MC68HC912DT128A/D

MC68HC912DT128A
MC68HC912DG128A

Technical Data

Rev 2.0

June 12, 2001

MOTOROLA

Revision History

This section lists the revision history of the document since the first release. Data for previous internal drafts is unavailable.

Changes from Rev 1.0 to Rev 2.0

Section	Page (in Rev 2.0)	Description of change
Pinout and Signal Descriptions	28, 29	Figures 4 and 5, pin 97 changed to TEST and note added.
	31	Text added about connection of power supplies.
	33	Note about non-standard oscillator circuit expanded.
	33	DC bias capacitor added to Figure 8 and notes added to cover DC bias.
	39, 45	Text added in Table 6 and Port CAN pin descriptions about non-connection of TxCAN pins when MSCAN modules are not used.
Registers	53, 58	CD bit name corrected in ATD0CTL5 and ATD1CTL5.
EEPROM	102	New section added 'EEPROM Selective Write More Zeros'
Clock Functions	141 to 158	Major rewrite of Limp-Home and Fast STOP Recovery modes.
	159	System Clock Frequency Formulae updated for clarification.
	160	Figure 18 modified for clarification.
	163	Figure 21 modified for clarification.
Enhanced Capture Timer	192	Figure 28 modified for clarification.
Analog-To-Digital Converter	321	Shading removed from CC bit in Table 54.
	320, 321	CD bit name corrected.

Revision History

Section	Page (in Rev 2.0)	Description of change
Electrical Characteristics	351, 352	Preliminary notes removed.
	359	EEPROM Programming Maximum Time to ‘AUTO’ Bit Set and EEPROM Erasing Maximum Time to ‘AUTO’ Bit Set added to Table 73.
	367	Several values in Table 78 updated.
	359	Changed note after Table 73 and Table 74 to read ‘Based on the average life time operating temperature of 70°C.’
Appendix A: MC68HC912DT128A	375	New section added ‘2d. EEPROM Selective Write More Zeros’. New section added ‘6. Port ADx’. New section added ‘7. ATD’.
Appendix B: CGM Practical Aspects	378	New section added ‘DC Bias’.

Changes from first version (internal release, no revision number) to Rev 1.0

Section	Page (in Rev 1.0)	Description of change
General Description	21	Ordering Information updated.
EEPROM	109	Addition of Caution regarding attempts to erase or program protected locations.
MSI	236	Clarification of SP0DR register state on reset.

List of Sections

List of Sections

Table of Contents

Table of Contents

Table of Contents

Table of Contents

General Description

Contents

Introduction

The MC68HC912DT128A microcontroller unit (MCU) is a 16-bit device composed of standard on-chip peripherals including a 16-bit central processing unit (CPU12), 128K bytes of flash EEPROM, 8K bytes of RAM, 2K bytes of EEPROM, two asynchronous serial communications interfaces (SCI), a serial peripheral interface (SPI), an inter-IC interface (I^2C), an enhanced capture timer (ECT), two 8-channel, 10-bit analog-to-digital converters (ADC), a four-channel pulse-width modulator (PWM), and three CAN 2.0 A, B software compatible modules (MSCAN12). System resource mapping, clock generation, interrupt control and bus interfacing are managed by the lite integration module (LIM). The MC68HC912DT128A has full 16-bit data paths throughout, however, the external bus can operate in an 8-bit narrow mode so single 8-bit wide memory can be interfaced for lower cost systems. The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements. In addition to the I/O ports available in each module, 16 I/O ports are available with Key-Wake-Up capability from STOP or WAIT mode.

The MC68HC912DG128A device is similar to the MC68HC912DT128A, but it has only two MSCAN12 modules. The entire databook applies also to the MC68HC912DG128A, except where differences are noted.

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
- Multiplexed bus
  - Single chip or expanded
  - 16 address/16 data wide or 16 address/8 data narrow modes
- Memory
  - 128K byte flash EEPROM, made of four 32K byte modules with 8K bytes protected BOOT section in each module
  - 2K byte EEPROM
  - 8K byte RAM with Vstby, made of two 4K byte modules.
- Two Analog-to-digital converters
  - 2 times 8-channels, 10-bit resolution
- Three 1M bit per second, CAN 2.0 A, B software compatible modules on the MC68HC912DT128A (two on the MC68HC912DG128A)
  - Two receive and three transmit buffers per CAN
  - 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 per CAN
  - Low-pass filter wake-up function
  - Loop-back for self test operation

  - Programmable link to a timer input capture channel, for time-stamping and network synchronization.
- Enhanced capture timer (ECT)
  - 16-bit main counter with 7-bit prescaler
  - 8 programmable input capture or output compare channels; 4 of the 8 input captures with buffer
  - Input capture filters and buffers, three successive captures on four channels, or two captures on four channels with a capture/compare selectable on the remaining four
  - Four 8-bit or two 16-bit pulse accumulators
  - 16-bit modulus down-counter with 4-bit prescaler
  - Four user-selectable delay counters for signal filtering
- 4 PWM channels with programmable period and duty cycle
  - 8-bit 4-channel or 16-bit 2-channel
  - Separate control for each pulse width and duty cycle
  - Center- or left-aligned outputs
  - Programmable clock select logic with a wide range of frequencies
- Serial interfaces
  - Two asynchronous serial communications interfaces (SCI)
  - Inter IC bus interface (I^2C)
  - Synchronous serial peripheral interface (SPI)
- LIM (lite integration module)
  - WCR (windowed COP watchdog, real time interrupt, clock monitor)
  - ROC (reset and clocks)
  - MEBI (multiplexed external bus interface)
  - MMI (memory map and interface)
  - INT (interrupt control)

  - BKP (breakpoints)
  - BDM (background debug mode)
- Two 8-bit ports with key wake-up interrupt
- Clock generation
  - Phase-locked loop clock frequency multiplier
  - Limp home mode in absence of external clock
  - Slow mode divider
  - Low power 0.5 to 16 MHz crystal oscillator reference clock
- 112-Pin TQFP package
  - Up to 67 general-purpose I/O lines on the MC68HC912DT128A (up to 69 on the MC68HC912DG128A), plus up to 18 input-only lines
  - 5.0V operation at 8 MHz
- Development support
  - Single-wire background debug™ mode (BDM)
  - On-chip hardware breakpoints

MC68HC912DT128A Block Diagram

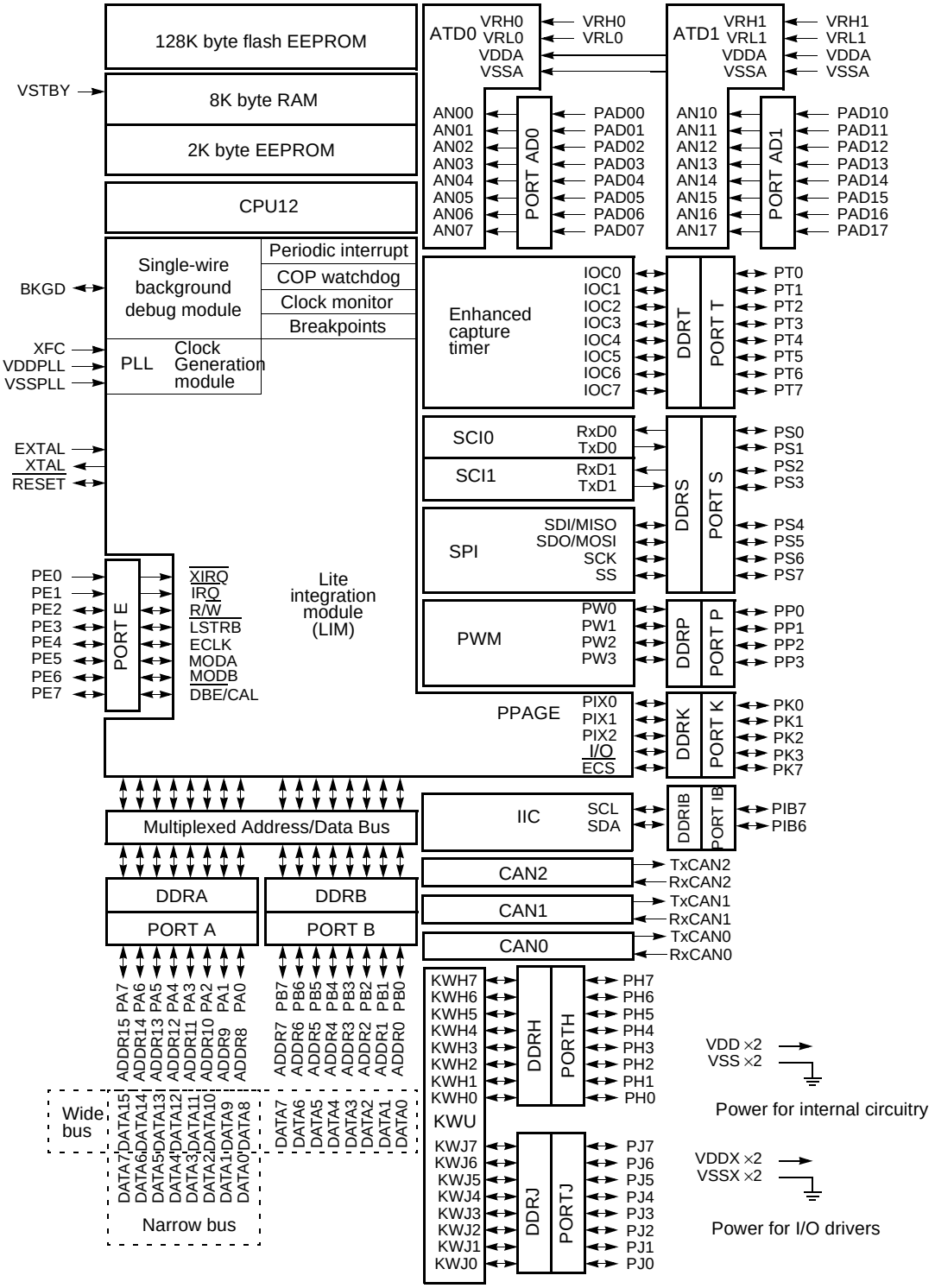


Figure 1 MC68HC912DT128A Block Diagram

MC68HC912DG128A Block Diagram

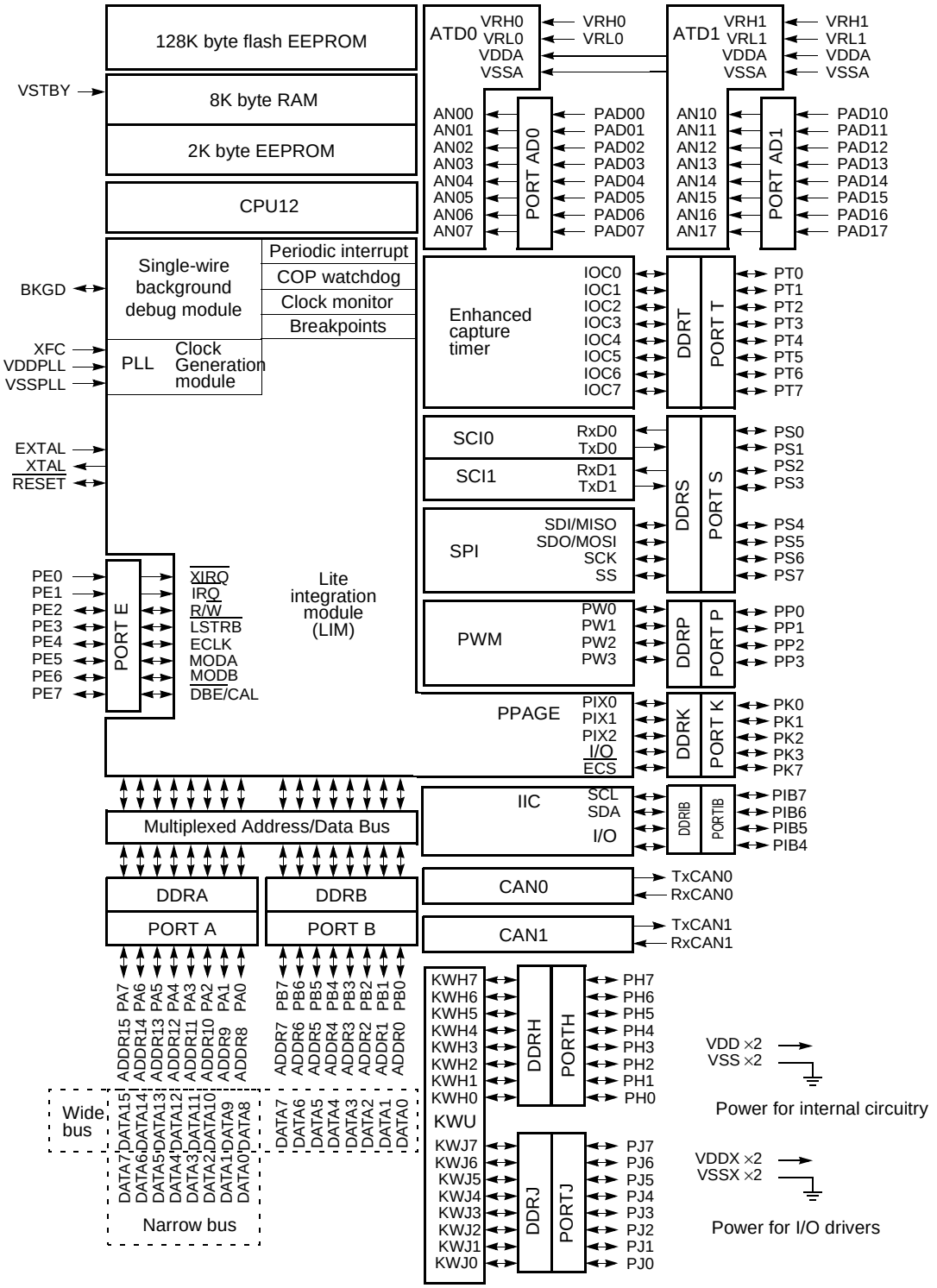


Figure 2 MC68HC912DG128A Block Diagram

Ordering Information

Table 1 Device Ordering Information

Package	Temperature		Voltage	Frequency	Order Number
	Range	Designator			
112-Pin TQFP	–40 to +85°C	C	5V	8 MHz	Consult factory
	–40 to +105°C	V			
	–40 to +125°C	M			

Table 2 Development Tools Ordering Information

Description	Details	Order Number
Evaluation board kit	EVB and user's manual only	M68EVB912DG128
Serial Debug Interface	Low voltage serial debug interface cable can be ordered separately	M68SDIL12
Complete evaluation board kit	EVB, MCUez debug software, SDIL low voltage serial debug interface cable	M68KIT912DG128
Adapter	112 pin TQFP adapter is also available.	M68ADP912DG128PV

General Description

Central Processing Unit

Contents

Introduction

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.

