Any SPI output whose corresponding DDRS bit is cleared can be used as a general-purpose input.

A bidirectional serial pin is possible using the DDRS as the direction control.

13.3.1 SPI Baud Rate Generation

The P clock is input to a divider series and the resulting SPI clock rate may be selected to be P divided by 2, 4, 8, 16, 32, 64, 128 or 256. Three bits in the SP0BR register control the SPI clock rate. This baud rate generator is activated only when SPI is in the master mode and serial transfer is taking place. Otherwise this divider is disabled to save power.

13.3.2 SPI Operation

In the SPI system the 8-bit data register in the master and the 8-bit data register in the slave are linked to form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master so the data is effectively exchanged between the master and the slave. Data written to the SP0DR register of the master becomes the output data for the slave and data read from the SP0DR register of the master after a transfer operation is the input data from the slave.

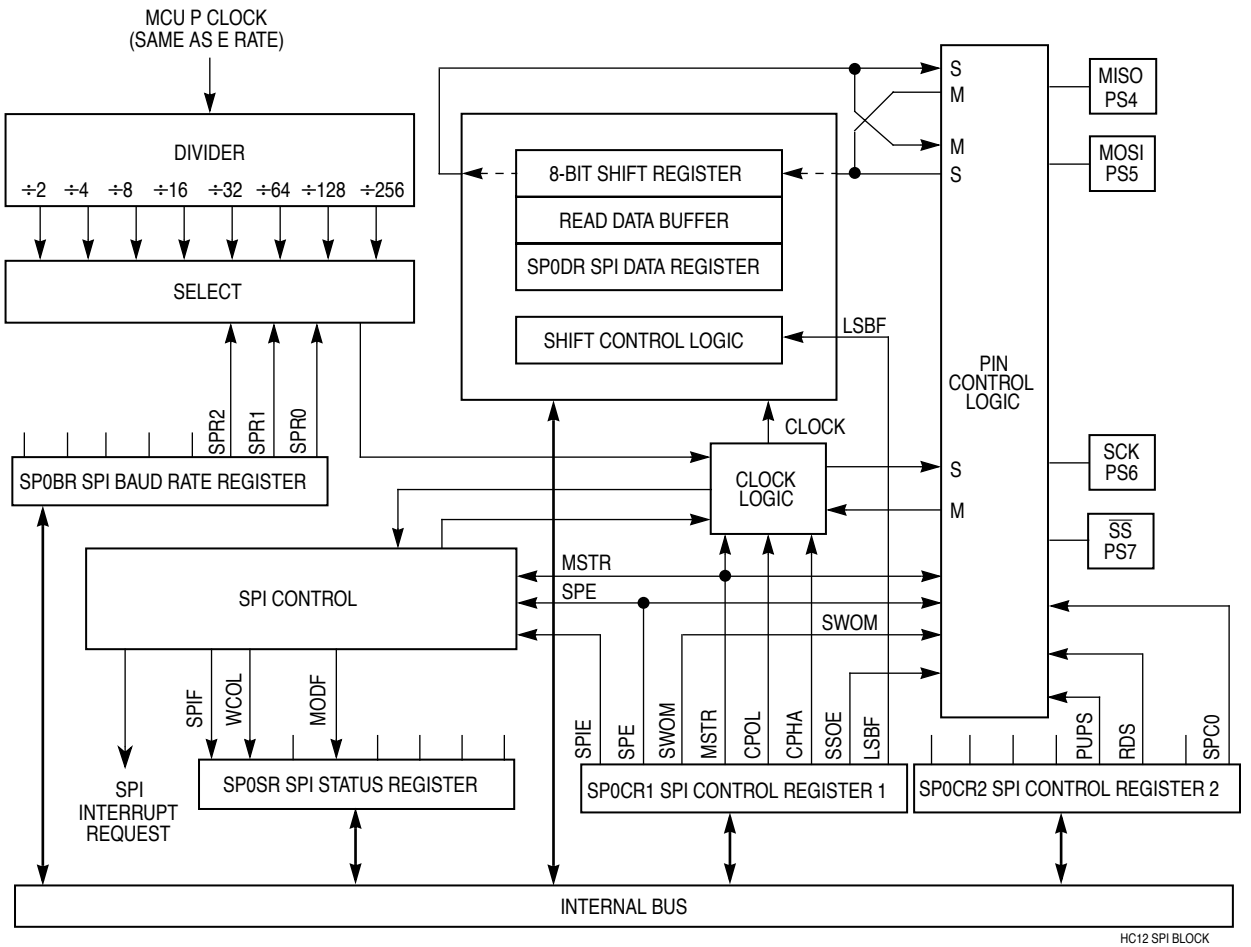


Figure 21 Serial Peripheral Interface Block Diagram

A clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SP0CR1 register select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by shifting the clock by one half cycle or no phase shift.

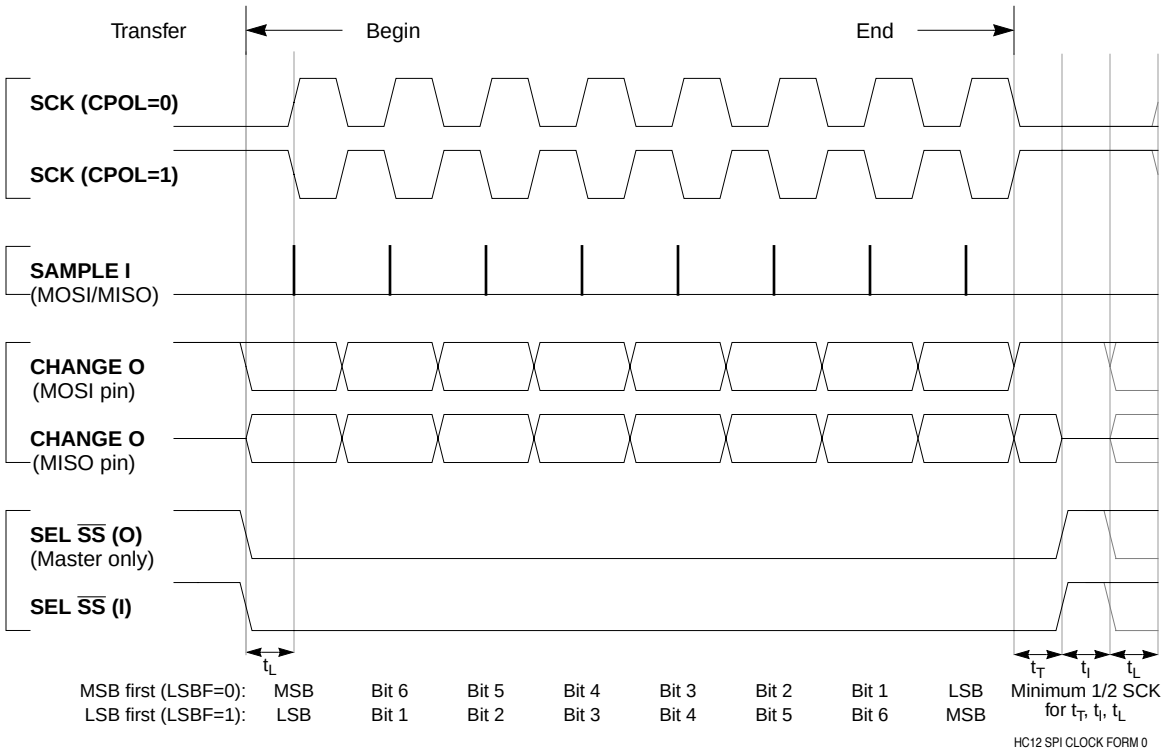


Figure 22 SPI Clock Format 0 (CPHA = 0)

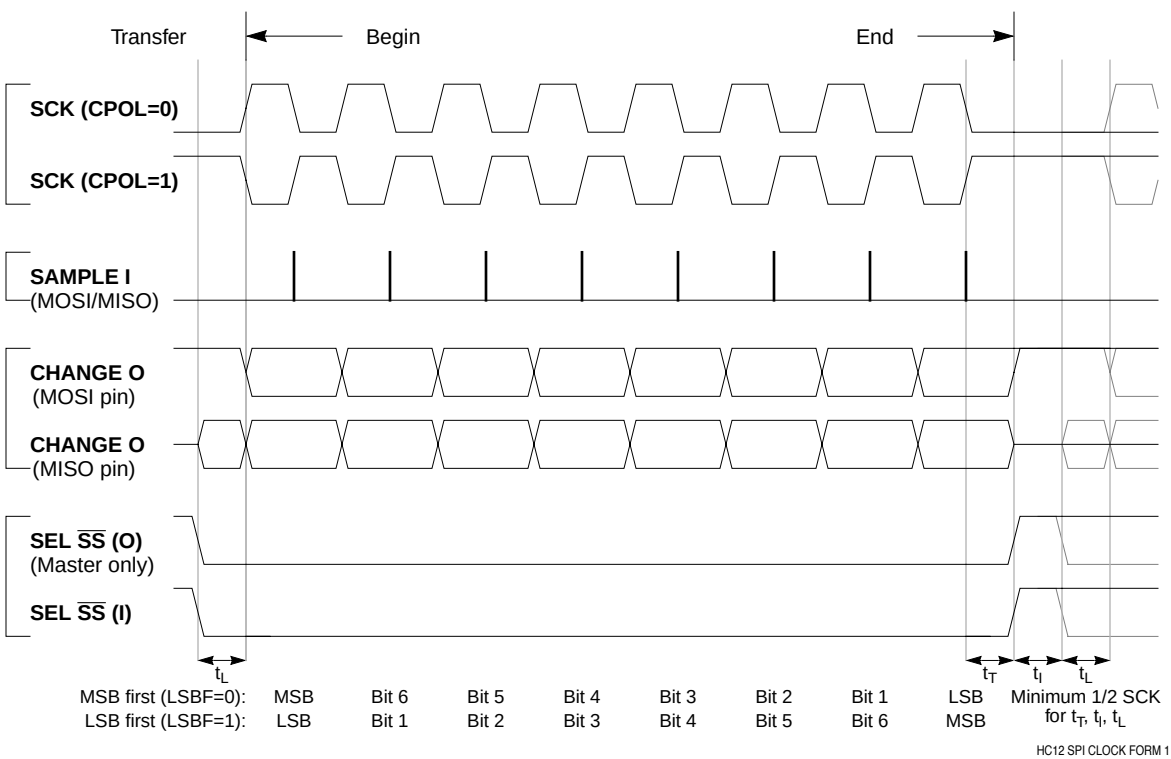


Figure 23 SPI Clock Format 1 (CPHA = 1)

13.3.3 $\overline{SS}$ Output

Available in master mode only, $\overline{SS}$ output is enabled with the SSOE bit in the SP0CR1 register if the corresponding DDRS bit is set. The $\overline{SS}$ output pin will be connected to the $\overline{SS}$ input pin of the external slave device. The $\overline{SS}$ output automatically goes low for each transmission to select the external device and it goes high during each idling state to deselect external devices.

Table 30 $\overline{SS}$ Output Selection

DDS7	SSOE	Master Mode	Slave Mode
0	0	$\overline{SS}$ Input with MODF Feature	$\overline{SS}$ Input
0	1	Reserved	$\overline{SS}$ Input
1	0	General-Purpose Output	$\overline{SS}$ Input
1	1	$\overline{SS}$ Output	$\overline{SS}$ Input

13.3.4 Bidirectional Mode (MOMI or SISO)

In bidirectional mode, the SPI uses only one serial data pin for external device interface. The MSTR bit decides which pin to be used. The MOSI pin becomes serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The direction of each serial I/O pin depends on the corresponding DDRS bit.

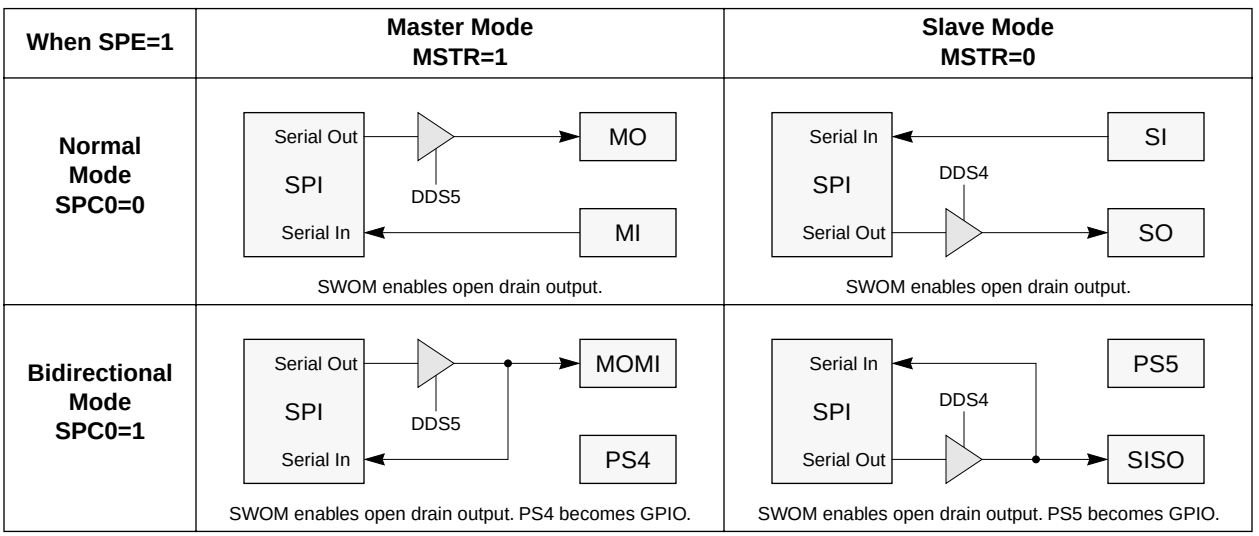


Figure 24 Normal Mode and Bidirectional Mode

13.3.5 Register Descriptions

Control and data registers for the SPI subsystem are described below. The memory address indicated for each register is the default address that is in use after reset. The entire 512-byte register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space. For more information refer to **5 Operating Modes and Resource Mapping**.

SP0CR1 — SPI Control Register 1 **$00D0**

	Bit 7	6	5	4	3	2	1	Bit 0
	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF
RESET:	0	0	0	0	0	1	0	0

Read or write anytime.

SPIE — SPI Interrupt Enable
1 = Hardware interrupt sequence is requested each time the SPIF or MODF status flag is set
0 = SPI interrupts are inhibited

SPE — SPI System Enable
0 = SPI internal hardware is initialized and SPI system is in a low-power disabled state.
1 = PS[4:7] are dedicated to the SPI function
When MODF is set, SPE always reads zero. SP0CR1 must be written as part of a mode fault recovery sequence.

SWOM — Port S Wired-OR Mode
Controls not only SPI output pins but also the general-purpose output pins (PS[4:7]) which are not used by SPI.
0 = SPI and/or PS[4:7] output buffers operate normally
1 = SPI and/or PS[4:7] output buffers behave as open-drain outputs

MSTR — SPI Master/Slave Mode Select
0 = Slave mode
1 = Master mode

CPOL, CPHA — SPI Clock Polarity, Clock Phase
These two bits are used to specify the clock format to be used in SPI operations. When the clock polarity bit is cleared and data is not being transferred, the SCK pin of the master device is low. When CPOL is set, SCK idles high. See **Figure 22** and **Figure 23**.

SSOE — Slave Select Output Enable
The $\overline{SS}$ output feature is enabled only in the master mode by asserting the SSOE and DDS7.

LSBF — SPI LSB First enable
0 = Data is transferred most significant bit first
1 = Data is transferred least significant bit first
Normally data is transferred most significant bit first.This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register will always have MSB in bit 7.

SP0CR2 — SPI Control Register 2 **$00D1**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	SSWAI	SPC0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

SSWAI — SSI Stop in Wait Mode
0 = SSI clock operate normally
1 = Halt SSI clock generation when in Wait mode

SPC0 — Serial Pin Control 0
This bit decides serial pin configurations with MSTR control bit.

Table 31

Pin Mode		SPC0[1]	MSTR	MISO[2]	MOSI[3]	SCK[4]	SS[5]
#1	Normal	0	0	Slave Out	Slave In	SCK In	SS In
#2			1	Master In	Master Out	SCK Out	SS I/O
#3	Bidirectional	1	0	Slave I/O	GPI/O	SCK In	SS In
#4			1	GPI/O	Master I/O	SCK Out	SS I/O

NOTES:
1. The serial pin control 0 bit enables bidirectional configurations.
2. Slave output is enabled if DDS4 = 1, SS = 0 and MSTR = 0. (#1, #3)
3. Master output is enabled if DDS5 = 1 and MSTR = 1. (#2, #4)
4. SCK output is enabled if DDS6 = 1 and MSTR = 1. (#2, #4)
5. SS output is enabled if DDS7 = 1, SSOE = 1 and MSTR = 1. (#2, #4)

SP0BR — SPI Baud Rate Register **$00D2**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	SPR2	SPR1	SPR0
RESET:	0	0	0	0	0	0	0	0

Read anytime. Write anytime.
At reset, E Clock divided by 2 is selected.

SPR[2:0] — SPI Clock (SCK) Rate Select Bits
These bits are used to specify the SPI clock rate.