Programming Model

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

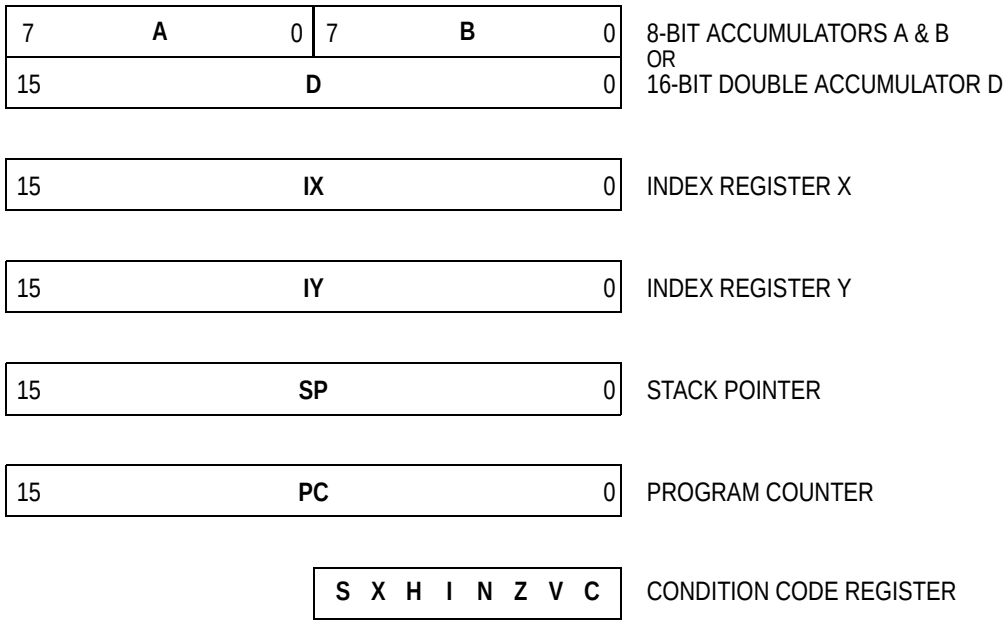


Figure 3 Programming Model

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.

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.

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 3 is a summary of the available addressing modes.

Table 3 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

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 4 Summary of Indexed Operations

Postbyte Code (xb)	Source Code Syntax	Comments
rr0nnnnn	,r n,r –n,r	**5-bit constant offset** n = –16 to +15 rr 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) 1 = 16-bit if z = s = 1, 16-bit offset indexed-indirect (see below) rr can specify X, Y, SP, or PC
111rr011	[n,r]	**16-bit offset indexed-indirect** rr can specify X, Y, SP, or PC
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 rr can specify X, Y, or SP (PC not a valid choice)
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 rr can specify X, Y, SP, or PC
111rr111	[D,r]	**Accumulator D offset indexed-indirect** rr can specify X, Y, SP, or PC

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.

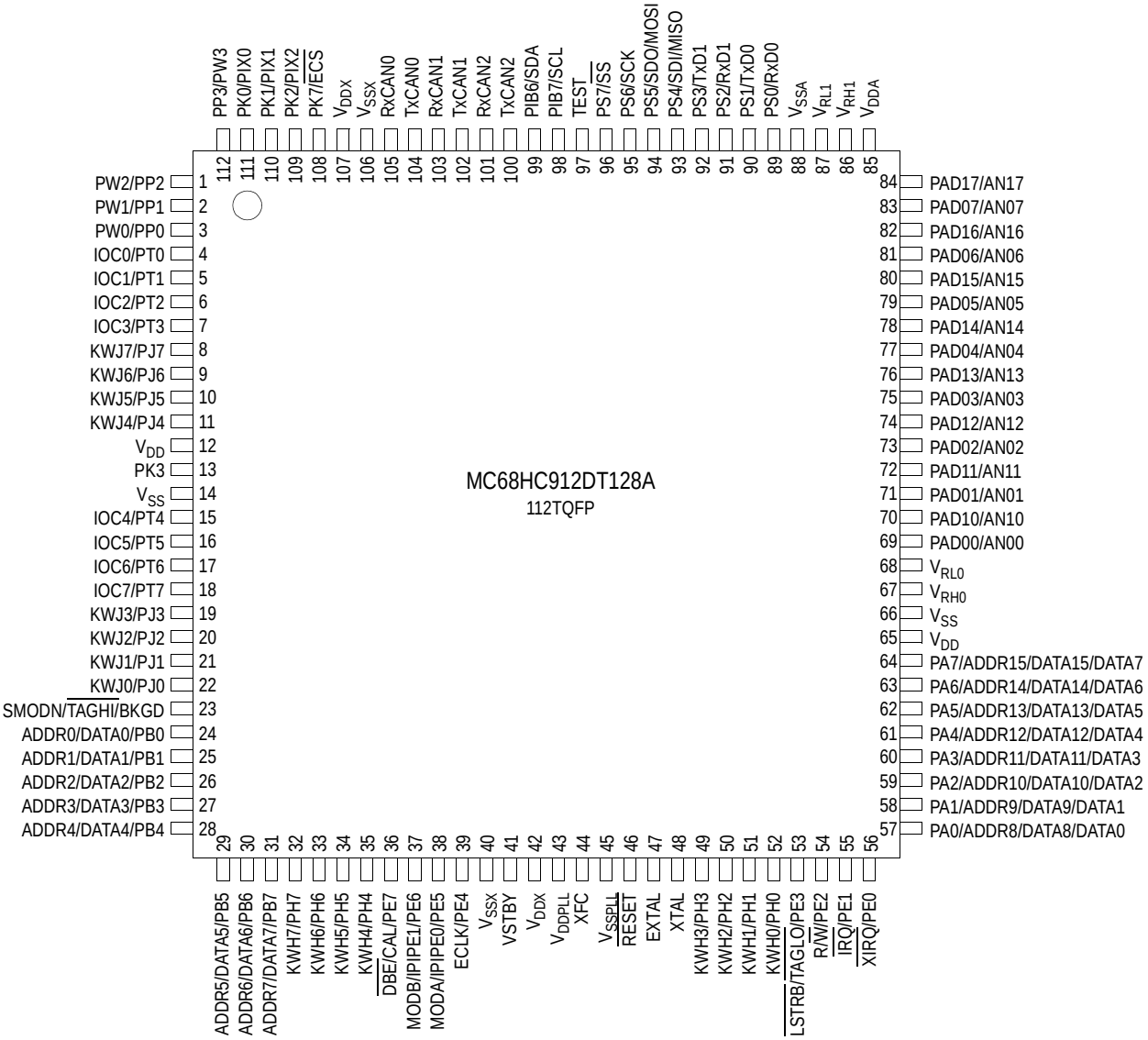
To provide additional addressing flexibility, opcodes can also be followed by a postbyte or extension bytes. Postbytes implement certain forms of indexed addressing, transfers, exchanges, and loop primitives. Extension bytes contain additional program information such as addresses, offsets, and immediate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

Pinout and Signal Descriptions

Contents

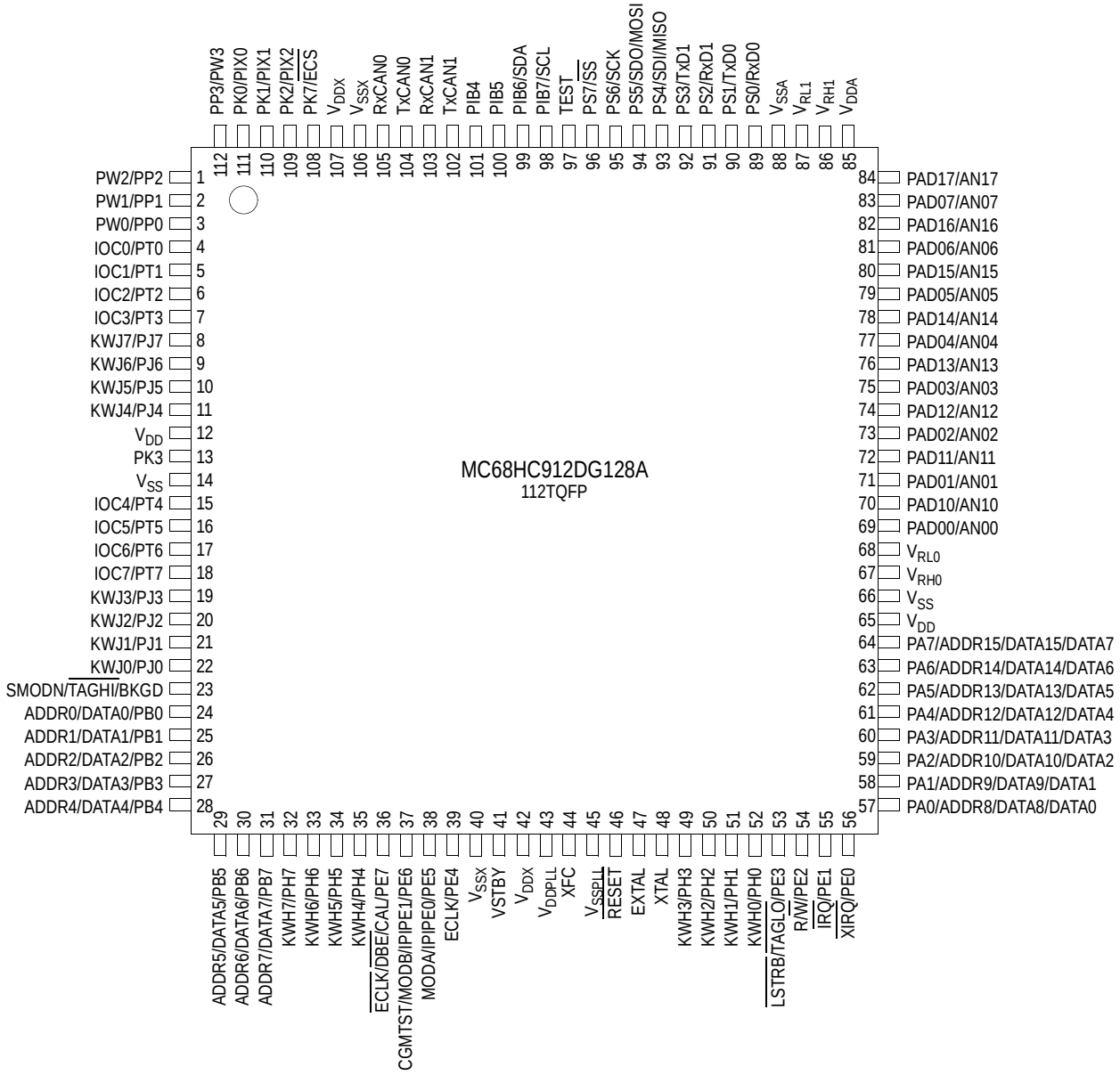
MC68HC912DT128A Pin Assignments in 112-pin QFP

The MC68HC912DT128A is available in a 112-pin thin quad flat pack (TQFP). Most pins perform two or more functions, as described in the Signal Descriptions. Figure 4 shows pin assignments. In expanded narrow modes the lower byte data is multiplexed with higher byte data through pins 57-64.



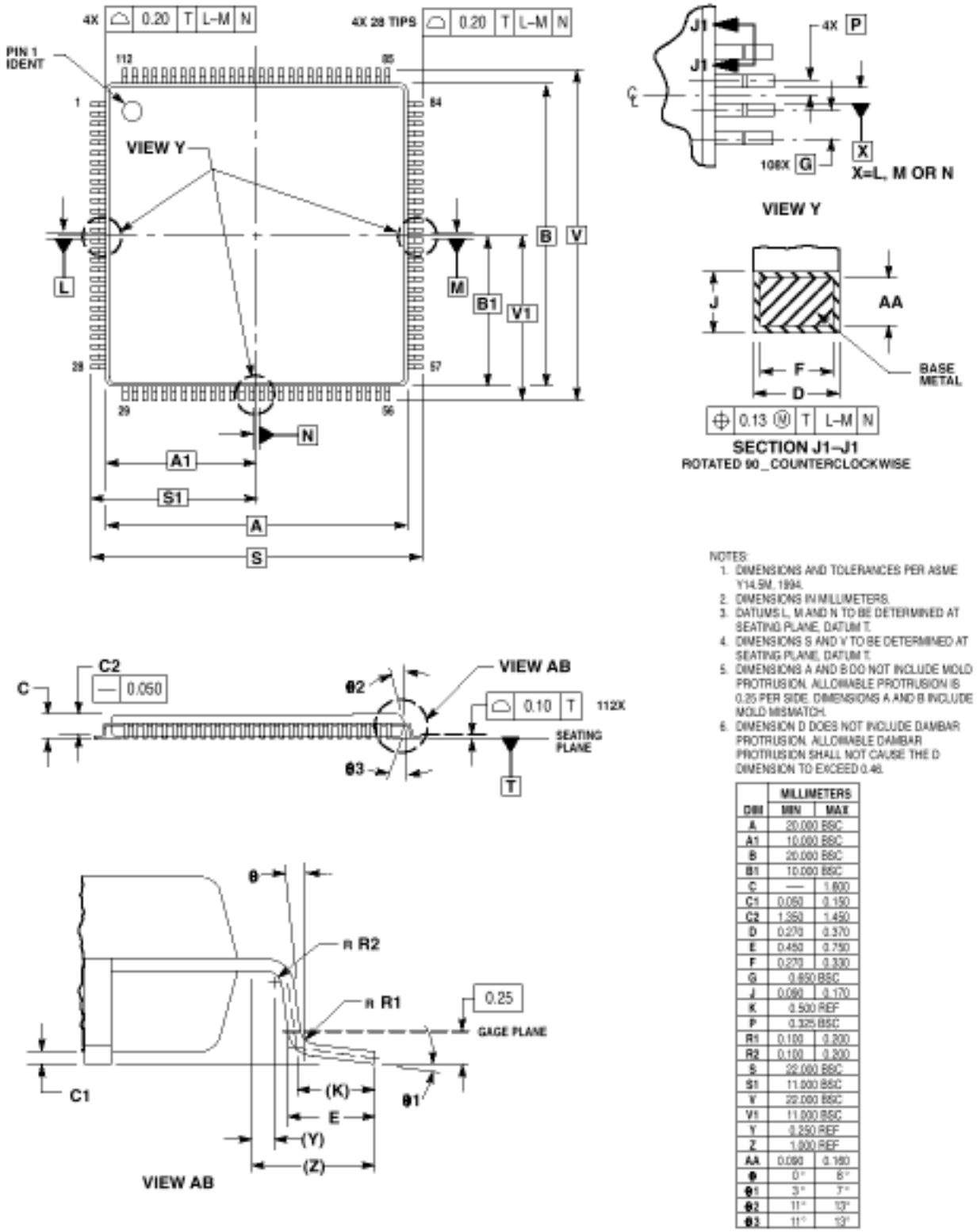
Note: TEST = On early production devices this pin is used for factory test purposes. It is recommended that this pin is not connected within the application, but it may be connected to VSS or 5.5V max without issue. On later production devices this pin is not bonded out.

Figure 4 Pin Assignments in 112-pin QFP for MC68HC912DT128A



Note: TEST = On early production devices this pin is used for factory test purposes. It is recommended that this pin is not connected within the application, but it may be connected to VSS or 5.5V max without issue. On later production devices this pin is not bonded out.

Figure 5 Pin Assignments in 112-pin QFP for MC68HC912DG128A



NOTES:
1. DIMENSIONS AND TOLERANCES PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DATUMS L, M AND N TO BE DETERMINED AT SEATING PLANE, DATUM T.
4. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM T.
5. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B INCLUDE MOLD MISMATCH.
6. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.46.

	MILLIMETERS	
DIM	MIN	MAX
A	20.000 BSC	
A1	10.000 BSC	
B	20.000 BSC	
B1	10.000 BSC	
C	—	1.600
C1	0.050	0.150
C2	1.350	1.450
D	0.270	0.370
E	0.450	0.750
F	0.270	0.330
G	0.650 BSC	
J	0.090	0.170
K	0.500 REF	
P	0.325 BSC	
R1	0.100	0.200
R2	0.100	0.200
S	22.000 BSC	
S1	11.000 BSC	
V	22.000 BSC	
V1	11.000 BSC	
Y	0.250 REF	
Z	1.000 REF	
AA	0.090	0.160
θ	0°	8°
θ1	3°	7°
θ2	11°	13°
θ3	11°	13°

Figure 6 112-pin QFP Mechanical Dimensions (case no. 987)

Power Supply Pins

MC68HC912DT128A power and ground pins are described below and summarized in Table 5.

All power supply pins must be connected to appropriate supplies. On no account must any pins be left floating.

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.

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.

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.

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

V_{RH0}, V_{RL0}: reference voltage high and low for ATD converter 0.

V_{RH1}, V_{RL1}: reference voltage high and low for ATD converter 1.

If the ATD modules are not used, leaving V_{RH} connected to V_{DD} will not result in an increase of power consumption.

V_{DDPLL}, V_{SSPLL}

Provides operating voltage and ground for the Phase-Locked Loop. This allows the supply voltage to the PLL to be bypassed independently.

NOTE: *The VSSPLL pin should always be grounded even if the PLL is not used. The VDDPLL pin should not be left floating. It is recommended to connect the VDDPLL pin to ground if the PLL is not used.*

XFC PLL loop filter. Please see Appendix B: CGM Practical Aspects for information on how to calculate PLL loop filter elements. Any current leakage on this pin must be avoided.

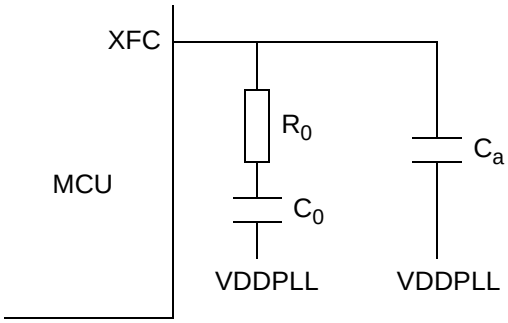


Figure 7 PLL Loop FIlter Connections

V_{STBY} Stand-by voltage supply to static RAM. Used to maintain the contents of RAM with minimal power when the rest of the chip is powered down.

Table 5 MC68HC912DT128A Power and Ground Connection Summary

Mnemonic	Pin Number 112-pin QFP	Description
V_{DD}	12, 65	Internal power and ground.
V_{SS}	14, 66	
V_{DDX}	42, 107	External power and ground, supply to pin drivers.
V_{SSX}	40, 106	
V_{DDA}	85	Operating voltage and ground for the analog-to-digital converter, allows the supply voltage to the A/D to be bypassed independently.
V_{SSA}	88	
V_{RH1}	86	Reference voltages for the analog-to-digital converter 1
V_{RL1}	87	
V_{RH0}	67	Reference voltages for the analog-to-digital converter 0.
V_{RL0}	68	

Table 5 MC68HC912DT128A Power and Ground Connection Summary

Mnemonic	Pin Number 112-pin QFP	Description
V_{DDPLL}	43	Provides operating voltage and ground for the Phase-Locked Loop. This allows the supply voltage to the PLL to be bypassed independently.
V_{SSPLL}	45	
V_{STBY}	41	Stand-by voltage supply to maintain the contents of RAM with minimal power when the rest of the chip is powered down.

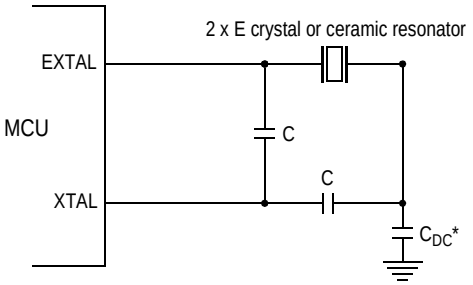
Signal Descriptions

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. Please see Appendix B: CGM Practical Aspects for detailed information on oscillator design.

NOTE: *THE CRYSTAL CIRCUIT FOR COLPITTS OSCILLATOR IS CHANGED FROM THE STANDARD PIERCE OSCILLATOR.*

NOTE: *The internal return path for the oscillator is the VSSPLL pin. Therefore it is recommended to connect the common node of the resonator and the capacitor directly to the VSSPLL pin.*



* Due to the nature of the translated ground Colpitts oscillator a DC voltage bias is applied to the crystal.

Please contact the crystal manufacturer for specific DC bias conditions and recommended capacitance value (if applicable).

Figure 8 Common Crystal Connections

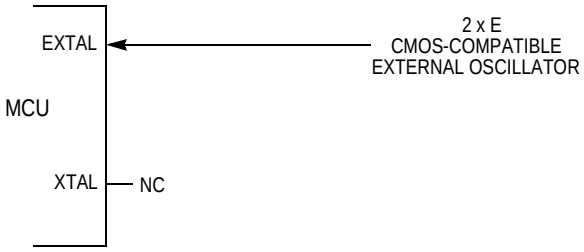


Figure 9 External Oscillator Connections

XTAL is the crystal output.The XTAL pin must be left without terminal 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 in the oscillator circuits include all stray layout capacitances. Refer to Figure 8 and Figure 9 for diagrams of oscillator circuits.

E-Clock Output (ECLK)

ECLK is the output connection for the internal bus clock. It is used to demultiplex the address and data in expanded modes and is used as a timing reference. ECLK frequency is equal to 1/2 the crystal frequency out of reset. The E-clock output is turned off in single chip user mode to reduce the effects of RFI. It can be turned on if necessary. In special single-chip mode, the E-clock is turned ON at reset and 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.

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 important to use an external low-voltage reset circuit (such as MC34064 or MC34164) to prevent corruption of RAM or EEPROM due to power transitions.

The reset sequence is initiated by any of the following events:

- Power-on-reset (POR)
- COP watchdog enabled and watchdog timer times out
- Clock monitor enabled and Clock monitor detects slow or stopped clock
- User applies a low level to the reset pin

External circuitry connected to the reset pin should not include a large capacitance that would interfere with the ability of this signal to rise to a valid logic one within nine bus cycles after the low drive is released. Upon detection of any reset, an internal circuit drives the reset pin low and a clocked reset sequence controls when the MCU can begin normal processing. In the case of POR or a clock monitor error, a 4096 cycle oscillator startup delay is imposed before the reset recovery sequence starts (reset is driven low throughout this 4096 cycle delay). The internal reset recovery sequence then drives reset low for 16 to 17 cycles and releases the drive to allow reset to rise. Nine cycles later this circuit samples the reset pin to see if it has risen to a logic one level. If reset is low at this point, the reset is assumed to be coming from an external request and the internally latched states of the COP timeout and clock monitor failure are cleared so the normal reset vector ($FFFE:FFFF) is taken when reset is finally released. If reset is high after this nine cycle delay, the reset source is tentatively assumed to be either a COP failure or a clock monitor fail. If the internally latched state of the clock monitor fail circuit is true, processing begins by fetching the clock monitor vector ($FFFC:FFFD). If no clock monitor failure is indicated, and the latched state of the COP timeout is true, processing begins by fetching the COP vector ($FFFA:FFFB). If neither clock monitor fail nor COP timeout are pending, processing begins by fetching the normal reset vector ($FFFE:FFFF).

Maskable Interrupt Request (IRQ)

The 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). IRQ is always enabled and configured to level-sensitive triggering at reset. It can be disabled by clearing IRQEN bit (INTCR register). When the MCU is reset the IRQ function is masked in the condition code register. 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.

Nonmaskable Interrupt (XIRQ)

The 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 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. XIRQ is often used as a power loss detect interrupt.

Whenever XIRQ or IRQ are used with multiple interrupt sources (IRQ must be configured for level-sensitive operation if there is more than one source of 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).

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 in expanded modes. MODA and MODB have active pulldowns during reset.

The SMODN pin has an active pullup when configured as an input. The pin can be used as BKGD or TAGHI after reset.

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 an input; BKGD has no pullup control. Refer to **Development Support**.

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

External bus pins share functions 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 pins 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.

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

In all modes this pin can be used as a general-purpose I/O and is an 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.

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

In all modes this pin can be used as a general-purpose I/O and is an 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. Therefore 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.