Table 32 SPI Clock Rate Selection

SPR2	SPR1	SPR0	E Clock Divisor	Frequency at E Clock = 4 MHz	Frequency at E Clock = 8 MHz
0	0	0	2	2.0 MHz	4.0 MHz
0	0	1	4	1.0 MHz	2.0 MHz
0	1	0	8	500 KHz	1.0 MHz
0	1	1	16	250 KHz	500 KHz
1	0	0	32	125 KHz	250 KHz
1	0	1	64	62.5 KHz	125 KHz
1	1	0	128	31.3 KHz	62.5 KHz
1	1	1	256	15.6 KHz	31.3 KHz

SP0SR — SPI Status Register **$00D3**

	Bit 7	6	5	4	3	2	1	Bit 0
	SPIF	WCOL	0	MODF	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

Read anytime. Write has no meaning or effect.

SPIF — SPI Interrupt Request
SPIF is set after the eighth SCK cycle in a data transfer and it is cleared by reading the SP0SR register (with SPIF set) followed by an access (read or write) to the SPI data register.

WCOL — Write Collision Status Flag

The MCU write is disabled to avoid writing over the data being transferred. No interrupt is generated because the error status flag can be read upon completion of the transfer that was in progress at the time of the error. Automatically cleared by a read of the SP0SR (with WCOL set) followed by an access (read or write) to the SP0DR register.

0 = No write collision

1 = Indicates that a serial transfer was in progress when the MCU tried to write new data into the SP0DR data register.

MODF — SPI Mode Error Interrupt Status Flag

This bit is set automatically by SPI hardware if the MSTR control bit is set and the slave select input pin becomes zero. This condition is not permitted in normal operation. In the case where DDRS bit 7 is set, the PS7 pin is a general-purpose output pin or $\overline{SS}$ output pin rather than being dedicated as the $\overline{SS}$ input for the SPI system. In this special case the mode fault function is inhibited and MODF remains cleared. This flag is automatically cleared by a read of the SP0SR (with MODF set) followed by a write to the SP0CR1 register.

SP0DR — SPI Data Register **$00D5**

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

Read anytime (normally only after SPIF flag set). Write anytime (see WCOL write collision flag).
Reset does not affect this address.

This 8-bit register is both the input and output register for SPI data. Reads of this register are double buffered but writes cause data to written directly into the serial shifter. In the SPI system the 8-bit data register in the master and the 8-bit data register in the slave are linked by the MOSI and MISO wires to form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master so the data is effectively exchanged between the master and the slave. Note that some slave devices are very simple and either accept data from the master without returning data to the master or pass data to the master without requiring data from the master.

13.4 Port S

In all modes, port S bits PS[7:0] can be used for either general-purpose I/O, or with the SCI and SPI subsystems. During reset, port S pins are configured as high-impedance inputs (DDRS is cleared).

PORTS — Port S Data Register **$00D6**

	Bit 7	6	5	4	3	2	1	Bit 0
	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
Pin Function	$\overline{SS}$ / $\overline{CS}$	SCK	MOSI / MOMI	MISO / SISO	I/O	I/O	TXD0	RXD0

PORTS can be read anytime. When configured as an input, a read will return the pin level. When configured as output, a read will return the latched output data. Writes do not change pin state when pin configured for SPI or SCI output.

After reset all bits are configured as general-purpose inputs.

Port S shares function with the on-chip serial systems (SPI0 and SCI0).

DDRS — Data Direction Register for Port S **$00D7**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDS7	DDS6	DDS5	DDS4	DDS3	DDS2	DDS1	DDS0
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.
After reset, all general-purpose I/O are configured for input only.
0 = Configure the corresponding I/O pin for input only
1 = Configure the corresponding I/O pin for output

DDS0 — Data Direction for Port S, Bit 0
If the SCI receiver is configured for two-wire SCI operation, corresponding port S pins will be input regardless of the state of these bits.

DDS1 — Data Direction for Port S, Bit 1
If the SCI transmitter is configured for two-wire SCI operation, corresponding port S pins will be output regardless of the state of these bits.

DDS[2:3] — Data Direction for Port S Bit 2 and Bit 3
These bits are for general purpose I/O only.

DDS[6:4] — Data Direction for Port S Bits 6 through 4
If the SPI is enabled and expects the corresponding port S pin to be an input, it will be an input regardless of the state of the DDRS bit. If the SPI is enabled and expects the bit to be an output, it will be an output only if the DDRS bit is set.

DDS7 — Data Direction for Port S Bit 7
In SPI slave mode, DDS7 has no meaning or effect; the PS7 pin is dedicated as the $\overline{SS}$ input. In SPI master mode, DDS7 determines whether PS7 is an error detect input to the SPI or a general-purpose or slave select output line.

PURDS — Pullup and Reduced Drive for Port S **$00DB**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	RDPS2	RDPS1	RDPS0	0	PUPS2	PUPS1	PUPS0
RESET:	0	0	0	0	0	1	1	1

Read or write anytime.

RDPS2 — Reduce Drive of PS[7:4]
0 = Port S output drivers for bits 7 through 4 operate normally.
1 = Port S output pins for bits 7 through 4 have reduced drive capability for lower power and less noise.

RDPS1 — Reduce Drive of PS[3:2]
0 = Port S output drivers for bits 3 and 2 operate normally.
1 = Port S output pins for bits 3 and 2 have reduced drive capability for lower power and less noise.

RDPS0 — Reduce Drive of PS[1:0]
0 = Port S output drivers for bits 1 and 0 operate normally.
1 = Port S output pins for bits 1 and 0 have reduced drive capability for lower power and less noise.

PUPS2 — Pull-Up Port S Enable PS[7:4]

0 = No internal pull-ups on port S bits 7 through 4.

1 = Port S input pins for bits 7 through 4 have an active pull-up device. If a pin is programmed as output, the pull-up device becomes inactive.

PUPS1 — Pull-Up Port S Enable PS[3:2]

0 = No internal pull-ups on port S bits 3 and 2.

1 = Port S input pins for bits 3 and 2 have an active pull-up device. If a pin is programmed as output, the pull-up device becomes inactive.

PUPS0 — Pull-Up Port S Enable PS[1:0]

0 = No internal pull-ups on port S bits 1 and 0.

1 = Port S input pins for bits 1 and 0 have an active pull-up device. If a pin is programmed as output, the pull-up device becomes inactive.

14 MSCAN12 Controller

The MSCAN12 is the specific implementation of the Motorola Scalable CAN (MSCAN) concept targeted for the Motorola M68HC12 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN12 utilizes an advanced buffer arrangement resulting in a predictable real-time behaviour and simplifies the application software.

14.1 Features

The basic features of the MSCAN12 are as follows:

- Modular Architecture
- Implementation of the CAN protocol - Version 2.0A/B
  - Standard and extended data frames.
  - 0 - 8 bytes data length.
  - Programmable bit rate up to 1 Mbps[1].
- Support for Remote Frames.
- Double buffered receive storage scheme.
- Triple buffered transmit storage scheme with internal priorisation using a “local priority” concept.
- Flexible maskable identifier filter supports alternatively two full size extended identifier filters or four 16-bit filters or eight 8-bit filters.
- Programmable wake-up functionality with integrated low-pass filter.
- Programmable Loop-Back mode supports self-test operation.
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (Warning, Error Passive, Bus-Off).
- Programmable MSCAN12 clock source either CPU bus clock or crystal oscillator output.
- Programmable link to on-chip Timer Interface Module (TIM) for time-stamping and network synchronization.
- Low power Sleep Mode.

1. Depending on the actual bit timing and the clock jitter of the PLL.

14.2 External Pins

The MSCAN12 uses 2 external pins, 1 input (RxCAN) and 1 output (TxCAN). The TxCAN output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state.

RxCAN is on bit 0 of Port CAN, TxCAN is on bit 1. The remaining six pins of Port CAN are controlled by registers in the MSCAN12 address space (see **PCTLCAN — MSCAN12 Port CAN Control Register $xx3D** through **DDRCAN — MSCAN12 Port CAN Data Direction Register $xx3F**).

A typical CAN system with MSCAN12 is shown in **Figure 25** below.

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection, against defected CAN or defected stations.

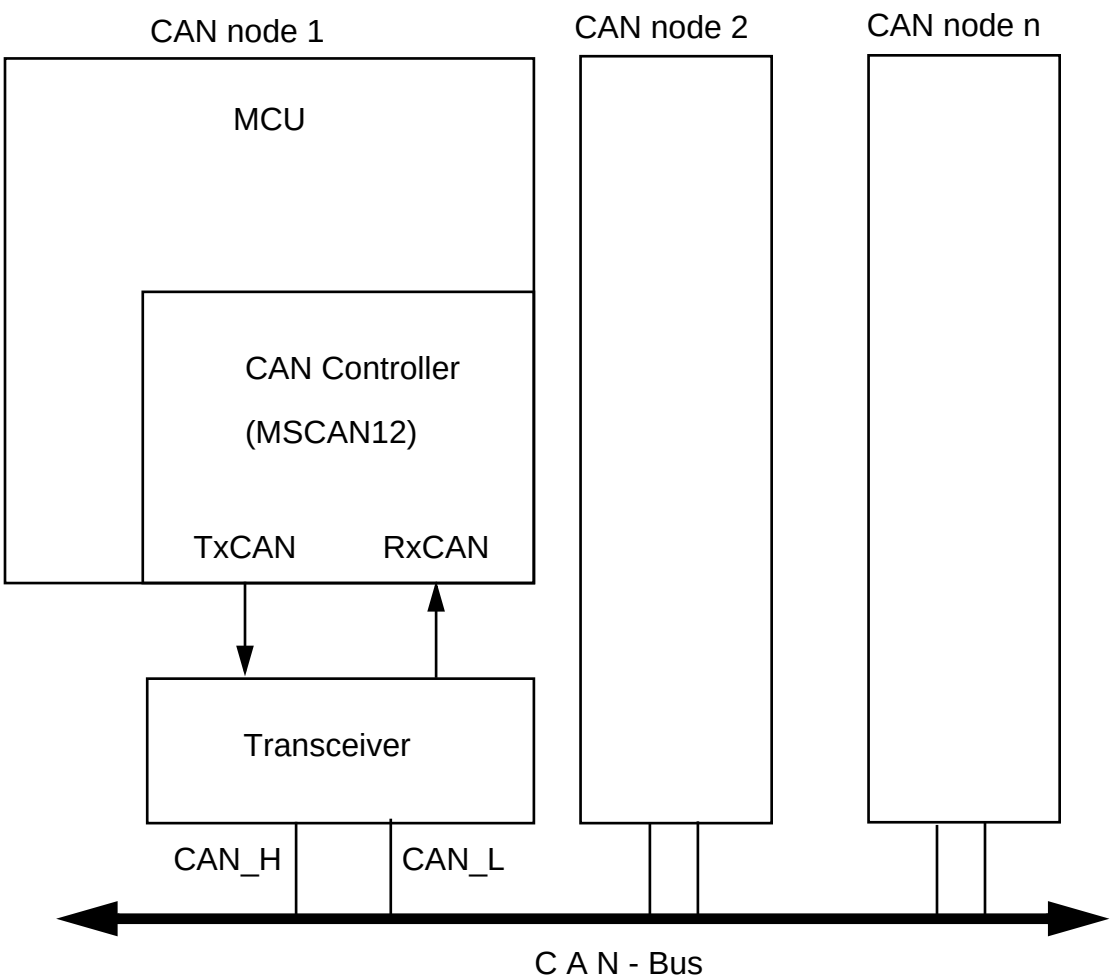


Figure 25 The CAN System

14.3 Message Storage

MSCAN12 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

14.3.1 Background

Modern application layer software is built upon two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.

2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behaviour can not be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the Inter-Frame Sequence (IFS) in order to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message sending and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of above requirements under all circumstances. The MSCAN12 has three transmit buffers.

The second requirement calls for some sort of internal priorisation which the MSCAN12 implements with the “local priority” concept described below.

14.3.2 Receive Structures

The received messages are stored in a two stage input FIFO. The two message buffers are mapped into a single memory area (see **Figure 26**). While the background receive buffer (RxBG) is exclusively associated to the MSCAN12, the foreground receive buffer (RxFG) is addressable by the CPU12. This scheme simplifies the handler software as only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended) and the data contents (for details see **14.11 Programmer’s Model of Message Storage**).

The Receiver Full flag (RXF) in the MSCAN12 Receiver Flag Register (CRFLG) (see **CRFLG — MSCAN12 Receiver Flag Register $xx04**) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier this flag is set.

After the MSCAN12 successfully received a message into the background buffer and if the message passes the filter (for details see **14.4 Identifier Acceptance Filter**) it copies the content of RxBG into RxFG[2], sets the RXF flag, and emits a receive interrupt to the CPU[3]. A new message - which may follow immediately after the IFS field of the CAN frame - will be received into RxBG. The over-writing of the background buffer is independent of the identifier filter function.

2. Only if the RXF flag is not set.
3. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

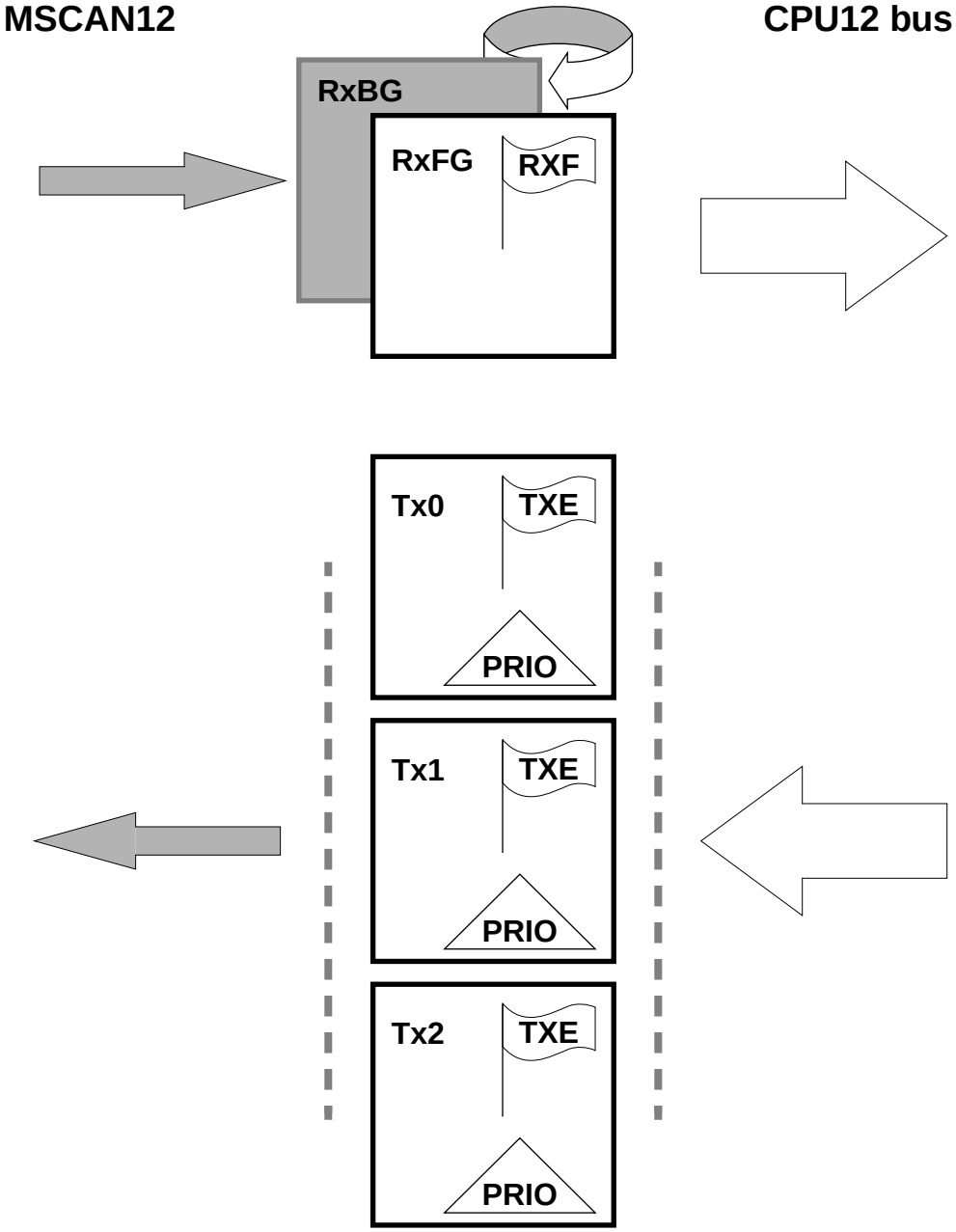


Figure 26 User Model for Message Buffer Organization

The user's receive handler has to read the received message from RxFG and to reset the RXF flag in order to acknowledge the interrupt and to release the foreground buffer.

An overrun conditions occurs when both, the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and a further message is being received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. While in the overrun situation, the MSCAN12 will stay synchronized to the CAN bus and is able to transmit messages but will discard all incoming messages.

NOTE

MSCAN12 will receive its own messages into the background receive buffer RxBG but will NOT overwrite RxFG and will NOT emit a receive interrupt nor will it acknowledge (ACK) its own messages on the CAN bus. The exception to this rule is that when in loop-back mode MSCAN12 will treat its own messages exactly like all other incoming messages.

14.3.3 Transmit Structures

The MSCAN12 has a triple transmit buffer scheme in order to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in **Figure 26**.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers (see **14.11 Programmer's Model of Message Storage**). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit so called "Local Priority" field (PRIO) (see **TBPR — Transmit Buffer Priority Registers $xxbD**).

In order to transmit a message, the CPU12 has to identify an available transmit buffer which is indicated by a set Transmit Buffer Empty (TXE) Flag in the MSCAN12 Transmitter Flag Register (CTFLG) (see **CTFLG — MSCAN12 Transmitter Flag Register $xx06**).