Instruction Queue Tracking Signals (IPIPE1 and IPIPE0)

IPIPE1 (PE6) and IPIPE0 (PE5) signals are used to track the state of the internal instruction queue. Data movement and execution state information is time-multiplexed on the two signals. Refer to **Development Support**.

Data Bus Enable (DBE)

The $\overline{\text{DBE}}$ pin (PE7) is an active low signal that will be asserted low during E-clock high time. $\overline{\text{DBE}}$ provides separation between output of a multiplexed address and the input of data. When an external address is stretched, $\overline{\text{DBE}}$ 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 $\overline{\text{DBE}}$ is controlled by the NDBE bit in the PEAR register. $\overline{\text{DBE}}$ is enabled out of reset in expanded modes.

Inverted E clock ($\overline{\text{ECLK}}$)

The $\overline{\text{ECLK}}$ pin (PE7) can be used to latch the address for de-multiplexing. It has the same behavior as the ECLK, except is inverted. In expanded modes this pin is used to enable the drive control of external buses during external reads. Use of the $\overline{\text{ECLK}}$ is controlled by the NDBE and DBENE bits in the PEAR register.

Calibration reference (CAL)

The CAL pin (PE7) is the output of the Slow Mode programmable clock divider, SLWCLK, and is used as a calibration reference. The SLWCLK frequency is equal to the crystal frequency out of reset and always has a 50% duty. If the $\overline{\text{DBE}}$ function is enabled it will override the enabled CAL output. The CAL pin output is disabled by clearing CALE bit in the PEAR register.

Clock generation module test(CGMTST)

The CGMTST pin (PE6) is the output of the clocks tested when CGMTE bit is set in PEAR register. The PIPOE bit must be cleared for the clocks to be tested

Table 6 MC68HC912DT128A Signal Description Summary

Pin Name	Shared port	Pin Number	Description
		112-pin	
EXTAL	-	47	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	-	48	
RESET	-	46	An active low bidirectional control signal, 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.
ADDR[7:0] DATA[7:0]	PB[7:0]	31–24	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]	PA[7:0]	64–57	
DBE	PE7	36	Data bus control and, in expanded mode, enables the drive control of external buses during external reads.
ECLK	PE7	36	Inverted E clock used to latch the address.
CAL	PE7	36	CAL is the output of the Slow Mode programmable clock divider, SLWCLK, and is used as a calibration reference for functions such as time of day. It is overridden when DBE function is enabled. It always has a 50% duty.
CGMTST	PE6	37	Clock generation module test output.
MODB/IPIPE1, MODA/IPIPE0	PE6, PE5	37, 38	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.
ECLK	PE4	39	E Clock is the output connection for the external bus clock. ECLK is used as a timing reference and for address demultiplexing.
LSTRB/TAGLO	PE3	53	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 SZ8 signal. (The SZ8 internal signal indicates the size 16/8 access.) Pin function TAGLO used in instruction tagging. See **Development Support**.
R/W	PE2	54	Indicates direction of data on expansion bus. Shares function with general-purpose I/O. Read/write in expanded modes.
IRQ	PE1	55	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).
XIRQ	PE0	56	Provides a means of requesting asynchronous nonmaskable interrupt requests after reset initialization
SMODN/BKGD/ TAGHI	-	23	During reset, this pin determines special or normal operating mode. After reset, single-wire background interface pin is dedicated to the background debug function. Pin function TAGHI used in instruction tagging. See **Development Support**.

Table 6 MC68HC912DT128A Signal Description Summary

Pin Name	Shared port	Pin Number	Description
		112-pin	
IX[2:0]	PK[2:0]	109-111	Page Index register emulation outputs.
ECS	PK7	108	Emulation Chip select.
PW[3:0]	PP[3:0]	112, 1–3	Pulse Width Modulator channel outputs.
SS	PS7	96	Slave select output for SPI master mode, input for slave mode or master mode.
SCK	PS6	95	Serial clock for SPI system.
SDO/MOSI	PS5	94	Master out/slave in pin for serial peripheral interface
SDI/MISO	PS4	93	Master in/slave out pin for serial peripheral interface
TxD1	PS3	92	SCI1 transmit pin
RxD1	PS2	91	SCI1 receive pin
TxD0	PS1	90	SCI0 transmit pin
RxD0	PS0	89	SCI0 receive pin
IOC[7:0]	PT[7:0]	18–15, 7–4	Pins used for input capture and output compare in the timer and pulse accumulator subsystem.
AN1[7:0]	PAD1[7:0]	84/82/80/78/76/74/72/70	Analog inputs for the analog-to-digital conversion module 1
AN0[7:0]	PAD0[7:0]	83/81/79/77/75/73/71/69	Analog inputs for the analog-to-digital conversion module 0
TxCAN2(1)	-	100	MSCAN2 transmit pin (MC68HC912DT128A only). Leave unconnected if MSCAN2 is not used.
RxCAN2(1)	-	101	MSCAN2 receive pin (MC68HC912DT128A only).
TxCAN1	-	102	MSCAN1 transmit pin. Leave unconnected if MSCAN1 is not used.
RxCAN1	-	103	MSCAN1 receive pin.
TxCAN0	-	104	MSCAN0 transmit pin. If the MSCAN is not used, Leave unconnected if MSCAN0 is not used.
RxCAN0	-	105	MSCAN0 receive pin.
SCL	PIB7	98	I^2C bus serial clock line pin
SDA	PIB6	99	I^2C bus serial data line pin
KWJ[7:0]	PJ[7:0]	8–11, 19–22	Key wake-up and general purpose I/O; can cause an interrupt when an input transitions from high to low or from low to high (KWPJ).
KWH[7:0]	PH[7:0]	32–35, 49–52	Key wake-up and general purpose I/O; can cause an interrupt when an input transitions from high to low or from low to high (KWPH).

1 MC68HC912DT128A only

Port Signals

The MC68HC912DT128A incorporates eleven 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 AD0, port AD1, PE[1:0], RxCAN and TxCAN, a data direction register which controls the direction of each pin. After reset all general purpose I/O pins are configured as input.

Port A

Port A pins are used for address and data in expanded modes. When this port is not used for external access such as in single-chip mode, these pins can be used as general purpose I/O. 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. PUCR is not in the address map in peripheral mode.

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 **Bus Control and Input/Output**.

Port B

Port B pins are used for address and data in expanded modes. When this port is not used for external access such as in single-chip mode, these pins can be used as general purpose I/O. 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. PUCR is not in the address map in peripheral mode.

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 **Bus Control and Input/Output**.

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[7,3,2,1,0] are pulled up. PE[7,3,2,0] are active pull-up devices. PUPCR is not in the address map in peripheral mode.

Neither port E nor DDRE is in the map in peripheral mode or 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 **Bus Control and Input/Output**.

Port H

Port H pins are used for key wake-ups that can be used with the pins configured as inputs or outputs. The key wake-ups are triggered with either a rising or falling edge signal (KWPH). An interrupt is generated if the corresponding bit is enabled (KWIEH). If any of the interrupts is not enabled, the corresponding pin can be used as a general purpose I/O pin. Refer to **I/O Ports With Key Wake-Up**.

Register DDRH determines whether each port H pin is an input or output. Setting a bit in DDRH makes the corresponding bit in port H an output; clearing a bit in DDRH makes the corresponding bit in port H an input. The default reset state of DDRH is all zeroes.

Register KWPH not only determines what type of edge the key wake ups are triggered, but it also determines what type of resistive load is used for port H input pins when PUPH bit is set in the PUCR register. Setting a bit in KWPH makes the corresponding key wake up input pin trigger at rising edges and loads a pull down in the corresponding port H input pin. Clearing a bit in KWPH makes the corresponding key wake up input pin trigger at falling edges and loads a pull up in the corresponding port H input pin. The default state of KWPH is all zeroes.

Setting the RDPH bit in register RDRIV causes all port H 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 **Bus Control and Input/Output**.

Port J

Port J pins are used for key wake-ups that can be used with the pins configured as inputs or outputs. The key wake-ups are triggered with either a rising or falling edge signal (KWPJ). An interrupt is generated if the corresponding bit is enabled (KWIEJ). If any of the interrupts is not enabled, the corresponding pin can be used as a general purpose I/O pin. Refer to **I/O Ports With Key Wake-Up**.

Register DDRJ determines whether each port J pin is an input or output. Setting a bit in DDRJ makes the corresponding bit in port J an output;

clearing a bit in DDRJ makes the corresponding bit in port J an input. The default reset state of DDRJ is all zeroes.

Register KWPJ not only determines what type of edge the key wake ups are triggered, but it also determines what type of resistive load is used for port J input pins when PUPJ bit is set in the PUCR register. Setting a bit in KWPJ makes the corresponding key wake up input pin trigger at rising edges and loads a pull down in the corresponding port J input pin. Clearing a bit in KWPJ makes the corresponding key wake up input pin trigger at falling edges and loads a pull up in the corresponding port J input pin. The default state of KWPJ is all zeroes.

Setting the RDPJ bit in register RDRIV causes all port J 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 **Bus Control and Input/Output**.

Port K

Port K pins are used for page index emulation in expanded or peripheral modes. When page index emulation is not enabled, EMK is not set in MODE register, or the part is in single chip mode, these pins can be used for general purpose I/O. Port K bit 3 is used as a general purpose I/O pin only. The port data register is not in the address map during expanded and peripheral mode operation with EMK set. When it is in the map, port K can be read or written at anytime.

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

When the PUPK bit in the PUCR register is set, all port K input pins are pulled-up internally by an active pull-up device. PUCR is not in the address map in peripheral mode.

Setting the RDPK bit in register RDRIV causes all port K 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 **Bus Control and Input/Output**.

Port CAN2 (MC68HC912DT128A only)

The MSCAN2 uses two external pins, one input (RxCAN2) and one output (TxCAN2). The TxCAN2 output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state. RxCAN2 is on bit 0 of Port CAN2, TxCAN2 is on bit 1. If the MSCAN2 is not used, TxCAN2 should be left unconnected.

Port CAN1

The MSCAN1 uses two external pins, one input (RxCAN1) and one output (TxCAN1). The TxCAN1 output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state. RxCAN1 is on bit 0 of Port CAN1, TxCAN1 is on bit 1. If the MSCAN1 is not used, TxCAN1 should be left unconnected.

Port CAN0

The MSCAN0 uses two external pins, one input (RxCAN0) and one output (TxCAN0). The TxCAN0 output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state. RxCAN0 is on bit 0 of Port CAN0, TxCAN0 is on bit 1. If the MSCAN0 is not used, TxCAN0 should be left unconnected.

Port IB

Bidirectional pins to IIC bus interface subsystem. The IIC bus interface uses a Serial Data line (SDA) and Serial Clock line (SCL) for data transfer. The pins are connected to a positive voltage supply via a pull up resistor. The pull ups can be enabled internally or connected externally. The output stages have open drain outputs in order to perform the wired-AND function. When the IIC is disabled the pins can be used as general purpose I/O pins. SCL is on bit 7 of Port IB and SDA is on bit 6. On the MC68HC912DG128A, the remaining two pins of Port IB (PIB5 and PIB4) are controlled by registers in the IIC address space.

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

When the PUPIB bit in the IBPURD register is set, all input pins are pulled up internally by an active pull-up device. Pullups are disabled after reset, except for input ports 0 through 5, which are always on regardless of PUPIB bit.

Setting the RDPIB bit in the IBPURD register configures all port IB outputs to have reduced drive levels. Levels are at normal drive capability after reset. The IBPURD register can be read or written anytime after reset. Refer to section **Inter-IC Bus**.

Port AD1

This port is an analog input interface to the analog-to-digital subsystem and used for general-purpose input. When analog-to-digital functions are not enabled, the port has eight general-purpose input pins, PAD1[7:0]. The ADPU bit in the ATD1CTL2 register enables the A/D function.

Port AD1 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 **Analog-To-Digital Converter (ATD)**.

Port AD0

This port is an analog input interface to the analog-to-digital subsystem and used for general-purpose input. When analog-to-digital functions are not enabled, the port has eight general-purpose input pins, PAD0[7:0]. The ADPU bit in the ATD0CTL2 register enables the A/D function.

Port AD0 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 **Analog-To-Digital Converter (ATD)**.

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 **Pulse-Width Modulator**.

Port S

Port S is the 8-bit interface to the standard serial interface consisting of the two serial communications interfaces (SCI1 and SCI0) and the serial peripheral interface (SPI) subsystems. Port S pins are available for general-purpose 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 SC0CR1register is set, the P-channel drivers of the output buffers are disabled (wire-or mode) for pins 0 through 3. If SWOM bit in the SP0CR1 register is set, the P-channel drivers of the output buffers are disabled (wire-or mode) for pins 4 through 7. The open drain control affects both the serial and the general-purpose outputs. If the RDPS bit in the SP0CR2 register is set, Port S pin drive capabilities are reduced. If PUPS bit in the SP0CR2 register is set, a pull-up device is activated for each port S pin 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 PUPS bit. See **Multiple Serial Interface**.

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 **Enhanced Capture Timer**.

Table 7 MC68HC912DT128A Port Description Summary

Port Name	Pin Numbers 112-pin	Data Direction Register (Address)	Description
Port A PA[7:0]	64-57	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]	31–24	In/Out DDRB ($0003)	
Port AD1 PAD1[7:0]	84/82/80/ 78/76/74/ 72/70	In	Analog-to-digital converter 1 and general-purpose I/O.
Port AD0 PAD0[7:0]	83/81/79/ 77/75/73/ 71/69	In	Analog-to-digital converter 0 and general-purpose I/O.
Port CAN2 PCAN2[1:0] (1)	100–101	PCAN2[1] Out PCAN2[0] In	PCAN2[1:0] are used with the MSCAN2 module and cannot be used as general purpose I/O (MC68HC912DT128A only).
Port CAN1 PCAN1[1:0]	102–103	PCAN1[1] Out PCAN1[0] In	PCAN1[1:0] are used with the MSCAN1 module and cannot be used as general purpose I/O.
Port CAN0 PCAN0[1:0]	104–105	PCAN0[1] Out PCAN0[0] In	PCAN0[1:0] are used with the MSCAN0 module and cannot be used as general purpose I/O.
Port IB PIB[7:6]	98–99	In/Out DDRIB ($00E7)	General purpose I/O. PIB[7:6] are used with the I-Bus module when enabled.
Port IB PIB[5:4](2)	100–101	In/Out DDRIB ($00E7)	General purpose I/O (MC68HC912DG128A only).
Port E PE[7:0]	36–39, 53–56	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 K PK[7,3:0]	13, 108-111	In/Out DDRK ($00FD)	Page index emulation signals in expanded or peripheral mode or general-purpose I/O.

Table 7 MC68HC912DT128A Port Description Summary

Port Name	Pin Numbers	Data Direction Register (Address)	Description
	112-pin		
Port P PP[3:0]	112, 1–3	In/Out DDRP ($0057)	General-purpose I/O. PP[3:0] are used with the pulse-width modulator when enabled.
Port S PS[7:0]	96–89	In/Out DDRS ($00D7)	Serial communications interfaces 1 and 0 and serial peripheral interface subsystems; or general-purpose I/O.
Port T PT[7:0]	18–15, 7–4	In/Out DDRT ($00AF)	General-purpose I/O when not enabled for input capture and output compare in the timer and pulse accumulator subsystem.

1 MC68HC912DT128A only

2 MC68HC912DG128A only

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 1 summarizes the port pull-up default status and controls.

Table 1 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, PE[3:2]	Pull-up	PUCR ($000C)	PUPE	Enabled	RDRIV ($000D)	RDPE	Full drive
PE[1:0]	Pull-up	PUCR ($000C)	PUPE	Enabled	—		
PE[6:4]	None	—			RDRIV ($000D)	RDPE	Full drive
Port H	Pull-up or Pull-down	PUCR ($000C)	PUPH	Disabled	RDRIV ($000D)	RDPH	Full drive
Port J	Pull-up or Pull-down	PUCR ($000C)	PUPJ	Disabled	RDRIV ($000D)	RDPJ	Full drive
Port K	Pull-up	PUCR ($000C)	PUPK	Disabled	RDRIV ($000D)	RDPK	Full drive
Port P	Pull-up	PWCTL ($0054)	PUPP	Disabled	PWCTL ($0054)	RDPP	Full drive
Port S	Pull-up	SP0CR2 ($00D1)	PUPS	Enabled	SP0CR2 ($00D1)	RDPS	Full drive
Port T	Pull-up	TMSK2 ($008D)	TPU	Disabled	TMSK2 ($008D)	TDRB	Full drive

Table 1 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 IB[7:6]	Pull-up	IBPURD ($00E5)	PUPIB	Disabled	IBPURD ($00E5)	RDPIB	Full drive
Port IB[5:4](1)	Pull-up	IBPURD ($00E5)	PUPIB	Disabled	IBPURD ($00E5)	RDPIB	Full drive
Port AD0	None	—			—		
Port AD1	None	—			—		
Port CAN2[1](2)	None	—			—		
Port CAN2[0](2)	Pull-up	Always enabled			—		
Port CAN1[1]	None	—			—		
Port CAN1[0]	Pull-up	Always enabled			—		
Port CAN0[1]	None	—			—		
Port CAN0[0]	Pull-up	Always enabled			—		

1 MC68HC912DG128A only

2 MC68HC912DT128A only

Registers

Contents

Register Block

The register block can be mapped to any 2K byte boundary within the standard 64K byte 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 1K byte of the 2K byte block. Default addressing (after reset) is indicated in the table below. For additional information refer to **Operating Modes**.

Table 8 MC68HC912DT128A Register Map (Sheet 1 of 11)

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-$0007	0	0	0	0	0	0	0	0	Reserved(3)
$0008	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE(2)
$0009	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	0	0	DDRE(2)
$000A	NDBE	CGMTE	PIPOE	NECLK	LSTRE	RDWE	CALE	DBENE	PEAR(3)
$000B	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	EMK	EME	MODE(3)
$000C	PUPK	PUPJ	PUPH	PUPE	0	0	PUPB	PUPA	PUCR(3)
$000D	RDPK	RDPJ	RDPH	RDPE	0	0	RDPB	RDPA	RDRIV(3)
$000E	0	0	0	0	0	0	0	0	Reserved(3)
$000F	0	0	0	0	0	0	0	0	Reserved(3)
$0010	RAM15	RAM14	RAM13	0	0	0	0	0	INITRM
$0011	REG15	REG14	REG13	REG12	REG11	0	0	MMSWAI	INITRG

Table 8 MC68HC912DT128A Register Map (Sheet 2 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0012	EE15	EE14	EE13	EE12	0	0	0	EEON	INITEE
$0013	ROMTST	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	ROMHM	ROMON	MISC
$0014	RTIE	RSWAI	RSBCK	Reserved	RTBYP	RTR2	RTR1	RTR0	RTICTL
$0015	RTIF	0	0	0	0	0	0	0	RTIFLG
$0016	CME	FCME	FCMCOP	WCOP	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	ITB6	ITB8	ITBA	ITBC	ITBE	ITC0	ITC2	ITC4	ITST3
$001C	ITA6	ITA8	ITAA	ITAC	ITAE	ITB0	ITB2	ITB4	ITST4
$001D	0	0	0	0	0	0	0	0	Reserved
$001E	IRQE	IRQEN	DLY	0	0	0	0	0	INTCR
$001F	1	PSEL6	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
$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	0	0	0	0	0	0	0	0	Reserved
$0027	0	0	0	0	0	0	0	0	Reserved
$0028	PJ7	PJ6	PJ5	PJ4	PJ3	PJ2	PJ1	PJ0	PORTJ
$0029	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0	PORTH
$002A	DDJ7	DDJ6	DDJ5	DDJ4	DDJ3	DDJ2	DDJ1	DDJ0	DDRJ
$002B	DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0	DDRH
$002C	KWIEJ7	KWIEJ6	KWIEJ5	KWIEJ4	KWIEJ3	KWIEJ2	KWIEJ1	KWIEJ0	KWIEJ
$002D	KWIEH7	KWIEH6	KWIEH5	KWIEH4	KWIEH3	KWIEH2	KWIEH1	KWIEH0	KWIEH
$002E	KWIFJ7	KWIFJ6	KWIFJ5	KWIFJ4	KWIFJ3	KWIFJ2	KWIFJ1	KWIFJ0	KWIFJ
$002F	KWIFH7	KWIFH6	KWIFH5	KWIFH4	KWIFH3	KWIFH2	KWIFH1	KWIFH0	KWIFH
$0030	KWPJ7	KWPJ6	KWPJ5	KWPJ4	KWPJ3	KWPJ2	KWPJ1	KWPJ0	KWPJ
$0031	KWPH7	KWPH6	KWPH5	KWPH4	KWPH3	KWPH2	KWPH1	KWPH0	KWPH
$0032	0	0	0	0	0	0	0	0	Reserved
$0033	0	0	0	0	0	0	0	0	Reserved
$0034–$0037	Unimplemented(4)								Reserved
$0038	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0	SYNR
$0039	0	0	0	0	0	REFDV2	REFDV1	REFDV0	REFDV
$003A	TSTOUT7	TSTOUT6	TSTOUT5	TSTOUT4	TSTOUT3	TSTOUT2	TSTOUT1	TSTOUT0	CGTFLG
$003B	LOCKIF	LOCK	0	0	0	0	LHIF	LHOME	PLLFLG
$003C	LOCKIE	PLLON	AUTO	ACQ	0	PSTP	LHIE	NOLHM	PLLCR

Table 8 MC68HC912DT128A Register Map (Sheet 3 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$003D	0	BCSP	BCSS	0	0	MCS	0	0	CLKSEL
$003E	0	0	SLDV5	SLDV4	SLDV3	SLDV2	SLDV1	SLDV0	SLOW
$003F	OPNLE	$\overline{\text{TRK}}$	TSTCLKE	TST4	TST3	TST2	TST1	TST0	CGTCTL
$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	Bit 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	RDPP	PUPP	PSBCK	PWCTL
$0055	DISCR	DISCP	DISCAL	0	0	0	0	0	PWTST
$0056	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0	PORTP
$0057	DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0	DDRP
$0058-$005F	0	0	0	0	0	0	0	0	Reserved
$0060	Reserved								ATD0CTL0
$0061	Reserved								ATD0CTL1
$0062	ADPU	AFFC	ASWAI	DJM	DSGN	Reserved	ASCIE	ASCIF	ATD0CTL2
$0063	0	0	0	0	S1C	FIFO	FRZ1	FRZ0	ATD0CTL3
$0064	RES10	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0	ATD0CTL4
$0065	0	S8C	SCAN	MULT	CD	CC	CB	CA	ATD0CTL5
$0066	SCF	0	0	0	0	CC2	CC1	CC0	ATD0STAT0
$0067	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	ATD0STAT1
$0068	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	ATD0TESTH
$0069	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0	ATD0TESTL
$006A–$006E	0	0	0	0	0	0	0	0	Reserved

Table 8 MC68HC912DT128A Register Map (Sheet 4 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$006F	PAD07	PAD06	PAD05	PAD04	PAD03	PAD02	PAD01	PAD00	PORTAD0
$0070	Bit 15	14	13	12	11	10	9	Bit 8	ADR00H
$0071	Bit 7	Bit 6	0	0	0	0	0	0	ADR00L
$0072	Bit 15	14	13	12	11	10	9	Bit 8	ADR01H
$0073	Bit 7	Bit 6	0	0	0	0	0	0	ADR01L
$0074	Bit 15	14	13	12	11	10	9	Bit 8	ADR02H
$0075	Bit 7	Bit 6	0	0	0	0	0	0	ADR02L
$0076	Bit 15	14	13	12	11	10	9	Bit 8	ADR03H
$0077	Bit 7	Bit 6	0	0	0	0	0	0	ADR03L
$0078	Bit 15	14	13	12	11	10	9	Bit 8	ADR04H
$0079	Bit 7	Bit 6	0	0	0	0	0	0	ADR04L
$007A	Bit 15	14	13	12	11	10	9	Bit 8	ADR05H
$007B	Bit 7	Bit 6	0	0	0	0	0	0	ADR05L
$007C	Bit 15	14	13	12	11	10	9	Bit 8	ADR06H
$007D	Bit 7	Bit 6	0	0	0	0	0	0	ADR06L
$007E	Bit 15	14	13	12	11	10	9	Bit 8	ADR07H
$007F	Bit 7	Bit 6	0	0	0	0	0	0	ADR07L
$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
$0085	Bit 7	6	5	4	3	2	1	Bit 0	TCNT
$0086	TEN	TSWAI	TSBCK	TFFCA	Reserved				TSCR
$0087	Reserved								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
$0091	Bit 7	6	5	4	3	2	1	Bit 0	TC0
$0092	Bit 15	14	13	12	11	10	9	Bit 8	TC1
$0093	Bit 7	6	5	4	3	2	1	Bit 0	TC1
$0094	Bit 15	14	13	12	11	10	9	Bit 8	TC2
$0095	Bit 7	6	5	4	3	2	1	Bit 0	TC2
$0096	Bit 15	14	13	12	11	10	9	Bit 8	TC3
$0097	Bit 7	6	5	4	3	2	1	Bit 0	TC3