The CPU12 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged as being ready for transmission by clearing the TXE flag.

The MSCAN12 will then schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt will be emitted[4] when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN12 uses the "local priority" setting of the three buffers for priorisation. For this purpose every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority.

The internal scheduling process takes places whenever the MSCAN12 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission can not be aborted, the user has to request the abort by setting the corresponding Abort Request Flag (ABTRQ) in the Transmission Control Register (CTCR). The MSCAN12 will then grant the request if possible by setting the corresponding Abort Request Acknowledge (ABTAK) and the TXE flag in order to release the buffer and by emitting a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK=1) or has been sent in the meantime (ABTAK=0).

4. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

14.4 Identifier Acceptance Filter

A very flexible programmable generic identifier acceptance filter has been introduced in order to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to the full 29 bits of the identifier and to the following bits of the CAN frame: RTR, IDE, SRR. This mode implements a two filters for a full length CAN 2.0B compliant extended identifier. **Figure 27** shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to a) the 11 bits of the identifier and the RTR bit of CAN 2.0A messages or b) the 14 most significant bits of the identifier of CAN 2.0B messages. **Figure 28** shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 and 1 hits. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces filter 2 and 3 hits.

- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bit of a CAN 2.0A compliant standard identifier or of a CAN 2.0B compliant extended identifier. **Figure 29** shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 to 3 hits. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces filter 4 to 7 hits.

- Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

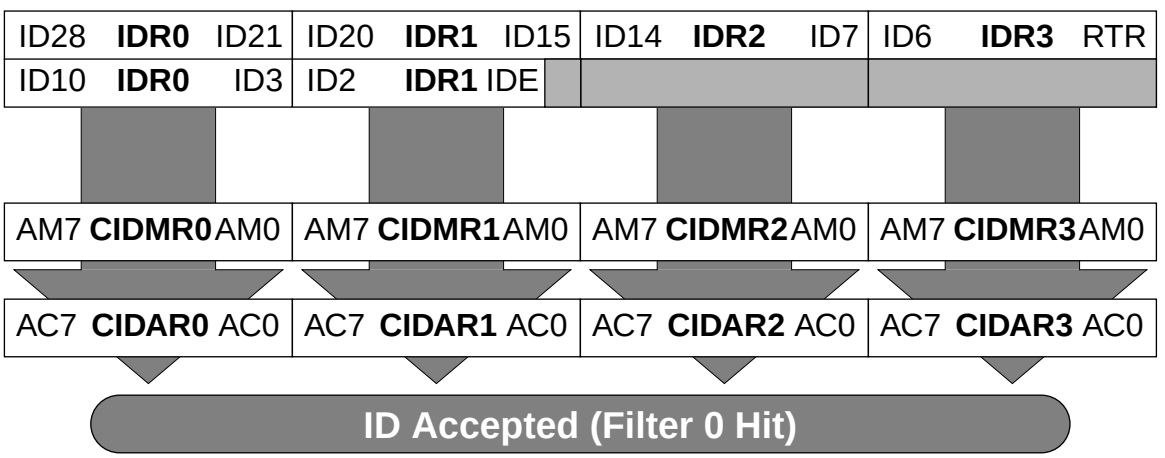


Figure 27 32-bit Maskable Identifier Acceptance Filter

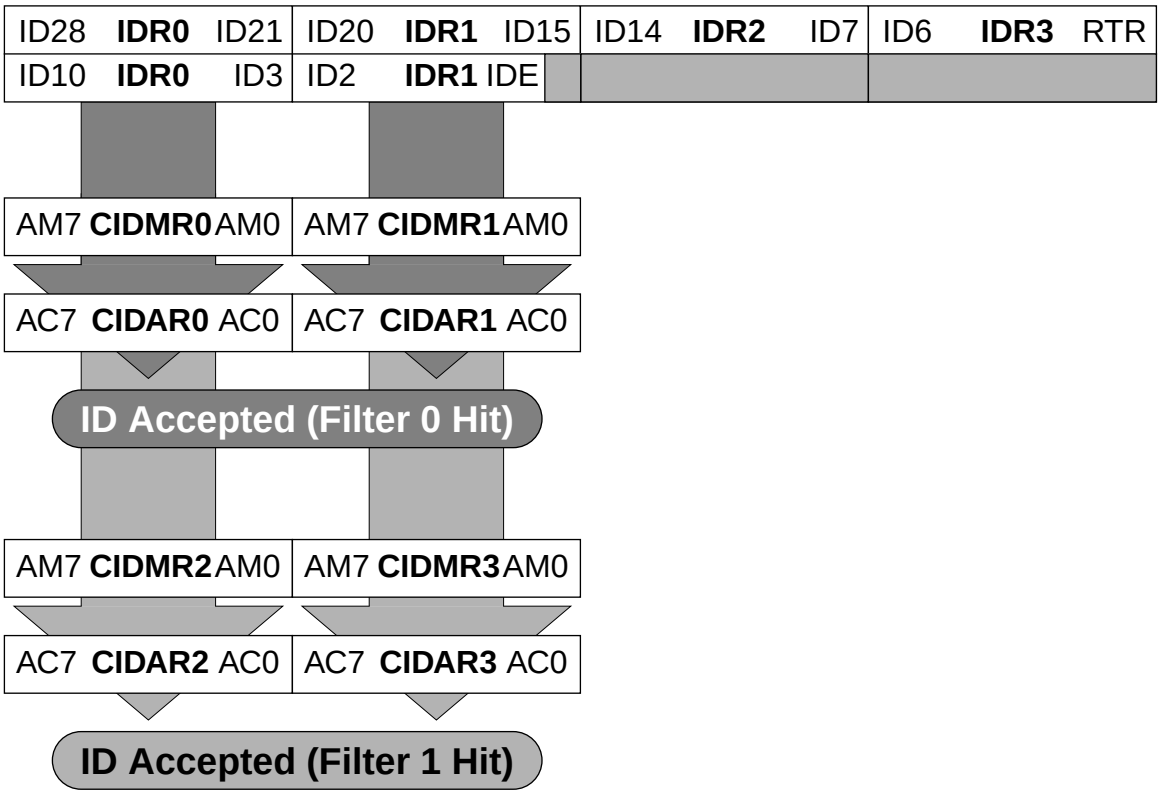


Figure 28 16-bit Maskable Acceptance Filters

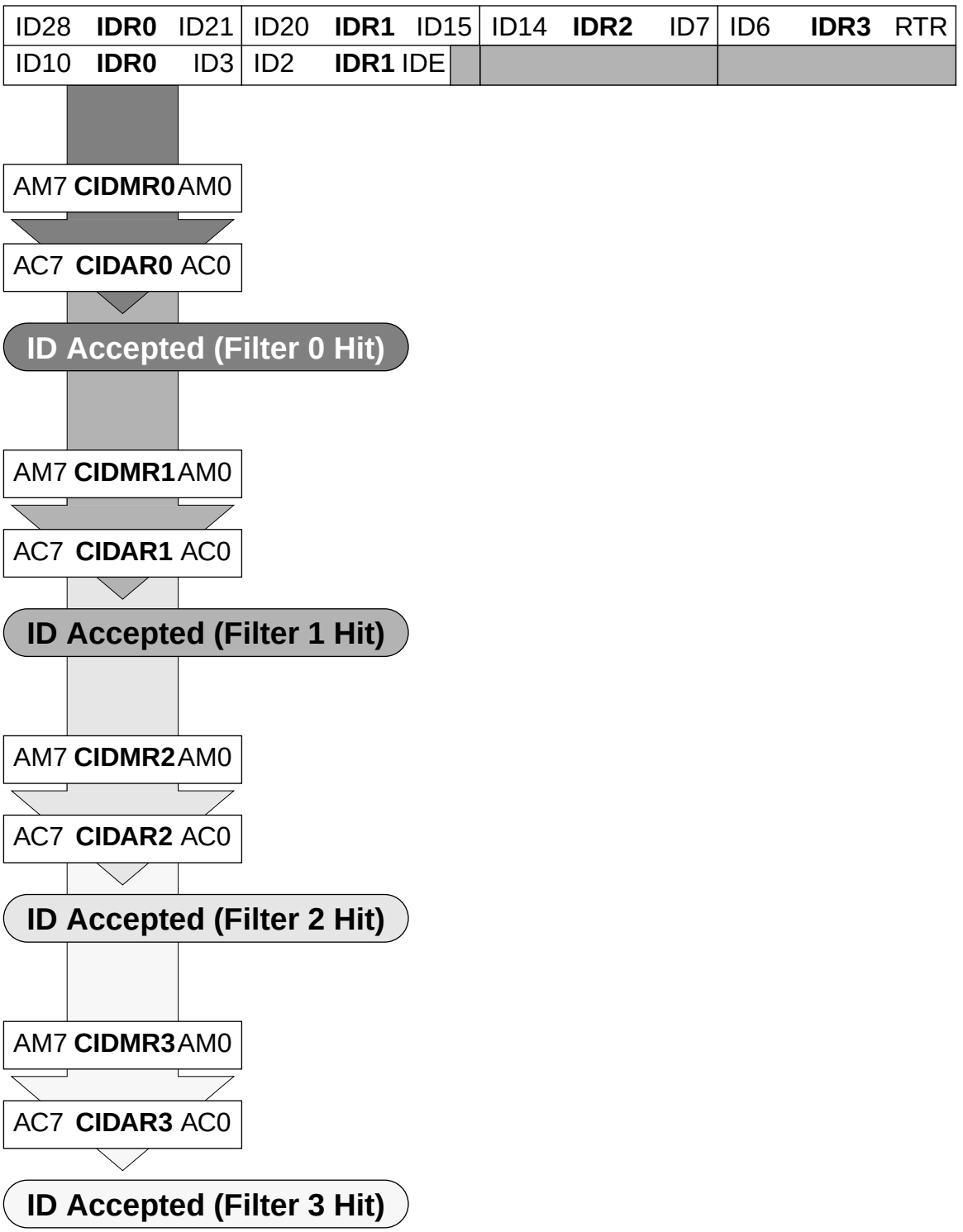


Figure 29 8-bit Maskable Acceptance Filters

The Identifier Acceptance Registers (CIDAR0-7) define the acceptable patterns of the standard or extended identifier (ID10 - ID0 or ID28 - ID0). Any of these bits can be marked ‘don’t care’ in the Identifier Mask Registers (CIDMR0-7).

A filter hit is indicated to the application software by a set RXF (Receive Buffer Full Flag, see **CRFLG — MSCAN12 Receiver Flag Register $xx04**) and three bits in the Identifier Acceptance Control Register (see **CIDAC — MSCAN12 Identifier Acceptance Control Register $xx08**). These Identifier Hit Flags (IDHIT2-0) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A hit will also cause a receiver interrupt if enabled.

14.5 Interrupts

The MSCAN12 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see **CRFLG — MSCAN12 Receiver Flag Register $xx04** to **CTCR — MSCAN12 Transmitter Control Register $xx07**).

- *Transmit Interrupt:* At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been successfully received and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wake-Up Interrupt*: An activity on the CAN bus occurred during MSCAN12 internal Sleep Mode.
- *Error Interrupt*: An overrun, error or warning condition occurred. The Receiver Flag Register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in **14.3.2 Receive Structures** has occurred.
  - *Receiver Warning*: The Receive Error Counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning*: The Transmit Error Counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive*: The Receive Error Counter has exceeded the Error Passive limit of 127 and MSCAN12 has gone to Error Passive state.
  - *Transmitter Error Passive*: The Transmit Error Counter has exceeded the Error Passive limit of 127 and MSCAN12 has gone to Error Passive state.
  - *Bus Off*: The Transmit Error Counter has exceeded 255 and MSCAN12 has gone to Bus Off state.

14.5.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN12 Receiver Flag Register (CRFLG) or the MSCAN12 Transmitter Control Register (CTCR). Interrupts are pending as long as one of the corresponding flags is set. The flags in above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a “1” to the corresponding bit position. A flag can not be cleared if the respective condition still prevails.

WARNING

Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.

14.5.2 Interrupt Vectors

The MSCAN12 supports four interrupt vectors as shown in **Table 33**. The vector addresses are dependent on the chip integration and to be defined. The relative interrupt priority is also integration dependent and to be defined.

Table 33 MSCAN12 Interrupt Vectors

Function	Source	Local Mask	Global Mask
Wake-Up	WUPIF	WUPIE	I Bit
Error Interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

14.6 Protocol Violation Protection

The MSCAN12 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters can not be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN12 can not be modified while the MSCAN12 is on-line. The SFTRES bit in CMCR0 (see **CMCR0 — MSCAN12 Module Control Register $xx00**) serves as a lock to protect the following registers:
  - MSCAN12 Module Control Register 1 (CMCR1)
  - MSCAN12 Bus Timing Register 0 and 1 (CBTR0, CBTR1)
  - MSCAN12 Identifier Acceptance Control Register (CIDAC)
  - MSCAN12 Identifier Acceptance Registers (CIDAR0-7)
  - MSCAN12 Identifier Mask Registers (CIDMR0-7)
- The TxCAN pin is forced to Recessive if the CPU goes into STOP mode.

14.7 Low Power Modes

The MSCAN12 has three modes with reduced power consumption compared to Normal Mode. In Sleep and Soft Reset Mode, power consumption is reduced by stopping all clocks except those to access the registers. In Power Down Mode, all clocks are stopped and no power is consumed.

The WAI and STOP instruction put the MCU in low power consumption stand-by modes. **Table 34** summarizes the combinations of MSCAN12 and CPU modes. A particular combination of modes is entered for the given settings of the bits CSWAI, SLPAK, and SFTRES. For all modes, an MSCAN wake-up interrupt can occur only if SLPAK=WUPIE=1. While the CPU is in Wait Mode, the MSCAN12 can be operated in Normal Mode and emit interrupts (registers can be accessed via background debug mode).

Table 34 MSCAN12 vs. CPU operating modes

MSCAN Mode	CPU Mode		
	STOP	**WAIT**	**RUN**
Power Down	CSWAI = X[1] SLPAK = X SFTRES = X	CSWAI = 1 SLPAK = X SFTRES = X	
Sleep		CSWAI = 0 SLPAK = 1 SFTRES = 0	CSWAI = X SLPAK = 1 SFTRES = 0
Soft Reset		CSWAI = 0 SLPAK = 0 SFTRES = 1	CSWAI = X SLPAK = 0 SFTRES = 1
Normal		CSWAI = 0 SLPAK = 0 SFTRES = 0	CSWAI = X SLPAK = 0 SFTRES = 0

1. 'X' means don't care.

14.7.1 MSCAN12 Sleep Mode

The CPU can request the MSCAN12 to enter this low-power mode by asserting the SLPRQ bit in the Module Configuration Register (see **Figure 30**). The time when the MSCAN12 will then enter Sleep Mode depends on its current activity:

- if it is transmitting, it will continue to transmit until there is no more message to be transmitted, and then go into Sleep Mode
- if it is receiving, it will wait for the end of this message and then go into Sleep Mode
- if it is neither transmitting nor receiving, it will immediately go into Sleep Mode

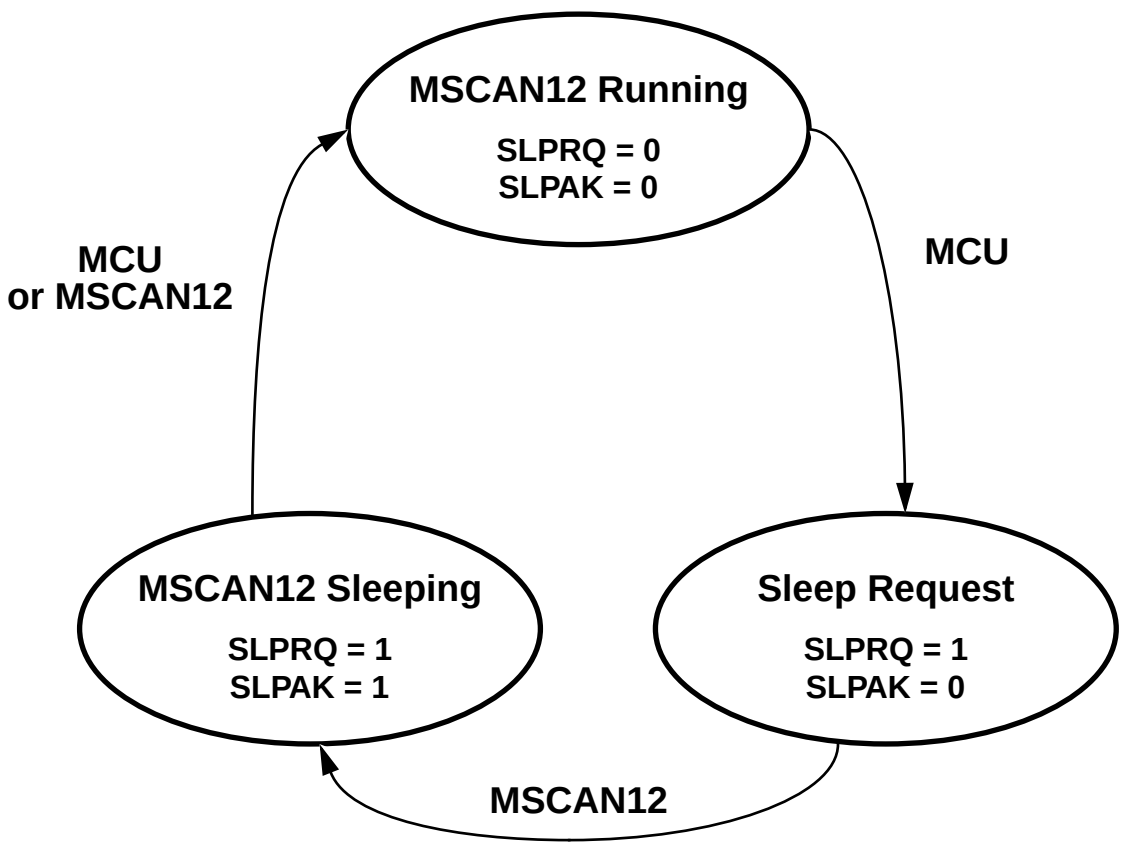


Figure 30 Sleep Request / Acknowledge Cycle

The application software must avoid to set up a transmission (by clearing one or more TXE flag(s)) and immediately request Sleep Mode (by setting SLPRQ). It will then depend on the exact sequence of operations whether the MSCAN12 will start transmitting or go into Sleep Mode directly.

During Sleep Mode, the SLPAK flag is set. The application software should use this flag as a handshake indication for the request to go into Sleep Mode. When in Sleep Mode, the MSCAN12 stops its own clocks and the TxCAN pin will stay in recessive state.

The MSCAN12 will leave Sleep Mode (wake-up) when

- bus activity occurs or
- the MCU clears the SLPRQ bit.

NOTE

The MCU can not clear the SLPRQ bit before the MSCAN12 is in Sleep Mode (SLPAK = 1).

14.7.2 MSCAN12 Soft Reset Mode

In Soft Reset Mode, the MSCAN12 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing, and the CAN message filter. See **CMCR0 — MSCAN12 Module Control Register $xx00** for a complete description of the Soft Reset Mode.

14.7.3 MSCAN12 Power Down Mode

The MSCAN12 is in Power Down Mode when

- the CPU is in Stop Mode or
- the CPU is in Wait Mode and the CSWAI bit is set (see **CMCR0 — MSCAN12 Module Control Register $xx00**).

When entering the Power Down Mode, the MSCAN12 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. The user is responsible to take care that the MSCAN12 is not active when Power Down Mode is entered. The recommended procedure is to bring the MSCAN12 into Sleep Mode before the STOP instruction - or the WAI instruction, if CSWAI is set - is executed.

To protect the CAN bus system from fatal consequences of violations to above rule, the MSCAN12 will drive the TxCAN pin into recessive state.

In Power Down Mode no registers can be accessed.

14.7.4 Programmable Wake-Up Function

The MSCAN12 can be programmed to apply a low-pass filter function to the RxCAN input line while in Sleep Mode (see control bit WUPM in the Module Control Register, **CMCR0 — MSCAN12 Module Control Register $xx00**). This feature can be used to protect the MSCAN12 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

14.8 Timer Link

The MSCAN12 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN12 receiver engine receives also the frames being sent by itself, a timer signal will also be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip Timer Interface Module (TIM). Under the control of the Timer Link Enable (TLNKEN) bit in the CMCR0 will this signal be connected to the Timer n Channel m input[5].

After Timer n has been programmed to capture rising edge events it can be used to generate 16-bit time stamps which can be stored under software control with the received message.

14.9 Clock System

Figure 31 shows the structure of the MSCAN12 clock generation circuitry. With this flexible clocking scheme the MSCAN12 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

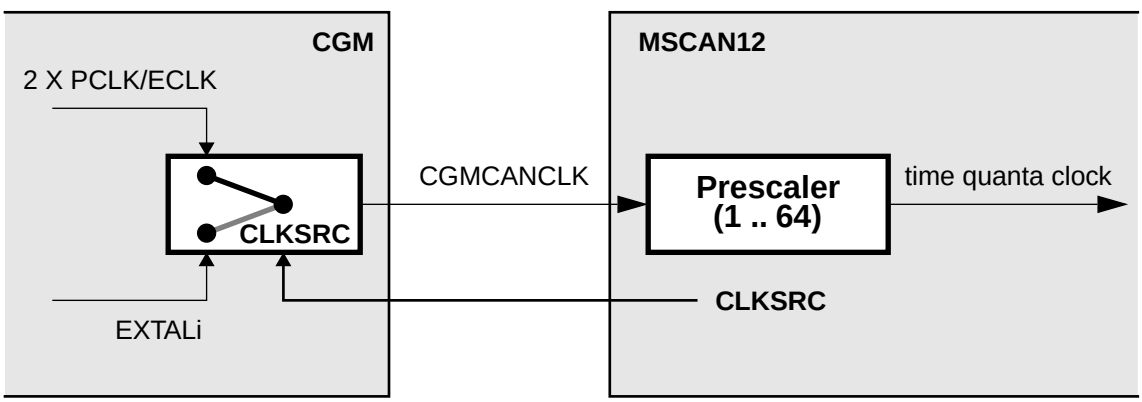


Figure 31 Clocking Scheme

The Clock Source bit (CLKSRC) in the MSCAN12 Module Control Register (CMCR1) (see **CBTR0 — MSCAN12 Bus Timing Register 0 $xx02**) defines whether the MSCAN12 is connected to the output of the crystal oscillator (EXTALi) or to a clock twice as fast as the system clock (ECLK).

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 50% duty cycle of the clock is required.

For microcontrollers without the CGM module, CGMCANCLK is driven from the crystal oscillator (EXTALi).

5. The timer channel being used for the timer link is PT0.

A programmable prescaler is used to generate out of MSCANCLK the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN12. A bit time is subdivided into three segments[6]:

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

The Synchronization Jump Width can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

Above parameters can be set by programming the Bus Timing Registers (CBTR0-1, see **CBTR0 — MSCAN12 Bus Timing Register 0 $xx02** and **CBTR1 — MSCAN12 Bus Timing Register 1 $xx03**).

It is the user's responsibility to make sure that his bit time settings are in compliance with the CAN standard. **Figure 35** gives an overview on the CAN conforming segment settings and the related parameter values.

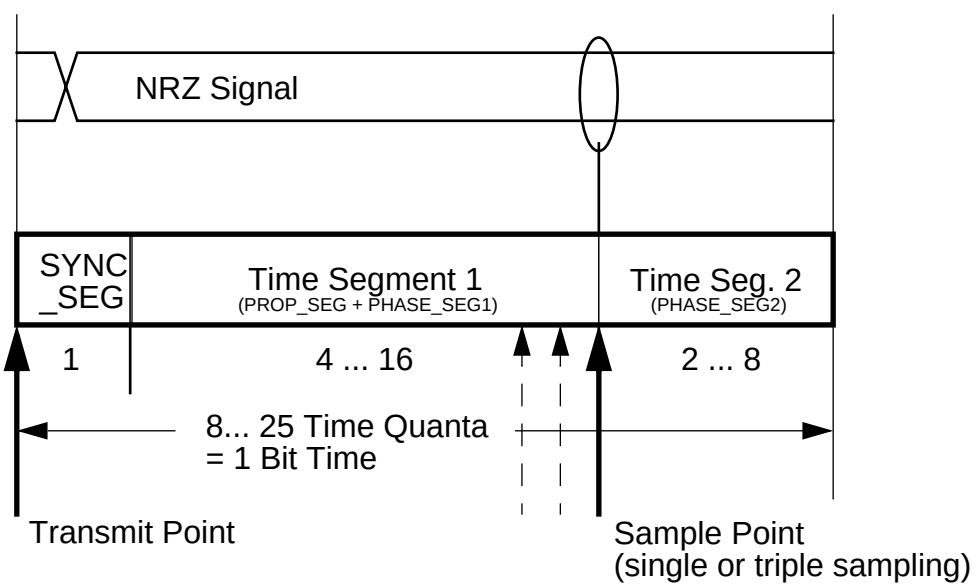


Figure 32 Segments within the Bit Time

6. For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.

Table 35 CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronize. Jump Width	SJW
5.. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

14.10 Memory Map

The MSCAN12 occupies 128 Byte in the CPU12 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128. The background receive buffer can only be read in test mode.

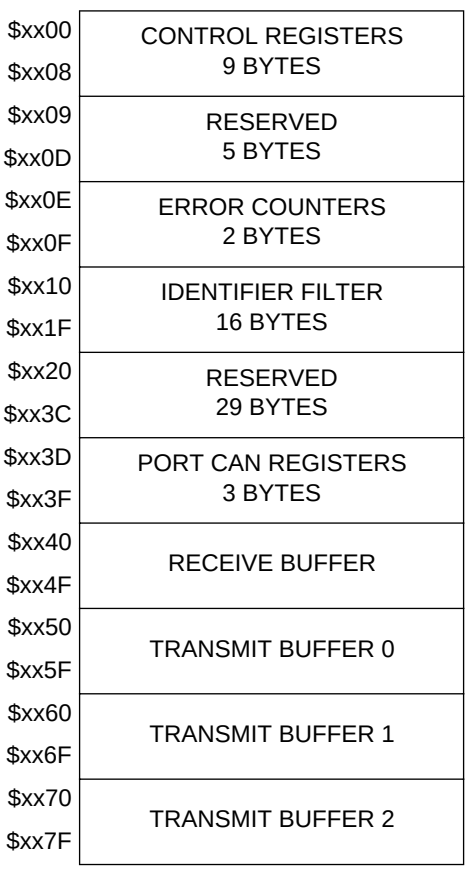


Figure 33 MSCAN12 Memory Map

NOTE

Due to design requirements, the absolute addresses and bit locations may change with later releases of the specification.

14.11 Programmer's Model of Message Storage

The following section details the organisation of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification the receive and transmit message buffers have the same outline. Each message buffer allocates 16 byte in the memory map containing a 13 byte data structure. An additional Transmit Buffer Priority Register (TBPR) is defined for the transmit buffers.

Addr	Register Name
xxb0	Identifier Register 0
xxb1	Identifier Register 1
xxb2	Identifier Register 2
xxb3	Identifier Register 3
xxb4	Data Segment Register 0
xxb5	Data Segment Register 1
xxb6	Data Segment Register 2
xxb7	Data Segment Register 3
xxb8	Data Segment Register 4
xxb9	Data Segment Register 5
xxbA	Data Segment Register 6
xxbB	Data Segment Register 7
xxbC	Data Length Register
xxbD	Transmit Buffer Priority Register[1]
xxbE	unused
xxbF	unused

1. Not applicable for receive buffers

Figure 34 Message Buffer Organisation

14.11.1 Message Buffer Outline

Figure 35 shows the common 13 byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in **Figure 36**. All bits of the 13 byte data structure are undefined out of reset.

Figure 35 Receive / Transmit Message Buffer Extended Identifier

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xxb0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	IDR0
$xxb1	ID20	ID19	ID18	SRR (1)	IDE (1)	ID17	ID16	ID15	IDR1
$xxb2	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	IDR2
$xxb3	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR	IDR3
$xxb4	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR0
$xxb5	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR1
$xxb6	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR2
$xxb7	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR3
$xxb8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR4
$xxb9	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR5
$xxbA	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR6
$xxbB	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DSR7
$xxbC					DLC3	DLC2	DLC1	DLC0	DLR

Figure 36 Standard Identifier Mapping

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xxb0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	IDR0
$xxb1	ID2	ID1	ID0	RTR	IDE(0)				IDR1
$xxb2									IDR2
$xxb3									IDR3

14.11.2 Identifier Registers (IDRn)

The identifiers consist of either 11 bits (ID10 – ID0) for the standard, or 29 bits (ID28 - ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR - Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer the flag indicates to the MSCAN12 what type of identifier to send.

1 = Extended format (29 bit)
0 = Standard format (11 bit)

RTR — Remote transmission request

This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In case of a receive buffer it indicates the status of the received frame and allows to support the transmission of an answering frame in software. In case of a transmit buffer this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

14.11.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

DLC3 – DLC0 — Data length code bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. **Table 36** shows the effect of setting the DLC bits.

Table 36 Data length codes

Data length code				Data byte count
DLC3	**DLC2**	**DLC1**	**DLC0**	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

14.11.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or being received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

TBPR — Transmit Buffer Priority Registers **$xxbD**

	Bit 7	6	5	4	3	2	1	Bit 0
	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
RESET:	U	U	U	U	U	U	U	U

PRIO7 - PRIO0— Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal priorisation process of the MSCAN12 and is defined to be highest for the smallest binary number. The MSCAN12 implements the following internal priorisation mechanism:

- All transmission buffers with a cleared TXE flag participate in the priorisation right before the SOF (Start of Frame) is sent.
- The transmission buffer with the lowest local priority field wins the priorisation.
- In case of more than one buffer having the same lowest priority the message buffer with the lower index number wins.

WARNING

To ensure data integrity, no registers of the transmit buffers shall be written while the associated TXE flag is cleared. Similarly, to ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.

14.12 Programmer's Model of Control Registers

14.12.1 Overview

The programmer's model has been laid out for maximum simplicity and efficiency. The following figure gives an overview on the control register block of the MSCAN12:

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xx00	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	CMCR0
$xx01	0	0	0	0	0	LOOPB	WUPM	CLKSRC	CMCR1
$xx02	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	CBTR0
$xx03	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	CBTR1
$xx04	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	CRFLG
$xx05	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE	CRIER
$xx06	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	CTFLG
$xx07	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	CTCR
$xx08	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	CIDAC
$xx09-$xx0D									reserved
$xx0E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	CRXERR
$xx0F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	CTXERR
$xx10	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR0
$xx11	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR1
$xx12	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR2
$xx13	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR3
$xx14	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR0
$xx15	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR1
$xx16	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR2
$xx17	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR3
$xx18	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR4
$xx19	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR5
$xx1A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR6
$xx1B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR7
$xx1C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR4
$xx1D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR5
$xx1E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR6
$xx1F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR7
$xx20-$xx3C									reserved
$xx3D	0	0	0	0	0	0	PUECAN	RDRCAN	PCTLCAN
$xx3E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCan	RxCan	PORTCAN
$xx3F	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0	DDRCAN

Figure 37 MSCAN12 Control Register Structure

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
RESET:	0	0	1	0	0	0	0	1

CSWAI — CAN Stops in Wait Mode

1 = The module ceases to be clocked during WAIT mode.
0 = The module is not affected during WAIT mode.

SYNCH — Synchronized Status

This bit indicates whether the MSCAN12 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN12 is synchronized to the CAN bus
0 = MSCAN12 is not synchronized to the CAN bus

TLNKEN - Timer Enable

This flag is used to establish a link between the MSCAN12 and the on-chip timer (see **14.8 Timer Link**).

1 = The MSCAN12 timer signal output is connected to the timer input.
0 = The port is connected to the timer input.

SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN12 is in module internal Sleep Mode. It shall be used as a handshake for the Sleep Mode request (see **14.7.1 MSCAN12 Sleep Mode**).

1 = Sleep – The MSCAN12 is in Sleep Mode.
0 = Wake-up – The MSCAN12 is not in Sleep Mode.

SLPRQ — Sleep request, go to Sleep Mode

This flag allows to request the MSCAN12 to go into an internal power-saving mode (see **14.7.1 MSCAN12 Sleep Mode**).

1 = Sleep request – The MSCAN12 will go into Sleep Mode.
0 = Wake-up – The MSCAN12 will function normally.

SFTRES— Soft Reset

When this bit is set by the CPU, the MSCAN12 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers will go into and stay in the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0-7, CIDMR0-7 can only be written by the CPU when the MSCAN12 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN12 will try to synchronize to the CAN bus: If the MSCAN12 is not in bus-off state it will be synchronized after 11 recessive bits on the bus; if the MSCAN12 is in bus-off state it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 = MSCAN12 in soft reset state.
0 = Normal operation

CMCR1 — MSCAN12 Module Control Register **$xx01**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	LOOPB	WUPM	CLKSRC
RESET	0	0	0	0	0	0	0	0

LOOPB - Loop Back Self Test Mode

When this bit is set the MSCAN12 performs an internal loop back which can be used for self test operation: the bit stream output of the transmitter is fed back to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (1). Note that in this state the MSCAN12 ignores the ACK bit to insure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

1 = Activate loop back self test mode
0 = Normal operation

WUPM - Wake-Up Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN12 from spurious wake-ups (see **14.7.4 Programmable Wake-Up Function**).

1 = MSCAN12 will wake up the CPU only in case of dominant pulse on the bus which has a length of at least approximately T_{wup}.
0 = MSCAN12 will wake up the CPU after any recessive to dominant edge on the CAN bus.

CLKSRC - MSCAN12 Clock Source

This flag defines which clock source the MSCAN12 module is driven from (only for system with CGM module; see **14.9 Clock System**, **Figure 31**).

1 = The MSCAN12 clock source is twice the frequency of ECLK.
0 = The MSCAN12 clock source is EXTALi.

NOTE

The CMCR1 register can only be written if the SFTRES bit in CMCR0 is set.

CBTR0 — MSCAN12 Bus Timing Register 0 **$xx02**

	Bit 7	6	5	4	3	2	1	Bit 0
	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
RESET	0	0	0	0	0	0	0	0

SJW1, SJW0 — Synchronization Jump Width

The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see **Table 37**).

Table 37 Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

BRP5 – BRP0 — Baud Rate Prescaler

These bits determine the time quanta (Tq) clock, which is used to build up the individual bit timing, according to **Table 38**.

Table 38 Baud rate prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	1	1	64

NOTE

The CBTR0 register can only be written if the SFTRES bit in CMCR0 is set.

CBTR1 — MSCAN12 Bus Timing Register 1 **$xx03**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
RESET:	0	0	0	0	0	0	0	0

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. If set three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit.
0 = One sample per bit.

TSEG22 – TSEG10 — Time Segment

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

Table 39 Time segment syntax

SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit point	A node in transmit mode will transfer a new value to the CAN bus at this point.
Sample point	A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample.