Table 8 MC68HC912DT128A Register Map (Sheet 5 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0098	Bit 15	14	13	12	11	10	9	Bit 8	TC4
$0099	Bit 7	6	5	4	3	2	1	Bit 0	TC4
$009A	Bit 15	14	13	12	11	10	9	Bit 8	TC5
$009B	Bit 7	6	5	4	3	2	1	Bit 0	TC5
$009C	Bit 15	14	13	12	11	10	9	Bit 8	TC6
$009D	Bit 7	6	5	4	3	2	1	Bit 0	TC6
$009E	Bit 15	14	13	12	11	10	9	Bit 8	TC7
$009F	Bit 7	6	5	4	3	2	1	Bit 0	TC7
$00A0	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI	PACTL
$00A1	0	0	0	0	0	0	PAOVF	PAIF	PAFLG
$00A2	Bit 7	6	5	4	3	2	1	Bit 0	PACN3
$00A3	Bit 7	6	5	4	3	2	1	Bit 0	PACN2
$00A4	Bit 7	6	5	4	3	2	1	Bit 0	PACN1
$00A5	Bit 7	6	5	4	3	2	1	Bit 0	PACN0
$00A6	MCZI	MODMC	RDMCL	ICLAT	FLMC	MCEN	MCPR1	MCPR0	MCCTL
$00A7	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0	MCFLG
$00A8	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN	ICPAR
$00A9	0	0	0	0	0	0	DLY1	DLY0	DLYCT
$00AA	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0	ICOVW
$00AB	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ	ICSYS
$00AC	0	0	0	0	0	0	0	0	Reserved
$00AD	0	0	0	0	0	0	TCBYP	0	TIMTST
$00AE	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0	PORTT
$00AF	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0	DDRT
$00B0	0	PBEN	0	0	0	0	PBOVI	0	PBCTL
$00B1	0	0	0	0	0	0	PBOVF	0	PBFLG
$00B2	Bit 7	6	5	4	3	2	1	Bit 0	PA3H
$00B3	Bit 7	6	5	4	3	2	1	Bit 0	PA2H
$00B4	Bit 7	6	5	4	3	2	1	Bit 0	PA1H
$00B5	Bit 7	6	5	4	3	2	1	Bit 0	PA0H
$00B6	Bit 15	14	13	12	11	10	9	Bit 8	MCCNTH
$00B7	Bit 7	6	5	4	3	2	1	Bit 0	MCCNTL
$00B8	Bit 15	14	13	12	11	10	9	Bit 8	TC0H
$00B9	Bit 7	6	5	4	3	2	1	Bit 0	TC0H
$00BA	Bit 15	14	13	12	11	10	9	Bit 8	TC1H
$00BB	Bit 7	6	5	4	3	2	1	Bit 0	TC1H
$00BC	Bit 15	14	13	12	11	10	9	Bit 8	TC2H
$00BD	Bit 7	6	5	4	3	2	1	Bit 0	TC2H
$00BE	Bit 15	14	13	12	11	10	9	Bit 8	TC3H
$00BF	Bit 7	6	5	4	3	2	1	Bit 0	TC3H
$00C0	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8	SC0BDH

Table 8 MC68HC912DT128A Register Map (Sheet 6 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$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	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8	SC1BDH
$00C9	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0	SC1BDL
$00CA	LOOPS	WOMS	RSRC	M	WAKE	ILT	PE	PT	SC1CR1
$00CB	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SC1CR2
$00CC	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	SC1SR1
$00CD	0	0	0	0	0	0	0	RAF	SC1SR2
$00CE	R8	T8	0	0	0	0	0	0	SC1DRH
$00CF	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SC1DRL
$00D0	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF	SP0CR1
$00D1	0	0	0	0	PUPS	RDPS	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–$00DF	0	0	0	0	0	0	0	0	Reserved
$00E0	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0	IBAD
$00E1	0	0	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0	IBFD
$00E2	IBEN	IBIE	MS/SL	Tx/Rx	TXAK	RSTA	0	IBSWAI	IBCR
$00E3	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK	IBSR
$00E4	D7	D6	D5	D4	D3	D2	D1	D0	IBDR
$00E5	0	0	0	RDPIB	0	0	0	PUPIB	IBPURD
$00E6	PIB7	PIB6	PIB5	PIB4	PIB3	PIB2	PIB1	PIB0	PORTIB
$00E7	DDRIB7	DDRIB6	DDRIB5	DDRIB4	DDRIB3	DDRIB2	DDRIB1	DDRIB0	DDRIB
$00E8–$00EB	Unimplemented(4)								Reserved
$00EC–$00ED	0	0	0	0	0	0	0	0	Reserved
$00EE	0	0	0	0	0	0	EEDIV9	EEDIV8	EEDIVH
$00EF	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0	EEDIVL
$00F0	NOBDML	NOSHW	Reserved		1	EESWAI	PROTLCK	EERC	EEMCR
$00F1	SHPROT	1	BPROT5	BPROT4	BPROT3	BPROT2	BPROT1	BPROT0	EEPROT

Table 8 MC68HC912DT128A Register Map (Sheet 7 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$00F2	0	EREVTN	0	0	0	ETMSD	ETMR	ETMSE	EETST
$00F3	BULKP	0	AUTO	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	STRE	REVTUN	0	0	0	TMSD	TMR	TMSE	FEETST
$00F7	0	0	0	FEESWAI	HVEN	0	ERAS	PGM	FEECTL
$00F8	MT07	MT06	MT05	MT04	MT03	MT02	MT01	MT00	MTST0
$00F9	MT0F	MT0E	MT0D	MT0C	MT0B	MT0A	MT09	MT08	MTST1
$00FA	MT17	MT16	MT15	MT14	MT13	MT12	MT11	MT10	MTST2
$00FB	MT1F	MT1E	MT1D	MT1C	MT1B	MT1A	MT19	MT18	MTST3
$00FC	PK7	0	0	0	PK3	PK2	PK1	PK0	PORTK(5)
$00FD	DDK7	0	0	0	DDK3	DDK2	DDK1	DDK0	DDRK(5)
$00FE	0	0	0	0	0	0	0	0	Reserved
$00FF	0	0	0	0	0	PIX2	PIX1	PIX0	PPAGE
$0100	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	C0MCR0
$0101	0	0	0	0	0	LOOPB	WUPM	CLKSRC	C0MCR1
$0102	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	C0BTR0
$0103	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	C0BTR1
$0104	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	C0RFLG
$0105	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE	C0RIER
$0106	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	C0TFLG
$0107	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	C0TCR
$0108	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	C0IDAC
$0109–$010D	Unimplemented(4)								Reserved
$010E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	C0RXERR
$010F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	C0TXERR
$0110	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR0
$0111	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR1
$0112	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR2
$0113	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR3
$0114	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR0
$0115	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR1
$0116	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR2
$0117	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR3
$0118	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR4
$0119	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR5
$011A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR6
$011B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C0IDAR7
$011C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR4
$011D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR5

Table 8 MC68HC912DT128A Register Map (Sheet 8 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$011E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR6
$011F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C0IDMR7
$0120–$013C	Unimplemented(4)								Reserved
$013D	0	0	0	0	0	0	PUPCAN	RDPCAN	PCTLCAN0
$013E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN	PORTCAN0
$013F	DDCAN7	DDCAN6	DDCAN5	DDCAN4	DDCAN3	DDCAN2	0	0	DDRCAN0
$0140–$014F	FOREGROUND RECEIVE BUFFER 0								RxFG0
$0150–$015F	TRANSMIT BUFFER 00								Tx00
$0160–$016F	TRANSMIT BUFFER 01								Tx01
$0170–$017F	TRANSMIT BUFFER 02								Tx02
$0180–$01DF	Unimplemented(4)								Reserved
$01E0	Reserved								ATD1CTL0
$01E1	Reserved								ATD1CTL1
$01E2	ADPU	AFFC	ASWAI	DJM	DSGN	Reserved	ASCIE	ASCIF	ATD1CTL2
$01E3	0	0	0	0	S1C	FIFO	FRZ1	FRZ0	ATD1CTL3
$01E4	RES10	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0	ATD1CTL4
$01E5	0	S8C	SCAN	MULT	CD	CC	CB	CA	ATD1CTL5
$01E6	SCF	0	0	0	0	CC2	CC1	CC0	ATD1STAT0
$01E7	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	ATD1STAT1
$01E8	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	ATD1TESTH
$01E9	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0	ATD1TESTL
$01EA–$01EE	0	0	0	0	0	0	0	0	Reserved
$01EF	PAD17	PAD16	PAD15	PAD14	PAD13	PAD12	PAD11	PAD10	PORTAD1
$01F0	Bit 15	14	13	12	11	10	9	Bit 8	ADR10H
$01F1	Bit 7	Bit 6	0	0	0	0	0	0	ADR10L
$01F2	Bit 15	14	13	12	11	10	9	Bit 8	ADR11H
$01F3	Bit 7	Bit 6	0	0	0	0	0	0	ADR11L
$01F4	Bit 15	14	13	12	11	10	9	Bit 8	ADR12H
$01F5	Bit 7	Bit 6	0	0	0	0	0	0	ADR12L
$01F6	Bit 15	14	13	12	11	10	9	Bit 8	ADR13H
$01F7	Bit 7	Bit 6	0	0	0	0	0	0	ADR13L
$01F8	Bit 15	14	13	12	11	10	9	Bit 8	ADR14H
$01F9	Bit 7	Bit 6	0	0	0	0	0	0	ADR14L
$01FA	Bit 15	14	13	12	11	10	9	Bit 8	ADR15H
$01FB	Bit 7	Bit 6	0	0	0	0	0	0	ADR15L

Table 8 MC68HC912DT128A Register Map (Sheet 9 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$01FC	Bit 15	14	13	12	11	10	9	Bit 8	ADR16H
$01FD	Bit 7	Bit 6	0	0	0	0	0	0	ADR16L
$01FE	Bit 15	14	13	12	11	10	9	Bit 8	ADR17H
$01FF	Bit 7	Bit 6	0	0	0	0	0	0	ADR17L
(6)$0200	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	C2MCR0
(6)$0201	0	0	0	0	0	LOOPB	WUPM	CLKSRC	C2MCR1
(6)$0202	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	C2BTR0
(6)$0203	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	C2BTR1
(6)$0204	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	C2RFLG
(6)$0205	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE	C2RIER
(6)$0206	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	C2TFLG
(6)$0207	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	C2TCR
(6)$0208	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	C2IDAC
$0209–$020D	Unimplemented(4)								Reserved
(6)$020E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	C2RXERR
(6)$020F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	C2TXERR
(6)$0210	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR0
(6)$0211	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR1
(6)$0212	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR2
(6)$0213	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR3
(6)$0214	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR0
(6)$0215	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR1
(6)$0216	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR2
(6)$0217	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR3
(6)$0218	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR4
(6)$0219	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR5
(6)$021A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR6
(6)$021B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C2IDAR7
(6)$021C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR4
(6)$021D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR5
(6)$021E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR6
(6)$021F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C2IDMR7
$0220–$023C	Unimplemented(4)								Reserved
(6)$023D	0	0	0	0	0	0	PUPCAN	RDPCAN	PCTLCAN2
(6)$023E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN	PORTCAN2
(6)$023F	DDCAN7	DDCAN6	DDCAN5	DDCAN4	DDCAN3	DDCAN2	0	0	DDRCAN2
(6)$0240 – $024F	FOREGROUND RECEIVE BUFFER 2								RxFG2
(6)$0250 – $025F	TRANSMIT BUFFER 20								Tx20

Table 8 MC68HC912DT128A Register Map (Sheet 10 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
[6]$0260 – $026F	TRANSMIT BUFFER 21								Tx21
[6]$0270 – $027F	TRANSMIT BUFFER 22								Tx22
$0280-$02FF	Unimplemented[4]								Reserved
$0300	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	C1MCR0
$0301	0	0	0	0	0	LOOPB	WUPM	CLKSRC	C1MCR1
$0302	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	C1BTR0
$0303	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	C1BTR1
$0304	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	C1RFLG
$0305	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE	C1RIER
$0306	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	C1TFLG
$0307	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	C1TCR
$0308	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	C1IDAC
$0309–$030D	Unimplemented[4]								Reserved
$030E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	C1RXERR
$030F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	C1TXERR
$0310	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR0
$0311	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR1
$0312	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR2
$0313	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR3
$0314	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR0
$0315	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR1
$0316	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR2
$0317	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR3
$0318	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR4
$0319	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR5
$031A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR6
$031B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	C1IDAR7
$031C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR4
$031D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR5
$031E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR6
$031F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	C1IDMR7
$0320–$033C	Unimplemented[4]								Reserved
$033D	0	0	0	0	0	0	PUPCAN	RDPCAN	PCTLCAN1
$033E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN	PORTCAN1
$033F	DDCAN7	DDCAN6	DDCAN5	DDCAN4	DDCAN3	DDCAN2	0	0	DDRCAN1
$0340–$034F	FOREGROUND RECEIVE BUFFER 1								RxFG1

Table 8 MC68HC912DT128A Register Map (Sheet 11 of 11)

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$0350–$035F	TRANSMIT BUFFER 10								Tx10
$0360–$036F	TRANSMIT BUFFER 11								Tx11
$0370–$037F	TRANSMIT BUFFER 12								Tx12
$0380-$03FF	Unimplemented(4)								Reserved

= Reserved or unimplemented bits.

1. Port A, port B and data direction registers DDRA, DDRB are not in map in expanded and peripheral modes.
2. Port E and DDRE not in the map in peripheral and expanded modes with EME set.
3. Registers also not in map in peripheral mode.
4. Data read at these locations is undefined.
5. Port K and DDRK not in the map in peripheral and expanded modes with EMK set.
6. MC68HC912DT128A only. Locations are unimplemented on the MC68HC912DG128A.

Operating Modes

Contents

Introduction

Eight possible operating modes determine the operating configuration of the MC68HC912DT128A. Each mode has an associated default memory map and external bus configuration.

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.

Table 9 Mode Selection

BKGD	MODB	MODA	Mode	Port A	Port B
1	0	0	Normal Single Chip	G.P. I/O	G.P. I/O
1	0	1	Normal Expanded Narrow	ADDR/DATA	ADDR
1	1	0	Reserved (Forced to Peripheral)	—	—
1	1	1	Normal Expanded Wide	ADDR/DATA	ADDR/DATA
0	0	0	Special Single Chip	G.P. I/O	G.P. I/O
0	0	1	Special Expanded Narrow	ADDR/DATA	ADDR

Table 9 Mode Selection

BKGD	MODB	MODA	Mode	Port A	Port B
0	1	0	Special Peripheral	ADDR/DATA	ADDR/DATA
0	1	1	Special 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.

Normal Operating Modes

These modes provide three operating configurations. Background debug is available in all three modes, but must first be enabled for some operations by means of a BDM background command, then activated.

Normal Single-Chip Mode — There are no external address and data buses in this mode. The MCU operates as a stand-alone device and all program and data resources are on-chip. External port pins normally associated with address and data buses can be used for general-purpose I/O.

Normal Expanded Wide Mode — This is a normal mode of operation in which the expanded bus is present with a 16-bit data bus. Ports A and B are used for the 16-bit multiplexed address/data bus.

Normal Expanded Narrow Mode — This is a normal mode of operation in which the expanded bus is present with an 8-bit data bus. Ports A and B are used for the16-bit address bus. Port A is used as the data bus, multiplexed with addresses. In this mode, 16-bit data is presented one byte at a time, the high byte followed by the low byte. The address is automatically incremented on the second cycle.

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 Single-Chip Mode — This mode can be used to force the MCU to active BDM mode to allow system debug 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.

S**pecial Expanded Wide Mode** — This mode can be used for emulation of normal expanded wide mode and emulation of normal single-chip mode. Ports A and B are used for the 16-bit multiplexed address/data bus.

Special Expanded Narrow Mode — This mode can be used for emulation of normal expanded narrow mode. Ports A and B are used for the16-bit address bus. Port A is used as the data bus, multiplexed with addresses. In this mode, 16-bit data is presented one byte at a time, the high byte followed by the low byte. The address is automatically incremented on the second cycle.

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 functions.

MODE — Mode Register **$000B**

	Bit 7	6	5	4	3	2	1	Bit 0	
	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	EMK	EME	
RESET:	0	0	0	1	1	0	1	1	Special Single Chip
RESET:	0	0	1	1	1	0	1	1	Special Exp Nar
RESET:	0	1	0	1	1	0	1	1	Peripheral

MODE — Mode Register **$000B**

RESET:	0	1	1	1	1	0	1	1	Special Exp Wide
RESET:	1	0	0	1	0	0	0	0	Normal Single Chip
RESET:	1	0	1	1	0	0	0	0	Normal Exp Nar
RESET:	1	1	1	1	0	0	0	0	Normal Exp Wide

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

SMODN, MODB, MODA — 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 but the first write is ignored and in 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 and data de-multiplexing 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 (IVIS = 0).

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 external bus during accesses to internal locations. In Special Narrow Mode if this bit is set and an internal access occurs the data will appear wide on Ports A and B. This serves

the same function as the EMD bit of the non-multiplexed versions of the HC12 and allows for emulation. Visibility is not available when the part is operating in a single-chip mode.

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 — Multiplexed External Bus Interface Module Stops in Wait Mode

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

0 = MEBI continues functioning during wait mode.
1 = MEBI is shut down during wait mode.

Normal modes: write anytime; special modes: write never. Read anytime.

EMK — Emulate Port K

In single-chip mode PORTK and DDRK are always in the map regardless of the state of this bit.

0 = Port K and DDRK registers are in the memory map. Memory expansion emulation is disabled and all pins are general purpose I/O.
1 = In expanded or peripheral mode, PORTK and DDRK are removed from the internal memory map. Removing the registers from the map allows the user to emulate the function of these registers externally.

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

EME — Emulate Port E

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 = If in an expanded mode, PORTE and DDRE are removed from the internal memory map. Removing the registers from the map allows the user to emulate the function of these registers externally.

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

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 $FF20 to $FFFF, and 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.

Resource Mapping

Contents

Introduction

After reset, most system resources can be mapped to other addresses by writing to the appropriate control registers.

Internal Resource Mapping

The internal register block, RAM, and EEPROM have default locations within the 64K byte 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 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. The following table shows resource mapping precedence.

Table 10 Mapping Precedence

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

In expanded modes, all address space not used by internal resources is by default external memory.

The MC68HC912DT128A contains 128K bytes of Flash EEPROM nonvolatile memory which can be used to store program code or static data. This physical memory is composed of four 32k byte array modules, 00FEE32K, 01FEE32K, 10FEE32K and 11FEE32K. The 32K byte array 11FEE32K has a fixed location from $4000 to $7FFF and $C000 to $FFFF. The three 32K byte arrays 00FEE32K, 01FEE32K and 10FEE32K are accessible through a 16K byte program page window mapped from $8000 to $BFFF. The fixed 32K byte array 11FEE32K can also be accessed through the program page window.

Register Block Mapping

After reset the 1K byte register block resides at location $0000 but can be reassigned to any 2K byte boundary within the standard 64K byte address space. Mapping of internal registers is controlled by five bits in the INITRG register. The register block occupies the first 1K byte of the 2K byte 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.

Normal modes: write once; special modes: write anytime. Read anytime.

MMSWAI — Memory Mapping Interface Stop in Wait Control

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

Normal modes: write anytime; special modes: write never. Read anytime.

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

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

RAM Mapping

The MC68HC912DT128A has 8K bytes of fully static RAM that is used for storing instructions, variables, and temporary data during program execution. Since the RAM is actually implemented with two 4K RAM arrays, any misaligned word access between last address of first 4K RAM and first address of second 4K RAM will take two cycles instead of one. After reset, RAM addressing begins at location $2000 but can be assigned to any 8K byte boundary within the standard 64K byte address space. Mapping of internal RAM is controlled by three bits in the INITRM register.

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

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

RAM[15:13] — Internal RAM map position

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

Normal modes: write once; special modes: write anytime. Read anytime.

EEPROM Mapping

The MC68HC912DT128A has 2K 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 $0800 but can be mapped to any 4K byte boundary within the standard 64K byte address space. The EEPROM block occupies the last 2K bytes of the 4K byte block.

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.

Normal modes: write once; special modes: write anytime. Read anytime.

EEON — internal EEPROM On (Enabled)

Read or write anytime.

0 = Removes the EEPROM from the map.

1 = Places the on-chip EEPROM in the memory map at the address selected by EE[15:12].

Flash EEPROM mapping through internal Memory Expansion

The Page Index register or PPAGE provides memory management for the MC68HC912DT128A. PPAGE consists of three bits to indicate which physical location is active within the windows of the MC68HC912DT128A. The MC68HC912DT128A has a user's program

space window, a register space window for Flash module registers, and a test program space window.

The user's program page window consists of 16K Flash EEPROM bytes. One of eight pages is viewed through this window for a total of 128K accessible Flash EEPROM bytes.

The register space window consists of a 4-byte register block. One of four pages is viewed through this window for each of the 32K flash module register blocks of MC68HC912DT128A.

The test mode program page window consists of 32K Flash EEPROM bytes. One of the four 32K byte arrays is viewed through this window for a total 128K accessible Flash EEPROM bytes. This window is only available in special mode for test purposes and replaces the user's program page window.

MC68HC912DT128A has a five pin port, port K, for emulation and for general purpose I/O. Three pins are used to emulate the three page indices (PPAGE bits) and one pin is used as an emulation chip select. When these four pins are not used for emulation they serve as general purpose I/O pins. The fifth port K pin is used as a general purpose I/O pin.

Program space expansion

There are 128K bytes of Flash EEPROM for MC68HC912DT128A. With a 64K byte address space, the PPAGE register is needed to perform on chip memory expansion. A program space window of 16K byte pages is located from $8000 to $BFFF. Three page indices are used to point to one of eight different 16K byte pages.

Table 11 Program space Page Index

Page Index 2 (PPAGE bit 2)	Page Index 1 (PPAGE bit 1)	Page Index 0 (PPAGE bit 0)	16K Program space Page	Flash array
0	0	0	16K byte Page 0	00FEE32K
0	0	1	16K byte Page 1	00FEE32K
0	1	0	16K byte Page 2	01FEE32K
0	1	1	16K byte Page 3	01FEE32K
1	0	0	16K byte Page 4	10FEE32K
1	0	1	16K byte Page 5	10FEE32K

Table 11 Program space Page Index

Page Index 2 (PPAGE bit 2)	Page Index 1 (PPAGE bit 1)	Page Index 0 (PPAGE bit 0)	16K Program space Page	Flash array
1	1	0	16K byte Page 6*	11FEE32K
1	1	1	16K byte Page 7*	11FEE32K

* The 16K byte flash in program space page 6 can also be accessed at a fixed location from $4000 to $7FFF. The 16K byte flash in program space page 7 can also be accessed at a fixed location from $C000 to $FFFF.

Flash register space expansion

There are four 32K Flash arrays for MC68HC912DT128A and each requires a 4-byte register block. A register space window is used to access one of the four 4-byte blocks and the PPAGE register to map each one into the window. The register space window is located from $00F4 to $00F7 after reset. Only two page indices are used to point to one of the four pages of the register space.

Table 12 Flash Register space Page Index

Page Index 2 (PPAGE bit 2)	Page Index 1 (PPAGE bit 1)	Page Index 0 (PPAGE bit 0)	Flash register space Page	Flash array
0	0	X	$00F4-$00F7 Page 0	00FEE32K
0	1	X	$00F4-$00F7 Page 1	01FEE32K
1	0	X	$00F4-$00F7 Page 2	10FEE32K
1	1	X	$00F4-$00F7 Page 3	11FEE32K

Test mode Program space expansion

In special mode and for test purposes only, the 128K bytes of Flash EEPROM for MC68HC912DT128A can be accessed through a test program space window of 32K bytes. This window replaces the user's program space window to be able to access an entire array. In special mode and with ROMTST bit set in MISC register, a program space is located from $8000 to $FFFF. Only two page indices are used to point to one of the four 32K byte arrays. These indices can be viewed as expanded addresses X16 and X15.

Table 13 Test mode program space Page Index

Page Index 2 (PPAGE bit 2)	Page Index 1 (PPAGE bit 1)	Page Index 0 (PPAGE bit 0)	Flash register space Page	Flash array
0	0	X	32K byte array Page 0	00FEE32K
0	1	X	32K byte array Page 1	01FEE32K
1	0	X	32K byte array Page 2	10FEE32K
1	1	X	32K byte array Page 3	11FEE32K

Page Index register descriptions

PORTK — Port K Data Register **$00FC**

	Bit 7	6	5	4	3	2	1	Bit 0
PORT	PK7	0	0	0	PK3	PK2	PK1	PK0
Emulation	$\overline{ECS}$	0	0	0	-	PIX2	PIX1	PIX0
RESET:	-	0	0	0	-	-	-	-

Read and write anytime

Writes do not change pin state when pin configured for page index emulation output.

This port is associated with page index emulation pins. When the port is not enabled to emulate page index, the port pins are used as general-purpose I/O. Port K bit 3 is used as a general purpose I/O pin only. This register is not in the map in peripheral or expanded modes while the EMK control bit in MODE register is set.

When inputs, these pins can be selected to be high impedance or pulled up.

$\overline{ECS}$ — Emulation Chip Select of selected program space

When this signal is active low it indicates that the program space is accessed. This also applies to test mode program space. An access is made if address is at the program space window and either the Flash or external memory is accessed. The $\overline{ECS}$ timing is E clock high

and can be stretched when accessing external memory depending on the EXTR0 and EXTR1 bits in the MISC register. The $\overline{ECS}$ signal is only active when the EMK bit is set.

PIX[2:0] — The content of the PPAGE register emulated externally.

This content indicates which Flash module register space is in the memory map and which 16K byte Flash memory is in the program space. In special mode and with ROMTST bit set, the content of the Page Index register indicates which 32K byte Flash array is in the test program space.

I

DDRK — Port K Data Direction Register **$00FD**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDK7	0	0	0	DDK3	DDK2	DDK1	DDK0
RESET:	0	0	0	0	0	0	0	0

Read and write: anytime.

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

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

This register is not in the map in peripheral or expanded modes while the EMK control bit is set.

PPAGE — (Program) Page Index Register **$00FF**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	PIX2	PIX1	PIX0
RESET:	0	0	0	0	0	0	0	0

Read and write: anytime.

This register determines the active page viewed through MC68HC912DT128A windows.

CALL and RTC instructions have a special single wire mechanism to read and write this register without using an address bus.

Miscellaneous System Control Register

Additional mapping and external resource controls are available. To use external resources 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	Mode
	ROMTST	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	ROMHM	ROMON	
RESET:	0	0	0	0	1	1	0	0	Exp mode
RESET:	0	0	0	0	1	1	0	1	peripheral or SC mode

Normal modes: write once; Special modes: write anytime. Read anytime.