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in **Table 40**.

Table 40 Time segment values

TSEG 13	TSEG 12	TSEG 11	TSEG 10	Time segment 1
0	0	0	0	1 Tq clock cycle
0	0	0	1	2 Tq clock cycles
0	0	1	0	3 Tq clock cycles
0	0	1	1	4 Tq clock cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 Tq clock cycles

TSEG 22	TSEG 21	TSEG 20	Time segment 2
0	0	0	1 Tq clock cycle
0	0	1	2 Tq clock cycles
.	.	.	.
.	.	.	.
1	1	1	8 Tq clock cycles

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown above).

NOTE

The CBTR1 register can only be written if the SFTRES bit in CMCR0 is set

CRFLG — MSCAN12 Receiver Flag Register **$xx04**

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can only be cleared when the condition which caused the setting is no more valid. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

	Bit 7	6	5	4	3	2	1	Bit 0
	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
RESET:	0	0	0	0	0	0	0	0

WUPIF — Wake-up Interrupt Flag

If the MSCAN12 detects bus activity whilst it is in Sleep Mode, it clears the SLPAK bit in the CMCR0 register; the WUPIF bit will then be set. If not masked, a Wake-Up interrupt is pending while this flag is set.

1 = MSCAN12 has detected activity on the bus and requested wake-up.

0 = No wake-up activity has been observed while in Sleep Mode.

RWRNIF — Receiver Warning Interrupt Flag

This bit will be set when the MSCAN12 goes into warning status due to the Receive Error counter (REC) being in the range of 96 to 127 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set[7]. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN12 went into receiver warning status.
0 = No receiver warning status has been reached.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the MSCAN12 goes into warning status due to the Transmit Error counter (TEC) being in the range of 96 to 127 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set[8]. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN12 went into transmitter warning status.
0 = No transmitter warning status has been reached.

RERRIF — Receiver Error Passive Interrupt Flag

This bit will be set when the MSCAN12 goes into error passive status due to the Receive Error counter exceeding127 and the Bus-Off interrupt flag is not set[9]. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN12 went into receiver error passive status.
0 = No receiver error passive status has been reached.

TERRIF — Transmitter Error Passive Interrupt Flag

This bit will be set when the MSCAN12 goes into error passive status due to the Transmit Error counter exceeding 127 and the Bus-Off interrupt flag is not set[10]. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN12 went into transmitter error passive status.
0 = No transmitter error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This bit will be set when the MSCAN12 goes into bus-off status, due to the Transmit Error counter exceeded 255. It cannot be cleared before the MSCAN12 has monitored 128 times 11 consecutive ‘recessive’ bits on the bus. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN12 went into bus-off status.
0 = No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This bit will be set when a data overrun condition occurs. If not masked, an Error interrupt is pending while this flag is set.

1 = A data overrun has been detected.
0 = No data overrun has occurred.

RXF — Receive Buffer Full

The RXF flag is set by the MSCAN12 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be handshaked to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a Receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.
0 = The receive buffer is released (not full).

7. RWRNIF = (96 ≤ REC < 128) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
8. TWRNIF = (96 ≤ TEC < 128) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
9. RERRIF = (128 ≤ REC ≤ 255) & $\overline{\text{BOFFIF}}$
10. TERRIF = (128 ≤ TEC ≤ 255) & BOFFIF: TERRIF is set at the end of the Bus-Off period (due to counting 128* 11 consecutive recessive bits on the TEC). Thus TERRIF should be cleared together with BOFFIF.

NOTE

The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

CRIER — MSCAN12 Receiver Interrupt Enable Register **$xx05**

	Bit 7	6	5	4	3	2	1	Bit 0
	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
RESET:	0	0	0	0	0	0	0	0

WUPIE — Wake-up Interrupt Enable
- 1 = A wake-up event will result in a wake-up interrupt.
- 0 = No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable
- 1 = A receiver warning status event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable
- 1 = A transmitter warning status event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable
- 1 = A receiver error passive status event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable
- 1 = A transmitter error passive status event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

BOFFIE — Bus-Off Interrupt Enable
- 1 = A bus-off event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

OVRIE — Overrun Interrupt Enable
- 1 = An overrun event will result in an error interrupt.
- 0 = No interrupt will be generated from this event.

RXFIE — Receiver Full Interrupt Enable
- 1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
- 0 = No interrupt will be generated from this event.

NOTE

The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CTCR register. A hard or soft reset will clear the register.

	Bit 7	6	5	4	3	2	1	Bit 0
	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
RESET:	0	0	0	0	0	1	1	1

ABTAK2 - ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent in the meantime. The flag is reset implicitly whenever the associated TXE flag is set to 0.

1 = The message has been aborted.
0 = The massage has not been aborted, thus has been sent out.

TXE2 - TXE0 —Transmitter Buffer Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN12 will set the flag after the message has been sent successfully. The flag will also be set by the MSCAN12 when the transmission request was successfully aborted due to a pending abort request (**CTCR — MSCAN12 Transmitter Control Register $xx07**). If not masked, a Transmit interrupt is pending while this flag is set.

A reset of this flag will also reset the Abort Acknowledge (ABTAK, see above) and the Abort Request (ABTRQ, see **CTCR — MSCAN12 Transmitter Control Register $xx07**) flags of the particular buffer.

1 = The associated message buffer is empty (not scheduled).
0 = The associated message buffer is full (loaded with a message due for transmission).

NOTE

The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

	Bit 7	6	5	4	3	2	1	Bit 0
	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
RESET:	0	0	0	0	0	0	0	0

ABTRQ2 - ABTRQ0 — Abort Request

The CPU sets this bit to request that an already scheduled message buffer (TXE = 0) shall be aborted. The MSCAN12 will grant the request when the message is not already under transmission. When a message is aborted the associated TXE and the Abort Acknowledge flag (ABTAK, see **CTFLG — MSCAN12 Transmitter Flag Register $xx06**) will be set and an TXE interrupt will occur if enabled. The CPU can not reset ABTRQx. ABTRQx is reset implicitly whenever the associated TXE flag is set.

1 = Abort request pending.
0 = No abort request.

NOTE

The software must not clear one or more of the TXE flags in CTFGL and simultaneously set the respective ABTRQ bit(s).

TXEIE2 - TXEIE0 — Transmitter Empty Interrupt Enable

1 = A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.
0 = No interrupt will be generated from this event.

NOTE

The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.

CIDAC — MSCAN12 Identifier Acceptance Control Register **$xx08**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
RESET:	0	0	0	0	0	0	0	0

IDAM1- IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organisation (see **14.4 Identifier Acceptance Filter**). **Table 40** summarizes the different settings. In Filter Closed mode no messages will be accepted such that the foreground buffer will never be reloaded.

Table 41 Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32 bit Acceptance Filters
0	1	Four 16 bit Acceptance Filters
1	0	Eight 8 bit Acceptance Filters
1	1	Filter Closed

IDHIT2- IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN12 sets these flags to indicate an identifier acceptance hit (see **14.4 Identifier Acceptance Filter**). **Table 40** summarizes the different settings.

Table 42 Identifier Acceptance Hit Indication

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 Hit
0	0	1	Filter 1 Hit
0	1	0	Filter 2 Hit
0	1	1	Filter 3 Hit
1	0	0	Filter 4 Hit
1	0	1	Filter 5 Hit
1	1	0	Filter 6 Hit
1	1	1	Filter 7 Hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer the indicators are updated as well.

NOTE

The CIDAC register can only be written if the SFTRES bit in CMCR0 is set.

CRXERR — MSCAN12 Receive Error Counter **$xx0E**

	Bit 7	6	5	4	3	2	1	Bit 0
	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
RESET:	0	0	0	0	0	0	0	0

This register reflects the status of the MSCAN12 receive error counter. The register is read only.

CTXERR — MSCAN12 Transmit Error Counter **$xx0F**

	Bit7	6	5	4	3	2	1	Bit 0
	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
RESET:	0	0	0	0	0	0	0	0

This register reflects the status of the MSCAN12 transmit error counter. The register is read only.

NOTE

Both error counters may only be read when in Sleep or Soft Reset Mode.

14.12.2 MSCAN12 Identifier Acceptance Registers (CIDAR0-7)

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN12 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers all four acceptance and mask registers are applied. For standard identifiers only the first two (IDAR0, IDAR1) are applied. In the latter case it is required to program the mask register CIDMR1 in the three last bits (AC2 - AC0) to “don’t care”.

Address	Bit7	6	5	4	3	2	1	Bit 0	Name
$xx10	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR0
$xx11	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR1
$xx12	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR2
$xx13	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR3
RESET:	U	U	U	U	U	U	U	U	

Figure 38 Identifier acceptance register (1st bank)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xx18	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR4
$xx19	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR5
$xx1A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR6
$xx1B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	CIDAR7
RESET:	U	U	U	U	U	U	U	U	

Figure 39 Identifier acceptance registers (2nd bank)

AC7 – AC0 — Acceptance Code Bits

AC7 – AC0 comprise a user defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE

The CIDAR0-7 registers can only be written if the SFTRES bit in CMCR0 is set.

14.12.3 MSCAN12 Identifier Mask Registers (CIDMR0-7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xx14	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR0
$xx15	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR1
$xx16	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR2
$xx17	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR3
RESET:	U	U	U	U	U	U	U	U	

Figure 40 Identifier mask registers (1st bank)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$xx1C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR4
$xx1D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR5
$xx1E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR6
$xx1F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	CIDMR7
RESET:	U	U	U	U	U	U	U	U	

Figure 41 Identifier mask registers (2nd bank)

AM7 – AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit, before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

Bit description:

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

NOTE

The CIDMR0-7 registers can only be written if the SFTRES bit in CMCR0 is set.

PCTLCAN — MSCAN12 Port CAN Control Register **$xx3D**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	PUECAN	RDRCAN
RESET:	0	0	0	0	0	0	0	0

The following bits control pins 7 through 2 of Port CAN. Pins 1 and 0 are reserved for the RxCan (input only) and TxCan (output only) pins.

PUECAN — Pull Enable Port CAN
- 1 = Pull mode enabled for Port CAN.
- 0 = Pull mode disabled for Port CAN.

The pull mode (pull-up or pull-down) for Port CAN is defined in the chip specification.

RDRCAN — Reduced Drive Port CAN
- 1 = Reduced drive enabled for Port CAN.
- 0 = Reduced drive disabled for Port CAN.

PORTCAN — MSCAN12 Port CAN Data Register **$xx3E**

	Bit 7	6	5	4	3	2	1	Bit 0
	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCan	RxCan
RESET:	U	U	U	U	U	U	U	U

PCAN7 – PCAN2 — Port CAN Data Bits

Writing to PCANx stores the bit value in an internal bit memory. This value is driven to the respective pin only if DDRCANx = 1.

Reading PCANx returns

- the value of the internal bit memory driven to the pin, if DDRCANx = 1
- the value of the respective pin, if DDRCANx = 0

Reading bits 1 and 0 returns the value of the TxCan and RxCan pins, respectively.

DDRCAN — MSCAN12 Port CAN Data Direction Register **$xx3F**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0
RESET:	0	0	0	0	0	0	0	0

DDRCAN7 – DDRCAN2 — Data Direction Port CAN Bits
- 1 = Respective I/O pin is configured for output.
- 0 = Respective I/O pin is configured for input.

15 Analog-To-Digital Converter

The ATD is an 8-channel, 10-bit, multiplexed-input successive-approximation analog-to-digital converter, accurate to ±2 least significant bits (LSB). It does not require external sample and hold circuits because of the type of charge redistribution technique used. The ATD converter timing is synchronized to the system P clock. The ATD module consists of a 16-word (32-byte) memory-mapped control register block used for control, testing and configuration.

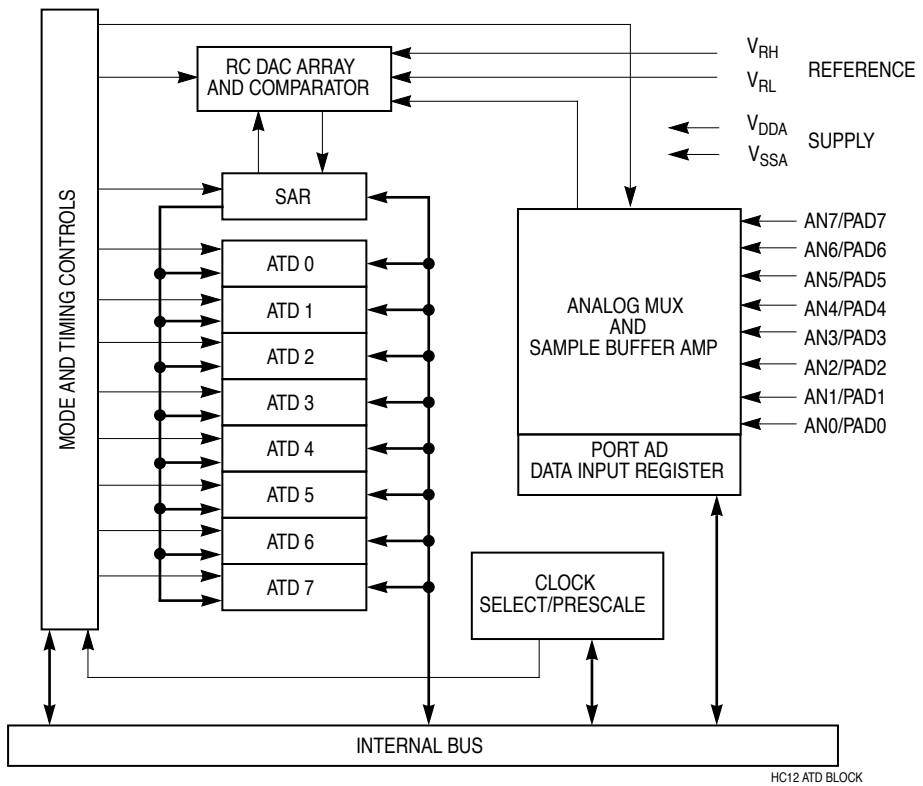


Figure 42 Analog-to-Digital Converter Block Diagram

15.1 Functional Description

A single conversion sequence consists of four or eight conversions, depending on the state of the select 8 channel mode (S8CM) bit when ATDCTL5 is written. There are eight basic conversion modes. In the non-scan modes, the SCF bit is set after the sequence of four or eight conversions has been performed and the ATD module halts. In the scan modes, the SCF bit is set after the first sequence of four or eight conversions has been performed, and the ATD module continues to restart the sequence. In both modes, the CCF bit associated with each register is set when that register is loaded with the appropriate conversion result. That flag is cleared automatically when that result register is read. The conversions are started by writing to the control registers.

15.2 ATD Registers

ATDCTL0 — Reserved **$0060**

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

Write to this register will abort current conversion sequence.
READ: any time, WRITE: any time.

ATDCTL1 — Reserved **$0061**

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

WRITE: write to this register has no meaning.
READ: Special Mode only.

ATDCTL2 — ATD Control Register 2 **$0062**

	Bit 7	6	5	4	3	2	1	Bit 0
	ADPU	AFFC	AWAI	0	0	0	ASCIE	ASCIF
RESET:	0	0	0	0	0	0	0	0

The ATD control register 2 and 3 are used to select the power up mode, interrupt control, and freeze control. Writes to these registers abort any current conversion sequence.
Read or write anytime except ASCIF bit, which cannot be written.
Bit positions ATDCTL2[4:2] and ATDCTL3[7:2] are unused and always read as zeros.

ADPU — ATD Disable
0 = Disables the ATD, including the analog section for reduction in power consumption.
1 = Allows the ATD to function normally.
Software can disable the clock signal to the ATD converter and power down the analog circuits to reduce power consumption. When reset to zero, the ADPU bit aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off, the ATD requires a period of recovery time to stabilize the analog circuits after setting the ADPU bit.

AFFC — ATD Fast Flag Clear All
0 = ATD flag clearing operates normally (read the status register before reading the result register to clear the associate CCF bit).
1 = Changes all ATD conversion complete flags to a fast clear sequence. Any access to a result register (ATD0–7) will cause the associated CCF flag to clear automatically if it was set at the time.

AWAI — ATD Stop in Wait Mode
0 = ATD continues to run when the MCU is in wait mode
1 = ATD stops to save power when the MCU is in wait mode

ASCIE — ATD Sequence Complete Interrupt Enable
0 = Disables ATD interrupt
1 = Enables ATD interrupt on sequence complete

ASCIF — ATD Sequence Complete Interrupt

Cannot be written in any mode.

0 = No ATD interrupt occurred

1 = ATD sequence complete

ATDCTL3 — ATD Control Register 3 **$0063**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	FRZ1	FRZ0
RESET:	0	0	0	0	0	0	0	0

FRZ1, FRZ0 — Background Debug (Freeze) Enable (suspend module operation at breakpoint)

When debugging an application, it is useful in many cases to have the ATD pause when a breakpoint is encountered. These two bits determine how the ATD will respond when background debug mode becomes active.

Table 43 ATD Response to Background Debug Enable

FRZ1	FRZ0	ATD Response
0	0	Continue conversions in active background mode
0	1	Reserved
1	0	Finish current conversion, then freeze
1	1	Freeze when BDM is active

ATDCTL4 — ATD Control Register 4 **$0064**

	Bit 7	6	5	4	3	2	1	Bit 0
	S10BM	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
RESET:	0	0	0	0	0	0	0	1

The ATD control register 4 is used to select the clock source and set up the prescaler. Writes to the ATD control registers initiate a new conversion sequence. If a write occurs while a conversion is in progress, the conversion is aborted and ATD activity halts until a write to ATDCTL5 occurs.

S10BM — ATD 10-bit Mode Control

0 = 8 bit operation

1 = 10 bit operation

SMP1, SMP0 — Select Sample Time

These bits are used to select one of four sample times after the buffered sample and transfer has occurred.

Table 44 Final Sample Time Selection

SMP1	SMP0	Final Sample Time	Total 8-Bit Conversion Time	Total 10-Bit Conversion Time
0	0	2 ATD clock periods	18 ATD clock periods	20 ATD clock periods
0	1	4 ATD clock periods	20 ATD clock periods	22 ATD clock periods
1	0	8 ATD clock periods	24 ATD clock periods	26 ATD clock periods
1	1	16 ATD clock periods	32 ATD clock periods	34 ATD clock periods

PRS4, PRS3, PRS2, PRS1, PRS0 — Select Divide-By Factor for ATD P-Clock Prescaler.

The binary value written to these bits (1 to 31) selects the divide-by factor for the modulo counter-based prescaler. The P clock is divided by this value plus one and then fed into a ÷2 circuit to generate the ATD module clock. The divide-by-two circuit insures symmetry of the output clock signal. Clearing these bits causes the prescale value default to one which results in a ÷2 prescale factor. This signal is then

fed into the ÷2 logic. The reset state divides the P clock by a total of four and is appropriate for nominal operation between 2 MHz and 8 MHz bus rate. **Table 45** shows the divide-by operation and the appropriate range of system clock frequencies.

Table 45 Clock Prescaler Values

Prescale Value	Total Divisor	Max P Clock[1]	Min P Clock[2]
00000	÷2	4 MHz	1 MHz
00001	÷4	8 MHz	2 MHz
00010	÷6	8 MHz	3 MHz
00011	÷8	8 MHz	4 MHz
00100	÷10	8 MHz	5 MHz
00101	÷12	8 MHz	6 MHz
00110	÷14	8 MHz	7 MHz
00111	÷16	8 MHz	8 MHz
01xxx	Do Not Use		
1xxxx			

NOTES:
1. Maximum conversion frequency is 2 MHz. Maximum P clock divisor value will become maximum conversion rate that can be used on this ATD module.
2. Minimum conversion frequency is 500 KHz. Minimum P clock divisor value will become minimum conversion rate that this ATD can perform.

ATDCTL5 — ATD Control Register 5 **$0065**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	S8CM	SCAN	MULT	CD	CC	CB	CA
RESET:	0	0	0	0	0	0	0	0

The ATD control register 5 is used to select the conversion modes, the conversion channel(s), and initiate conversions.
Read or write anytime. A write to ATDCTL5 initiates a new conversion sequence. If a conversion sequence is in progress when a write occurs, that sequence is aborted and the SCF and CCF bits are reset.

S8CM — Select 8 Channel Mode
- 0 = Conversion sequence consists of four conversions
- 1 = Conversion sequence consists of eight conversions

SCAN — Enable Continuous Channel Scan
- 0 = Single conversion sequence
- 1 = Continuous conversion sequences (scan mode)

When a conversion sequence is initiated by a write to the ATDCTL register, the user has a choice of performing a sequence of four (or eight, depending on the S8CM bit) conversions or continuously performing four (or eight) conversion sequences.

MULT — Enable Multichannel Conversion
- 0 = ATD sequencer runs all four or eight conversions on a **single** input channel selected via the CD, CC, CB, and CA bits.
- 1 = ATD sequencer runs each of the four or eight conversions on **sequential** channels in a specific group. Refer to **Table 46**.

Table 46 Multichannel Mode Result Register Assignment

S8CM	CD	CC	CB	CA	Channel Signal	Result in ADRx if MULT = 1
0	0	0	0	0	AN0	ADR0
			0	1	AN1	ADR1
			1	0	AN2	ADRx
			1	1	AN3	ADR3
0	0	1	0	0	AN4	ADR0
			0	1	AN5	ADR1
			1	0	AN6	ADR2
			1	1	AN7	ADR3
0	1	0	0	0	Reserved	ADR0
			0	1	Reserved	ADR1
			1	0	Reserved	ADR2
			1	1	Reserved	ADR3
0	1	1	0	0	V_{RH}	ADR0
			0	1	V_{RL}	ADR1
			1	0	$(V_{RH} + V_{RL})/2$	ADR2
			1	1	TEST/Reserved	ADR3
1	0	0	0	0	AN0	ADR0
		0	0	1	AN1	ADR1
		0	1	0	AN2	ADR2
		0	1	1	AN3	ADR3
		1	0	0	AN4	ADR4
		1	0	1	AN5	ADR5
		1	1	0	AN6	ADR6
		1	1	1	AN7	ADR7
1	1	0	0	0	Reserved	ADR0
		0	0	1	Reserved	ADR1
		0	1	0	Reserved	ADR2
		0	1	1	Reserved	ADR3
		1	0	0	V_{RH}	ADR4
		1	0	1	V_{RL}	ADR5
		1	1	0	$(V_{RH} + V_{RL})/2$	ADR6
		1	1	1	TEST/Reserved	ADR7

Shaded bits are “don’t care” if MULT = 1 and the entire block of four or eight channels make up a conversion sequence. When MULT = 0, all four bits (CD, CC, CB, and CA) must be specified and a conversion sequence consists of four or eight consecutive conversions of the single specified channel.

ATDSTAT — ATD Status Register **$0066**

	Bit 7	6	5	4	3	2	1	Bit 0
	SCF	0	0	0	0	CC2	CC1	CC0
RESET:	0	0	0	0	0	0	0	0

ATDSTAT — ATD Status Register **$0067**

	Bit 7	6	5	4	3	2	1	Bit 0
	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
RESET:	0	0	0	0	0	0	0	0

The ATD status registers contain the flags indicating the completion of ATD conversions.

Normally, it is read-only. In special mode, the SCF bit and the CCF bits may also be written.

SCF — Sequence Complete Flag

This bit is set at the end of the conversion sequence when in the single conversion sequence mode (SCAN = 0 in ATDCTL5) and is set at the end of the first conversion sequence when in the continuous conversion mode (SCAN = 1 in ATDCTL5). When AFFC = 0, SCF is cleared when a write is performed to ATDCTL5 to initiate a new conversion sequence. When AFFC = 1, SCF is cleared after the first result register is read.

CC[2:0] — Conversion Counter for Current Sequence of Four or Eight Conversions

This 3-bit value reflects the contents of the conversion counter pointer in a four or eight count sequence. This value also reflects which result register will be written next, indicating which channel is currently being converted.

CCF[7:0] — Conversion Complete Flags

Each of these bits are associated with an individual ATD result register. For each register, this bit is set at the end of conversion for the associated ATD channel and remains set until that ATD result register is read. It is cleared at that time if AFFC bit is set, regardless of whether a status register read has been performed (i.e., a status register read is not a pre-qualifier for the clearing mechanism when AFFC = 1). Otherwise the status register must be read to clear the flag.

ATDTSTH — ATD Test Register **$0068**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
RESET:	0	0	0	0	0	0	0	0

ATDTSTL — ATD Test Register **$0069**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0
RESET:	0	0	0	0	0	0	0	0

The test registers control various special modes which are used during manufacturing. The test register can be read or written only in the special modes. In the normal modes, reads of the test register return zero and writes have no effect.

SAR[9:0] — SAR Data

Reads of this byte return the current value in the SAR. Writes to this byte change the SAR to the value written. Bits SAR[9:0] reflect the ten SAR bits used during the resolution process for an 10-bit result.

RST — Module Reset Bit

When set, this bit causes all registers and activity in the module to assume the same state as out of power-on reset (except for ADPU bit in ATDCTL2, which remains set, allowing the ATD module to remain enabled).

TSTOUT — Multiplex Output of TST[3:0] (Factory Use)

TST[3:0] — Test Bits 3 to 0 (Reserved)

Selects one of 16 reserved factory testing modes

PORTAD — Port AD Data Input Register **$006F**

	Bit 7	6	5	4	3	2	1	Bit 0
	PAD7	PAD6	PAD5	PAD4	PAD3	PAD2	PAD1	PAD0
RESET:	–	–	–	–	–	–	–	–

PAD[7:0] — Port AD Data Input Bits

After reset these bits reflect the state of the input pins.

May be used for general-purpose digital input. When the software reads PORTAD, it obtains the digital levels that appear on the corresponding port AD pins. Pins with signals not meeting V_{IL} or V_{IH} specifications will have an indeterminate value. Writes to this register have no meaning at any time.

ADRx0H — A/D Converter Result Register 0 **$0070**
ADRx0L — A/D Converter Result Register 0 **$0071**
ADRx1H — A/D Converter Result Register 1 **$0072**
ADRx1L — A/D Converter Result Register 1 **$0073**
ADRx2H — A/D Converter Result Register 2 **$0074**
ADRx2L — A/D Converter Result Register 2 **$0075**
ADRx3H — A/D Converter Result Register 3 **$0076**
ADRx3L — A/D Converter Result Register 3 **$0077**
ADRx4H — A/D Converter Result Register 4 **$0078**
ADRx4L — A/D Converter Result Register 4 **$0079**
ADRx5H — A/D Converter Result Register 5 **$007A**
ADRx5L — A/D Converter Result Register 5 **$007B**
ADRx6H — A/D Converter Result Register 6 **$007C**
ADRx6L — A/D Converter Result Register 6 **$007D**
ADRx7H — A/D Converter Result Register 7 **$007E**
ADRx7L — A/D Converter Result Register 7 **$007F**

ADRxH	Bit 15	14	13	12	11	10	9	Bit 8
ADRxL	Bit 7	6	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

ADRxH[15:8], ADRxL[7:0] — ATD Conversion Result

The reset condition for these registers is undefined.

These bits contain the left justified, unsigned result from the ATD conversion. The channel from which this result was obtained is dependent on the conversion mode selected. These registers are always read-only in normal mode.

15.3 ATD Mode Operation

STOP — causes all clocks to halt (if the S bit in the CCR is zero). The system is placed in a minimum-power standby mode. This aborts any conversion sequence in progress. During STOP recovery, the ATD must delay for the STOP recovery time (t_{SR}) before initiating a new ATD conversion sequence.

WAIT — ATD conversion continues unless AWAI bit in ATDCTL2 register is set.

BDM — Debug options available as set in register ATDCTL3.

USER — ATD continues running unless ADPU is cleared.

ADPU — ATD operations are stopped if ADPU = 0, but registers are accessible.

16 Development Support

Development support involves complex interactions between MC68HC912BC32 resources and external development systems. The following section concerns instruction queue and queue tracking signals, background debug mode, breakpoints, and instruction tagging.

16.1 Instruction Queue

It is possible to monitor CPU activity on a cycle-by-cycle basis for debugging.The CPU12 instruction queue provides at least three bytes of program information to the CPU when instruction execution begins. The CPU12 always completely finishes executing an instruction before beginning to execute the next instruction. Status signals IPIPE[1:0] provide information about data movement in the queue and indicate when the CPU begins to execute instructions. Information available on the IPIPE[1:0] pins is time multiplexed. External circuitry can latch data movement information on rising edges of the E-clock signal; execution start information can be latched on falling edges. **Table 47** shows the meaning of data on the pins.

Table 47 IPIPE Decoding

Data Movement — IPIPE[1:0] Captured at Rising Edge of E Clock[1]		
IPIPE[1:0]	**Mnemonic**	**Meaning**
0:0	—	No Movement
0:1	LAT	Latch Data From Bus
1:0	ALD	Advance Queue and Load From Bus
1:1	ALL	Advance Queue and Load From Latch
Execution Start — IPIPE[1:0] Captured at Falling Edge of E Clock[2]		
IPIPE[1:0]	**Mnemonic**	**Meaning**
0:0	—	No Start
0:1	INT	Start Interrupt Sequence
1:0	SEV	Start Even Instruction
1:1	SOD	Start Odd Instruction

NOTES:
1. Refers to data that was on the bus at the previous E falling edge.
2. Refers to bus cycle starting at this E falling edge.

Program information is fetched a few cycles before it is used by the CPU. In order to monitor cycle-by-cycle CPU activity, it is necessary to externally reconstruct what is happening in the instruction queue. Internally the MCU only needs to buffer the data from program fetches. For system debug it is necessary to keep the data and its associated address in the reconstructed instruction queue. The raw signals required for reconstruction of the queue are ADDR, DATA, R/$\overline{W}$, ECLK, and status signals IPIPE[1:0].

The instruction queue consists of two 16-bit queue stages and a holding latch on the input of the first stage. To advance the queue means to move the word in the first stage to the second stage and move the word from either the holding latch or the data bus input buffer into the first stage. To start even (or odd) instruction means to execute the opcode in the high-order (or low-order) byte of the second stage of the instruction queue.

16.2 Background Debug Mode

Background debug mode (BDM) is used for system development, in-circuit testing, field testing, and programming. BDM is implemented in on-chip hardware and provides a full set of debug options.

Because BDM control logic does not reside in the CPU, BDM hardware commands can be executed while the CPU is operating normally. The control logic generally uses CPU dead cycles to execute these commands, but can steal cycles from the CPU when necessary. Other BDM commands are firmware based, and require the CPU to be in active background mode for execution. While BDM is active, the CPU executes a firmware program located in a small on-chip ROM that is available in the standard 64-Kbyte memory map only while BDM is active.

The BDM control logic communicates with an external host development system serially, via the BKGD pin. This single-wire approach minimizes the number of pins needed for development support.

16.2.1 BDM Serial Interface

The BDM serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 E-clock cycles per bit (nominal speed). The interface times out if 512 E-clock cycles occur between falling edges from the host. The hardware clears the command register when this time-out occurs.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to MCU clocks but asynchronous to the external host. The internal clock signal is shown for reference in counting cycles.

Figure 43 shows an external host transmitting a logic one or zero to the BKGD pin of a target M68HC12 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target E cycles later, the target senses the bit level on the BKGD pin. Typically the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Since the target does not drive the BKGD pin during this period, there is no need to treat the line as an open-drain signal during host-to-target transmissions.

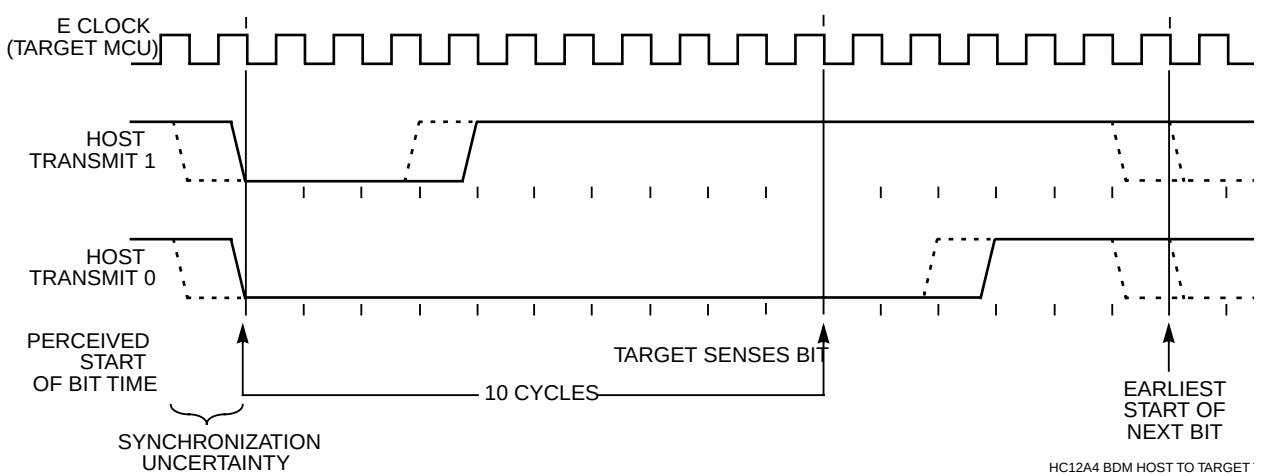


Figure 43 BDM Host to Target Serial Bit Timing

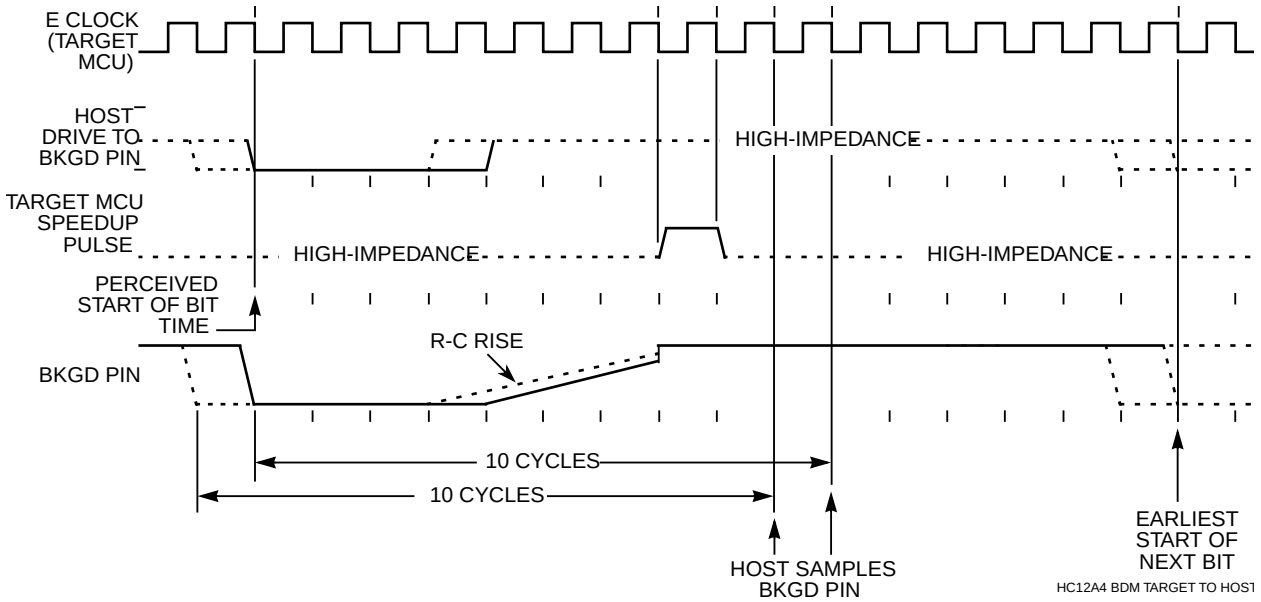


Figure 44 BDM Target to Host Serial Bit Timing (Logic 1)

Figure 44 shows the host receiving a logic one from the target MC68HC912BC32 MCU. Since the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target E cycles). The host must release the low drive before the target MCU drives a brief active-high speed-up pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about ten cycles after it started the bit time.

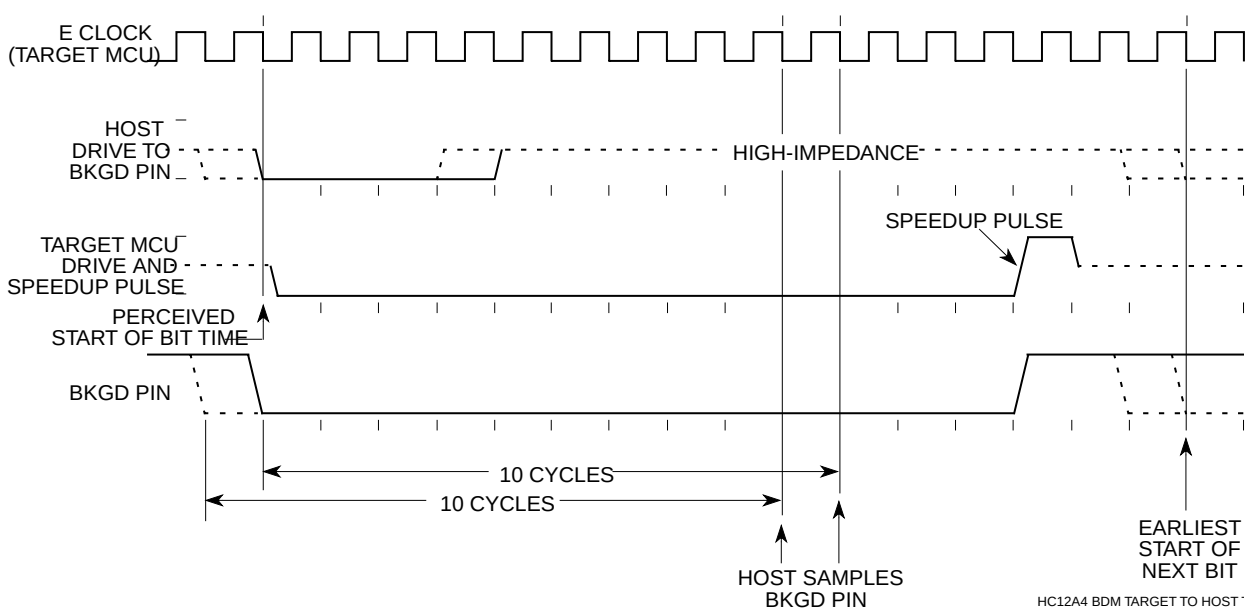


Figure 45 BDM Target to Host Serial Bit Timing (Logic 0)

Figure 45 shows the host receiving a logic zero from the target MC68HC912BC32 MCU. Since the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target MC68HC912BC32 finishes it. Since the target wants the host to receive a logic zero, it drives the BKGD pin low for 13 E-clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about ten cycles after starting the bit time.

16.2.2 Enabling BDM Firmware Commands

BDM is available in all operating modes, but must be made active before firmware commands can be executed. BDM is enabled by setting the ENBDM bit in the BDM STATUS register via the single wire interface (using a hardware command; WRITE_BD_BYTE at $FF01). BDM must then be activated to map BDM registers and ROM to addresses $FF00 to $FFFF and to put the MCU in active background mode.

After the firmware is enabled, BDM can be activated by the hardware BACKGROUND command, by the BDM tagging mechanism, or by the CPU BGND instruction. An attempt to activate BDM before firmware has been enabled causes the MCU to resume normal instruction execution after a brief delay.

BDM becomes active at the next instruction boundary following execution of the BDM BACKGROUND command, but tags activate BDM before a tagged instruction is executed.

In special single-chip mode, background operation is enabled and active immediately out of reset. This active case replaces the M68HC11 boot function, and allows programming a system with blank memory.

While BDM is active, a set of BDM control registers are mapped to addresses $FF00 to $FF06. The BDM control logic uses these registers which can be read anytime by BDM logic, not user programs. Refer to **16.2.4 BDM Registers** for detailed descriptions.

Some on-chip peripherals have a BDM control bit which allows suspending the peripheral function during BDM. For example, if the timer control is enabled, the timer counter is stopped while in BDM. Once normal program flow is continued, the timer counter is re-enabled to simulate real-time operations.

16.2.3 BDM Commands

All BDM command opcodes are eight bits long, and can be followed by an address and/or data, as indicated by the instruction. These commands do not require the CPU to be in active BDM mode for execution.

The host controller must wait 150 cycles for a non-intrusive BDM command to execute before another command can be sent. This delay includes 128 cycles for the maximum delay for a dead cycle. For data read commands, the host must insert this delay between sending the address and attempting to read the data.

BDM logic retains control of the internal buses until a read or write is completed. If an operation can be completed in a single cycle, it does not intrude on normal CPU operation. However, if an operation requires multiple cycles, CPU clocks are frozen until the operation is complete.

There are two types of BDM commands: hardware and firmware. Hardware commands allow target system memory to be read or written. Target system memory includes all memory that is accessible by the CPU12 including on-chip RAM, EEPROM, on-chip I/O and control registers, and external memory connected to the target HC12 MCU. Hardware commands are implemented in hardware logic and do not require the HC12 MCU to be in BDM mode for execution. The control logic watches the CPU12 buses to find a free bus cycle to execute the command so that the background access does not disturb the running application programs. If a free cycle is not found within 128 E-clock cycles, the CPU12 is momentarily frozen so the control logic can steal a cycle. Refer to **Table 48** for commands implemented in BDM control logic.

Table 48 BDM Hardware Commands

Command	Opcode (Hex)	Data	Description
BACKGROUND	90	None	Enter background mode (if firmware enabled).
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
STATUS[1]	E4	FF01, 0000 0000 (out)	READ_BD_BYTE $FF01. Running user code (BGND instruction is not allowed).
		FF01, 1000 0000 (out)	READ_BD_BYTE $FF01. BGND instruction is allowed.
		FF01, 1100 0000 (out)	READ_BD_BYTE $FF01. Background mode active (waiting for single wire serial command).
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access) must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access) must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
ENABLE_ FIRMWARE[2]	C4	FF01, 1xxx xxxx(in)	Write byte $FF01, set the ENBDM bit. This allows execution of commands which are implemented in firmware. Typically, read STATUS, OR in the MSB, write the result back to STATUS.
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access) must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access) must be aligned access.

NOTES:
1. STATUS command is a specific case of the READ_BD_BYTE command.
2. ENABLE_FIRMWARE is a specific case of the WRITE_BD_BYTE command.

The second type of BDM commands are called firmware commands because they are implemented in a small ROM within the HC12 MCU. The CPU must be in background mode to execute firmware commands. The usual way to get to background mode is by the hardware command BACKGROUND. The BDM ROM is located at $FF20 to $FFFF while BDM is active. There are also seven bytes of BDM registers which are located at $FF00 to $FF06 while BDM is active. The CPU executes code from this ROM to perform the requested operation. The BDM firmware watches for serial commands and executes them as they are received. The firmware commands are shown in **Table 49**.

Table 49 BDM Firmware Commands

Command	Opcode (Hex)	Data	Description
READ_NEXT	62	16-bit data out	X = X + 2; Read next word pointed-to by X
READ_PC	63	16-bit data out	Read program counter
READ_D	64	16-bit data out	Read D accumulator
READ_X	65	16-bit data out	Read X index register
READ_Y	66	16-bit data out	Read Y index register
READ_SP	67	16-bit data out	Read stack pointer
WRITE_NEXT	42	16-bit data in	X = X + 2; Write next word pointed-to by X
WRITE_PC	43	16-bit data in	Write program counter
WRITE_D	44	16-bit data in	Write D accumulator
WRITE_X	45	16-bit data in	Write X index register
WRITE_Y	46	16-bit data in	Write Y index register
WRITE_SP	47	16-bit data in	Write stack pointer
GO	08	None	Go to user program
TRACE1	10	None	Execute one user instruction then return to BDM
TAGGO	18	None	Enable tagging and go to user program

Each of the hardware and firmware BDM commands start with an 8-bit command code (opcode). Depending upon the commands, a 16-bit address and/or a 16-bit data word is required as indicated in the tables by the command. All the read commands output 16-bits of data despite the byte/word implication in the command name.

The external host should wait 150 E-clock cycles for a non-intrusive BDM command to execute before another command is sent. This delay includes 128 E-clock cycles for the maximum delay for a free cycle. For data read commands, the host must insert this delay between sending the address and attempting to read the data. In the case of a write command, the host must delay after the data portion, before sending a new command, to be sure the write has finished.

The external host should delay about 32 target E-clock cycles between a firmware read command and the data portion of these commands. This allows the BDM firmware to execute the instructions needed to get the requested data into the BDM SHIFTER register.

The external host should delay about 32 target E-clock cycles after the data portion of firmware write commands to allow BDM firmware to complete the requested write operation before a new serial command disturbs the BDM SHIFTER register.

The external host should delay about 64 target E-clock cycles after a TRACE1 or GO command before starting any new serial command. This delay is needed because the BDM SHIFTER register is used as a temporary data holding register during the exit sequence to user code.

BDM logic retains control of the internal buses until a read or write is completed. If an operation can be completed in a single cycle, it does not intrude on normal CPU12 operation. However, if an operation requires multiple cycles, CPU12 clocks are frozen until the operation is complete.

16.2.4 BDM Registers

Seven BDM registers are mapped into the standard 64-Kbyte address space when BDM is active. Mapping is shown in **Table 50**.

Table 50 BDM registers

Address	Register
$FF00	BDM Instruction Register
$FF01	BDM Status Register
$FF02–$FF03	BDM Shift Register
$FF04–$FF05	BDM Address Register
$FF06	BDM CCR Holding Register

The content of the INSTRUCTION register is determined by the type of background command being executed. The STATUS register indicates BDM operating conditions. The SHIFT register contains data being received or transmitted via the serial interface. The ADDRESS register is temporary storage for BDM commands. The CCRSAV register preserves the content of the CPU12 CCR while BDM is active.

The only registers of interest to users are the STATUS register and the CCRSAV register. The other BDM registers are only used by the BDM firmware to execute commands. The registers are accessed by means of the hardware READ_BD and WRITE_BD commands, but should not be written during BDM operation (except the CCRSAV register which could be written to modify the CCR value).

The INSTRUCTION register is written by the BDM hardware as a result of serial data shifted in on the BKGD pin. It is readable and writable in Special Peripheral mode on the parallel bus. It is discussed here for two conditions: when a **hardware** command is executed and when a **firmware** command is executed.

The INSTRUCTION register can be read or written in all modes. The hardware clears the INSTRUCTION register if 512 E-clock cycles occur between falling edges from the host.

INSTRUCTION — BDM Instruction Register (hardware command) **(BDM) $FF00**

	Bit 7	6	5	4	3	2	1	Bit 0
	H/F	DATA	R/W	BKGND	W/B	BD/U	0	0
RESET:	0	0	0	0	0	0	0	0

The bits in the BDM instruction register have the following meanings when a **hardware** command is executed.

H/F — Hardware/Firmware Flag
0 = Firmware instruction
1 = Hardware instruction

DATA — Data Flag
Indicates that data accompanies the command.
0 = No data
1 = Data included in command

R/W — Read/Write Flag
0 = Write
1 = Read

BKGND — Hardware request to enter active background mode
0 = Not a hardware background command
1 = Hardware background command (INSTRUCTION = $90)

W/B — Word/Byte Transfer Flag
0 = Byte transfer
1 = Word transfer

BD/U — BDM Map/User Map Flag
Indicates whether BDM registers and ROM are mapped to addresses $FF00 to $FFFF in the standard 64-Kbyte address space. Used only by hardware read/write commands.
0 = BDM resources not in map
1 = BDM resources in map

INSTRUCTION — BDM Instruction Register (firmware command) **(BDM) $FF00**

	Bit 7	6	5	4	3	2	1	Bit 0
	H/F	DATA	R/W	TTAGO			REGN	
RESET:	0	0	0	0	0	0	0	0

The bits in the BDM instruction register have the following meanings when a **firmware** command is executed.

H/F — Hardware/Firmware Flag
0 = Firmware control logic
1 = Hardware control logic

DATA — Data Flag
0 = No data
1 = Data included in command

R/W — Read/Write Flag
0 = Write
1 = Read

TTAGO — Trace, Tag, Go Field

Table 51 TTAGO Decoding

TTAGO Value	Instruction
00	—
01	GO
10	TRACE1
11	TAGGO

REGN — Register/Next Field
Indicates which register is being affected by a command. In the case of a READ_NEXT or WRITE_NEXT command, index register X is pre-incremented by 2 and the word pointed to by X is then read or written.

Table 52 REGN Decoding

REGN Value	Instruction
000	—
001	—
010	READ/WRITE NEXT
011	PC
100	D
101	X
110	Y
111	SP

STATUS — BDM Status Register **(BDM) $FF01**

	Bit 7	6	5	4	3	2	1	Bit 0	
	ENBDM	BDMACT	ENTAG	SDV	TRACE	0	0	0	
RESET	0[1]	1	0	0	0	0	0	0	Sp Sing Chip & Peripheral
RESET:	0	0	0	0	0	0	0	0	All other modes

NOTES:
1. ENBDM is set to 1 by the firmware in Special Single Chip mode.

This register can be read or written by BDM commands or firmware.

ENBDM — Enable BDM (permit active background debug mode)
0 = BDM cannot be made active (hardware commands still allowed)
1 = BDM can be made active to allow firmware commands

BDMACT — Background Mode Active Status
BDMACT becomes set as active BDM mode is entered so that the BDM firmware ROM is enabled and put into the map. BDMACT is cleared by a carefully timed store instruction in the BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map. This bit has 4 clock cycles write delay.
0 = BDM not active; BDM ROM and registers are not in map
1 = BDM active and waiting for serial commands; BDM ROM and registers are in map
Care should be taken that the BDMACT bit is not unintentionally changed with the WRITE_NEXT firmware command. If it is unintentionally changed from 1 to 0, it will cause a system runaway as it will disable the BDM firmware ROM while the CPU12 is executing BDM firmware.The following two commands show how BDMACT may unintentionally get changed from 1 to 0.
WRITE_X with data $FEFE
WRITE_NEXT with data $C400
The first command writes the data $FEFE to the X index register.The second command writes the data $C4 to the $FF00 INSTRUCTION register and also writes the data $00 to the $FF01 STATUS register.

ENTAG — Instruction Tagging Enable
Set by the TAGGO instruction and cleared when BDM is entered. The serial system is disabled and the tag function enabled 16 cycles after this bit is written.
0 = Tagging not enabled, or BDM active
1 = Tagging active (BDM cannot process serial commands while tagging is active.)

SDV — Shifter Data Valid

Shows that valid data is in the serial interface shift register. Used by BDM firmware.

0 = No valid data

1 = Valid Data

TRACE — Asserted by the TRACE1 instruction

SHIFTER — BDM Shift Register **(BDM) $FF02, $FF03**

Bit 15	14	13	12	11	10	9	Bit 8
S15	S14	S13	S12	S11	S10	S9	S8

Bit 7	6	5	4	3	2	1	Bit 0
S7	S6	S5	S4	S3	S2	S1	S0

The 16-bit SHIFTER register contains data being received or transmitted via the serial interface. It is also used by the BDM firmware for temporary storage. The register can be read or written in all modes but is not normally accessed by users.

ADDRESS — BDM Address Register **(BDM) $FF04, $FF05**

Bit 15	14	13	12	11	10	9	Bit 8
A15	A14	A13	A12	A11	A10	A9	A8

Bit 7	6	5	4	3	2	1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

The 16-bit ADDRESS register is temporary storage for BDM hardware and firmware commands. The register can be read in all modes but is not normally accessed by users. It is written only by BDM hardware.

CCRSAV — BDM CCR Holding Register **(BDM) $FF06**

Bit 7	6	5	4	3	2	1	Bit 0
CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0

The CCRSAV register is used to save the CCR of the users program when entering BDM. It is also used for temporary storage in the BDM firmware. The register is initialized by the firmware to equal the CPU CCR register.

16.3 Breakpoints

Hardware breakpoints are used to debug software on the MC68HC912BC32 by comparing actual address and data values to predetermined data in setup registers. A successful comparison will place the CPU in background debug mode (BDM) or initiate a software interrupt (SWI). Breakpoint features designed into the MC68HC912B32 include:

- Mode selection for BDM or SWI generation
- Program fetch tagging for cycle of execution breakpoint
- Second address compare in dual address modes
- Range compare by disable of low byte address
- Data compare in full feature mode for non-tagged breakpoint
- Byte masking for high/low byte data compares
- R/$\overline{W}$ compare for non-tagged compares
- Tag inhibit on BDM TRACE

16.3.1 Breakpoint Modes

Three modes of operation determine the type of breakpoint in effect.

- Dual address-only breakpoints, each of which will cause a software interrupt (SWI)
- Single full-feature breakpoint which will cause the part to enter background debug mode (BDM)
- Dual address-only breakpoints, each of which will cause the part to enter BDM

Breakpoints will not occur when BDM is active.

16.3.1.1 SWI Dual Address Mode

In this mode, dual address-only breakpoints can be set, each of which cause a software interrupt. This is the only breakpoint mode which can force the CPU to execute a SWI. Program fetch tagging is the default in this mode; data breakpoints are not possible. In the dual mode each address breakpoint is affected by the respective BKALE bit. The BKxRW, BKxRWE, BKMBH and BKMBL bits are ignored. In dual address mode the BKDBE becomes an enable for the second address breakpoint.

16.3.1.2 BDM Full Breakpoint Mode

This is a single full-featured breakpoint which causes the part to enter background debug mode. BK1ALE, BK1RW, and BK1RWE have no meaning in full breakpoint mode.

BKDBE enables data compare but has no meaning if BKPM=1. BKMBH and BKMBL allow masking of high and low byte compares but has no meaning if BKPM=1. BK0ALE enables compare of low address byte.

- Breakpoints are not allowed if the BDM mode is already active. Active mode means the CPU is executing out of the BDM ROM.
- BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. This is important because even if the ENABLE bit in the BDM is negated the CPU actually does execute the BDM ROM code. It checks the ENABLE and returns if not set. If the BDM is not serviced by the monitor then the breakpoint would be re-asserted when the BDM returns to normal CPU flow. There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

16.3.1.3 BDM Dual Address Mode

Dual address-only breakpoints, each of which cause the part to enter background debug mode. In the dual mode each address breakpoint is affected by the BKPM bit, the BKxALE bits, and the BKxRW and BKxRWE bits. In dual address mode the BKDBE becomes an enable for the second address breakpoint. The BKMBH and BKMBL bits will have no effect when in a dual address mode. BDM mode may be entered by a breakpoint only if an internal signal from the BDM indicates background debug mode is enabled. If BKPM = 1 then BKxRW, BKxRWE, BKMBH and BKMBL have no meaning.

- Breakpoints are not allowed if the BDM mode is already active. Active mode means the CPU is executing out of the BDM ROM.
- BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. This is important because even if the ENABLE bit in the BDM is negated the CPU actually does execute the BDM ROM code. It checks the ENABLE and returns if not set. If the BDM is not serviced by the monitor then the breakpoint would be re-asserted when the BDM returns to normal CPU flow. There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

16.3.2 Registers

Breakpoint operation consists of comparing data in the breakpoint address registers (BRKAH/BRKAL) to the address bus and comparing data in the breakpoint data registers (BRKDH/BRKDL) to the data bus. The breakpoint data registers can also be compared to the address bus. The scope of comparison can be expanded by ignoring the least significant byte of address or data matches.

The scope of comparison can be limited to program data only by setting the BKPM bit in breakpoint control register 0.

To trace program flow, setting the BKPM bit causes address comparison of program data only. Control bits are also available that allow checking read/write matches.

BRKCT0 — Breakpoint Control Register 0 **$0020**

	Bit 7	6	5	4	3	2	1	Bit 0
	BKEN1	BKEN0	BKPM	0	BK1ALE	BK0ALE	0	0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.
This register is used to control the breakpoint logic.

BKEN1, BKEN0 — Breakpoint Mode Enable

Table 53 Breakpoint Mode Control

BKEN1	BKEN0	Mode Selected	BRKAH/L Usage	BRKDH/L Usage	R/$\overline{W}$	Range
0	0	Breakpoints Off	—	—	—	—
0	1	SWI — Dual Address Mode	Address Match	Address Match	No	Yes
1	0	BDM — Full Breakpoint Mode	Address Match	Data Match	Yes	Yes
1	1	BDM — Dual Address Mode	Address Match	Address Match	Yes	Yes

BKPM — Break on Program Addresses

This bit controls whether the breakpoint will cause an immediate data breakpoint (next instruction boundary) or a delayed program breakpoint related to an executable opcode. Data and unexecuted opcodes cannot cause a break if this bit is set. This bit has no meaning in SWI dual address mode. The SWI mode only performs program breakpoints.

0 = On match, break at the next instruction boundary

1 = On match, break if the match is an instruction that will be executed. This uses tagging as its breakpoint mechanism.

BK1ALE — Breakpoint 1 Range Control

Only valid in dual address mode.

0 = BRKDL will not be used to compare to the address bus.

1 = BRKDL will be used to compare to the address bus.

BK0ALE — Breakpoint 0 Range Control

Valid in all modes.

0 = BRKAL will not be used to compare to the address bus.

1 = BRKAL will be used to compare to the address bus.

Table 54 Breakpoint Address Range Control

BK1ALE	BK0ALE	Address Range Selected
–	0	Upper 8-bit address only for full mode or dual mode BKP0
–	1	Full 16-bit address for full mode or dual mode BKP0
0	–	Upper 8-bit address only for dual mode BKP1
1	–	Full 16-bit address for dual mode BKP1

BRKCT1 — Breakpoint Control Register 1 **$0021**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	BKDBE	BKMBH	BKMBL	BK1RWE	BK1RW	BK0RWE	BK0RW
RESET:	0	0	0	0	0	0	0	0

This register is read/write in all modes.

BKDBE — Enable Data Bus

Enables comparing of address or data bus values using the BRKDH/L registers.

0 = BRKDH/L registers are not used in any comparison

1 = BRKDH/L registers are used to compare address or data (depending upon the mode selections BKEN1,0)

BKMBH — Breakpoint Mask High

Disables the comparing of the high byte of data when in full breakpoint mode. Used in conjunction with the BKDBE bit (which should be set)

0 = High byte of data bus (bits 15:8) are compared to BRKDH

1 = High byte is not used to in comparisons

BKMBL — Breakpoint Mask Low

Disables the matching of the low byte of data when in full breakpoint mode. Used in conjunction with the BKDBE bit (which should be set)

0 = Low byte of data bus (bits 7:0) are compared to BRKDL

1 = Low byte is not used to in comparisons.

BK1RWE — R/$\overline{W}$ Compare Enable

Enables the comparison of the R/$\overline{W}$ signal to further specify what causes a match. This bit is NOT useful in program breakpoints or in full breakpoint mode. This bit is used in conjunction with a second address in dual address mode when BKDBE=1.

0 = R/W is not used in comparisons
1 = R/W is used in comparisons

BK1RW — R/$\overline{W}$ Compare Value

When BK1RWE = 1, this bit determines the type of bus cycle to match.

0 = A write cycle will be matched
1 = A read cycle will be matched

BK0RWE — R/$\overline{W}$ Compare Enable

Enables the comparison of the R/$\overline{W}$ signal to further specify what causes a match. This bit is not useful in program breakpoints.

0 = R/$\overline{W}$ is not used in the comparisons
1 = R/$\overline{W}$ is used in comparisons

BK0RW — R/$\overline{W}$ Compare Value

When BK0RWE = 1, this bit determines the type of bus cycle to match on.

0 = Write cycle will be matched
1 = Read cycle will be matched

Table 55 Breakpoint Read/Write Control

BK1RWE	BK1RW	BK0RWE	BK0RW	Read/Write Selected
–	–	0	X	R/$\overline{W}$ is don't care for full mode or dual mode BKP0
–	–	1	0	R/$\overline{W}$ is write for full mode or dual mode BKP0
–	–	1	1	R/$\overline{W}$ is read for full mode or dual mode BKP0
0	X	–	–	R/$\overline{W}$ is don't care for dual mode BKP1
1	0	–	–	R/$\overline{W}$ is write for dual mode BKP1
1	1	–	–	R/$\overline{W}$ is read for dual mode BKP1

BRKAH — Breakpoint Address Register, High Byte **$0022**

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

These bits are used to compare against the most significant byte of the address bus.

BRKAL — Breakpoint Address Register, Low Byte **$0023**

	Bit 7	6	5	4	3	2	1	Bit 0
	Bit 7	6	5	4	3	2	1	Bit 0
Power-on RESET:	0	0	0	0	0	0	0	0

These bits are used to compare against the least significant byte of the address bus. These bits may be excluded from being used in the match if BK0ALE = 0.

BRKDH — Breakpoint Data Register, High Byte **$0024**

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

These bits are compared to the most significant byte of the data bus in full breakpoint mode or the most significant byte of the address bus in dual address modes. BKEN[1:0], BKDBE, and BKMBH control how this byte will be used in the breakpoint comparison.

BRKDL — Breakpoint Data Register, Low Byte **$0025**

	Bit 7	6	5	4	3	2	1	Bit 0
	Bit 7	6	5	4	3	2	1	Bit 0
Power-on RESET:	0	0	0	0	0	0	0	0

These bits are compared to the least significant byte of the data bus in full breakpoint mode or the least significant byte of the address bus in dual address modes. BKEN[1:0], BKDBE, BK1ALE, and BKMBL control how this byte will be used in the breakpoint comparison.

NOTE

After a power-on reset, registers BRKAH, BRKAL, BRKDH, and BRKDL are cleared but these registers are not affected by normal resets.

16.4 Instruction Tagging

The instruction queue and cycle-by-cycle CPU activity can be reconstructed in real time or from trace history that was captured by a logic analyzer. However, the reconstructed queue cannot be used to stop the CPU at a specific instruction, because execution has already begun by the time an operation is visible outside the MCU. A separate instruction tagging mechanism is provided for this purpose.

Executing the BDM TAGGO command configures two MCU pins for tagging. Tagging information is latched on the falling edge of ECLK along with program information as it is fetched. Tagging is allowed in all modes. Tagging is disabled when BDM becomes active and BDM serial commands cannot be processed while tagging is active.

$\overline{\text{TAGHI}}$ is a shared function of the BKGD pin.

$\overline{\text{TAGLO}}$ is a shared function of the PE3/$\overline{\text{LSTRB}}$ pin, a multiplexed I/O pin. For 1/4 cycle before and after the rising edge of the E clock, this pin is the $\overline{\text{LSTRB}}$ driven output.

$\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ inputs are captured at the falling edge of the E clock. A logic zero on $\overline{\text{TAGHI}}$ and/or $\overline{\text{TAGLO}}$ marks (tags) the instruction on the high and/or low byte of the program word that was on the data bus at the same falling edge of the E clock.

Table 56 shows the functions of the two tagging pins. The pins operate independently; the state of one pin does not affect the function of the other. The presence of logic level zero on either pin at the fall of ECLK performs the indicated function. Tagging is allowed in all modes. Tagging is disabled when BDM becomes active and BDM serial commands are not processed while tagging is active.

Table 56 Tag Pin Function

$\overline{\text{TAGHI}}$	$\overline{\text{TAGLO}}$	Tag
1	1	No Tag
1	0	Low Byte
0	1	High Byte
0	0	Both Bytes

The tag follows the information in the queue as the queue is advanced. When a tagged instruction reaches the head of the queue, the CPU enters active background debugging mode rather than executing the instruction. This is the mechanism by which a development system initiates hardware breakpoints.

17 Summary of Changes

The following summary lists significant changes since previous versions. Typographical errors and other minor errors which do not affect technical content or organization have not been noted.

Section 1 Introduction

Sec. 1.1. 2.7V-5.5V operation at 8 MHz changed to 4.5V-5.5V operation ay 8 MHz.
Flash EEPROM changed from 1 to 2-Kbyte Erase-Protected Boot Block.
Analog-to-digital converter changed from 8-bit to 10-bit.

Table 1 MC68HC912BC32VFU8 data removed.
MC68HC912BC32MFU8 data removed.
MC68C912BC32FU8 data removed.
MC68HC912BC32CFU8 data removed.
MC68B912BC32FU8 data removed.

Section 3 Signals

Sec. 3.3.2 Correction and addlitional information about E clock operation in single-chip, peripheral and Stop modes.

Section 4 Register Block

Table 8 Address $0043 PWPRES, Bit 7 changed to 0.
Address $0054 PWCTL, bit 2 changed from RDPP to RDP.
Address $0056, PORTP changed to PORTPP.
Address $0057, DDRP changed to PORTPD.
Address $0062, Bit 5 changed to AWAI.
Address $0064, Bit 7 changed to S10BM.
Addresses $0071, $0073, $0075, $0077, $0079, $007B, $007D and $007F added. These are registers ADR0-7 low.

Section 5 Operating Modes and Resource Mapping

Sec. 5.1 Table 9, row 2 and row 6, DATA[15:0] changed to DATA[7:0]

Sec. 5.3 Fourth paragraph, irrelevent information about memory expansion removed.

Sec 5.3.4 MISC register, corrections and additional information regarding reset conditions.

Section 8 EEPROM

Sec. 8.2 EEPROT register, STPROT bit described
EEPROG register, note added to EELAT bit description

Section 11 Pulse-Width Modulator

Page 71 Added description of two additional initialization operations which occur in Center Aligne mode.

Page 72-74 Updated versions of figures 15-17 included to replace block diagrams of PWM Left-Aligned output channel, Center-Aligned output channel and PWM clock sources

Page 75 Note added concerning bits 6 and 7.

Page 76 Descriptions of bits PPOL3–PPOL0, the term “clock cycle” has been changed to “period.”
Note added concerning high and low time count in the duty registers.

Page 77 In PWPRES, Bit 7 changed to 0.

Page 78 PWSCAL0 register description, procedure for generating clock S0 has been corrected.
PWSCAL1 register description, procedure for generating clock S1 has been corrected.

Page 79 PWCNTx register description, last sentence, removed phrase regarding center-aligned mode.
PWPERx register description, corrected formula for calculating period.
PWPERx register reset condition changed to $FF

Page 80 PWDTYx register description, corrected formula for calculating duty cycle (CENTR=1).
PWDTYx register reset condition changed to $FF.
PWCTL bit 2 changed from RDPP to RDP.

Page 81 PWTST register decription expanded to include functions only accessible in Special mode.

DISCP bit description, removed comment regarding center-aligned output mode.
Note added to PORTPPconcerning write to this register.

Page 82 DDRD changed to PORTPD - Port P data direction register.
Table 23, first row, Output is Low; second row, Output is High.

Throughout References to P clock changed to E clock

Section 13 Serial Interface

Page 97 Table 29, third row, fourth column (WOMS), 1 changed to 0.

Section 14 MSCAN12 Controller

Page 114 Figure 27. Bits in CIDMR0-3 registers changed from AC to AM.

Page 115 Figure 28. Bits in CIDMR0-3 registers changed from AC to AM.

Page 116 Figure 29. Bits in CIDMR0-3 registers changed from AC to AM.

Page 118 Warning modified .

Page 119 CIDAR0-3 and CIDARM0-3 changed to CIDAR0-7 and CIDAM0-7.

Sec. 14.7.3. Additional information about access to registers added.

Sec. 14.8. Footnote changed from timer channel....is integration dependent, to timer channel.....is PT0.

Sec.14.12.6. Note added at end of section describing register state when SFTRES bit in CMCR0 is set.

Sec. 14.12.7. Note added at end of section describing register state when SFTRES bit in CMCR0 is set.

Sec. 14.12.8. Note added at end of section describing register state when SFTRES bit in CMCR0 is set.

Sec. 14.12.9. Note added at end of section describing register state when SFTRES bit in CMCR0 is set.

Section 15 Analog-to-Digital Converter

Page 143 “The ATD is an 8-channel, 8-bit .....accurate to ± 1 LSB”, has been changed to; “The ATD is an 8-channel, 10-bit.....accurate to ± 2 least significant bits (LSB)”.

Page 144 Bit 5 in ATDCTL2 changed from ASWAI to AWAI.

Page 145 ATDCTL4 register, information added for calculating conversion time. Table 44 corrected.

Page 146“ Writes to the ATD control registers initiates a new conversion sequence” is changed to “A write to ATDCTL5 initiates a new conversion sequence.”

Section 16 Development Support

Sec. 16.2.1 and Sec. 16.2.2 order swapped.

Sec. 16.2.1 Second paragraph, “256” (E-clock cycles) changed to “512”
Fourth paragraph, “Nine” (target E cycles) changed to “Ten”
Fig. 43, “9” (cycles) changed to “10” and appropriate correction made to diagram

Sec. 16.2.3 Several paragraphs added with additional information about BDM commands.
Table 48, Note 2, “ENTER_TAG_MODE” removed from footnote.

Sec. 16.2.4 Reset information added to INSTRUCTION register.
Reset information corrected for STATUS register.
Additional information added to STATUS register bit descriptions.

Sec. 16.3.1 Corrections and clarifications as to the effect of bits to breakpoint modes.

Sec. 16.3.2 Additonal information added to BKPM bit description.
Additional information added regarding reset condition of registers BRKAH, BRKAL, BRKDH, and BRKDL

Sec. 16.4 Tag Pin Function table added with discussion

NOTES

NOTES

How to reach us:

USA/EUROPE: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140

Mfax: RMFAX0@email.sps.mot.com – TOUCHTONE 1- 602-244-6609. http://sps.motorola.com/mfax

US & CANADA ONLY: http://sps.motorola.com/mfax

HOME PAGE: http://motorola.com/sps/

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

CUSTOMER FOCUS CENTER: 1-800-521-6274