ROMTST — FLASH EEPROM Test mode

In normal modes, this bit is forced to zero.

0 = 16K window for Flash memory is located from $8000-$BFFF

1 = 32K window for Flash memory is located from $8000-$FFFF

NDRF — Narrow Data Bus for Register-Following Map Space

This bit enables a narrow bus feature for the 1K byte Register-Following Map. This is useful for accessing 8-bit peripherals and allows 8-bit and 16-bit external memory devices to be mixed in a system. In Expanded Narrow (eight bit) modes, Single Chip Modes, and Peripheral mode, this bit has no effect.

0 = Makes Register-Following MAP space act as a full 16 bit data bus.

1 = Makes the Register-Following MAP space act the same as an 8 bit only external data bus (data only goes through port A externally).

The Register-Following space is mapped from $0400 to $07FF after reset, which is next to the register map. If the registers are moved this space follows.

RFSTR1, RFSTR0 — Register Following Stretch

This two bit field determines the amount of clock stretch on accesses to the 1K 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 14 RFSTR Stretch Bit Definition

RFSTR1	RFSTR0	Number of E Clocks Stretched
0	0	0
0	1	1
1	0	2
1	1	3

EXSTR1, EXSTR0 — External Access Stretch

This two bit field determines the amount of clock stretch on accesses to an external address space. In Single Chip and Peripheral Modes this bit has no meaning or effect.

Table 15 EXSTR Stretch Bit Definition

EXSTR1	EXSTR0	Number of E Clocks Stretched
0	0	0
0	1	1
1	0	2
1	1	3

ROMHM — Flash EEPROM only in second Half of Map

This bit has no meaning if ROMON bit is clear.

0 = The 16K byte of fixed Flash EEPROM in location $4000-$7FFF can be accessed.

1 = Disables direct access to 16K byte Flash EEPROM from $4000-$7FFF in the memory map. The physical location of this16K byte Flash can still be accessed through the Program Page window.

In special mode and with ROMTST bit set, this bit will allow overlap of the four 32K Flash arrays and overlap the four 4-byte Flash register space in the same map space to be able to program all arrays at the same time.

0 = The four 32K Flash arrays are accessed with four pages for each.

1 = The four 32K Flash arrays coincide in the same space and are selected at the same time for programming.

CAUTION: *Bit must be cleared before reading any of the arrays or registers.*

ROMON — Enable Flash EEPROM

These bits are used to enable the Flash EEPROM arrays.

0 = Disables Flash EEPROM in the memory map.

1 = Enables Flash EEPROM in the memory map.

Mapping test registers

These registers are used in for testing the mapping logic. They can only be read and after each read they get cleared. A write to each register will have no effect.

MTST0 — Mapping Test Register 0 **$00F8**

	Bit 7	6	5	4	3	2	1	Bit 0
	MT07	MT06	MT05	MT04	MT03	MT02	MT01	MT00
RESET:	0	0	0	0	0	0	0	0

MTST1 — Mapping Test Register 1 **$00F9**

	Bit 7	6	5	4	3	2	1	Bit 0
	MT0F	MT0E	MT0D	MT0C	MT0B	MT0A	MT09	MT08
RESET:	0	0	0	0	0	0	0	0

MTST2 — Mapping Test Register 2 **$00FA**

	Bit 7	6	5	4	3	2	1	Bit 0
	MT17	MT16	MT15	MT14	MT13	MT12	MT11	MT10
RESET:	0	0	0	0	0	0	0	0

MTST3 — Mapping Test Register 3 **$00FB**

	Bit 7	6	5	4	3	2	1	Bit 0
	MT1F	MT1E	MT1D	MT1C	MT1B	MT1A	MT19	MT18
RESET:	0	0	0	0	0	0	0	0

Memory Maps

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

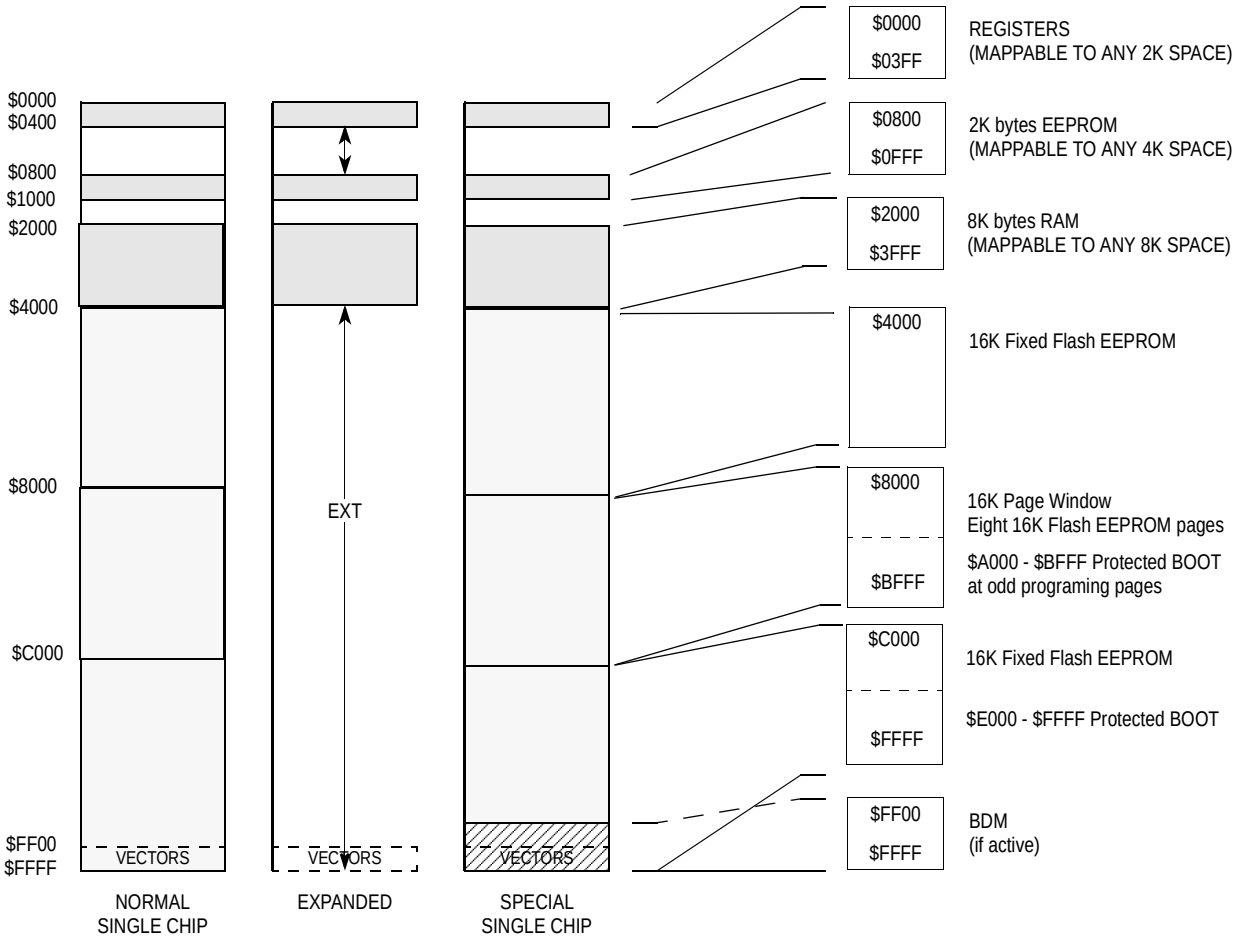


Figure 10 MC68HC912DT128A Memory Map after reset

The following diagram illustrates the memory paging scheme.

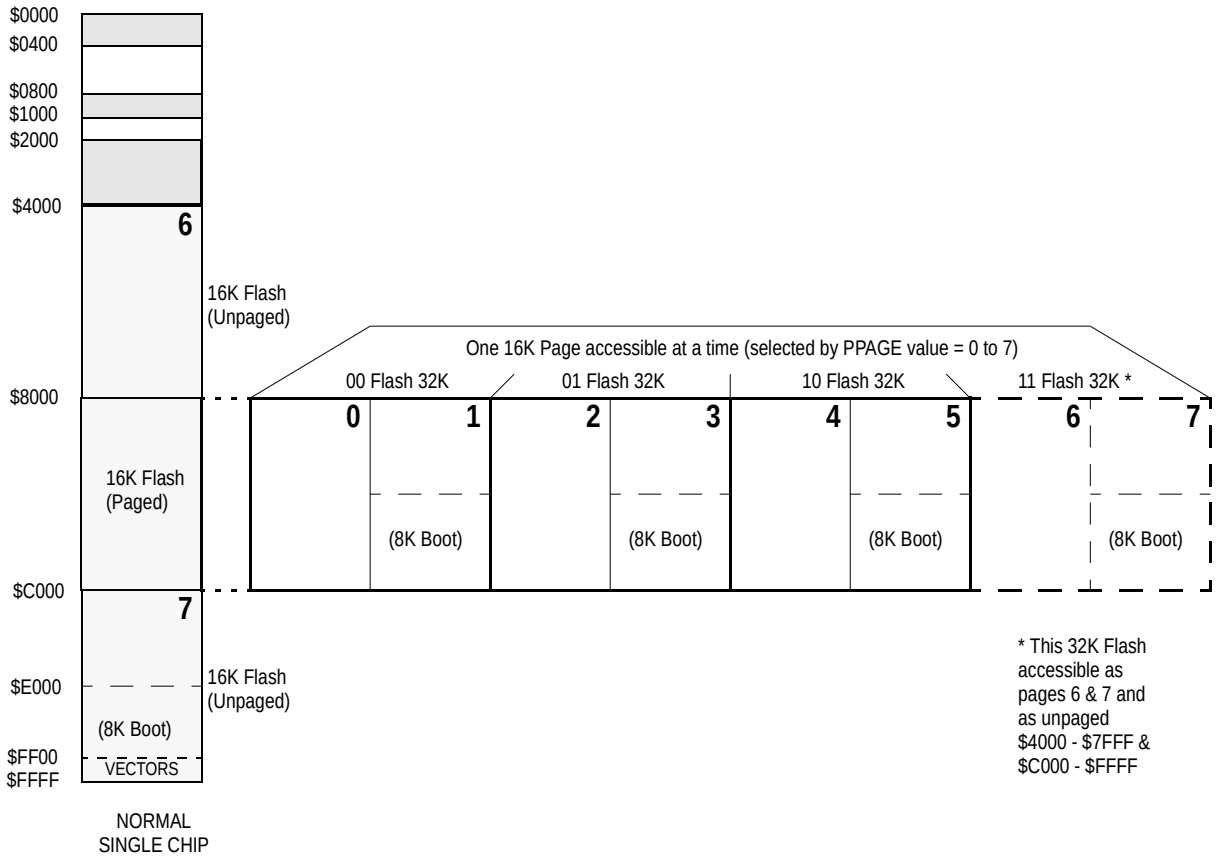


Figure 11 MC68HC912DT128A Memory Paging

Bus Control and Input/Output

Contents

Introduction

Internally the MC68HC912DT128A has full 16-bit data paths, but depending upon the operating mode and control registers, the external multiplexed 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.

It is possible to have a mix of 8 and 16 bit peripherals attached to the external multiplexed bus, using the NDRF bit in the MISC register while in expanded wide modes.

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.

Table 16 Access Type vs. Bus Control Pins

LSTRB	A0	R/$\overline{W}$	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 odd address (low/high data swapped)

Registers

Not all registers are visible in the MC68HC912DT128A 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/$\overline{W}$). 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:	—	—	—	—	—	—	—	—
Expanded & Periph:	ADDR15/ DATA15	ADDR14/ DATA14	ADDR13/ DATA13	ADDR12/ DATA12	ADDR11/ DATA11	ADDR10/ DATA10	ADDR9/ DATA9	ADDR8/ DATA8
Expanded narrow	ADDR15/ DATA15/ DATA7	ADDR14/ DATA14/ DATA6	ADDR13/ DATA13/ DATA5	ADDR12/ DATA12/ DATA4	ADDR11/ DATA11/ DATA3	ADDR10/ DATA10/ DATA2	ADDR9/ DATA9/ DATA1	ADDR8/ DATA8/ DATA0

Bits PA[7:0] are associated respectively with addresses ADDR[15:8], DATA[15:8] and DATA[7:0], in narrow mode. When this port is not used for external addresses 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:	—	—	—	—	—	—	—	—
Expanded & 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] (except in narrow mode) respectively. When this port is not used for external addresses 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
	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:	—	—	—	—	—	—	—	—
Alt. Pin Function	$\overline{\text{DBE}}$ or ECLK or CAL	MODB or IPIPE1 or CGMTST	MODA or IPIPE0	ECLK	$\overline{\text{LSTRB}}$ or TAGLO	R/$\overline{\text{W}}$	$\overline{\text{IRQ}}$	$\overline{\text{XIRQ}}$

This register is associated with external bus control signals and interrupt inputs, including data bus enable ($\overline{\text{DBE}}$), mode select (MODB/IPIPE1, MODA/IPIPE0), E clock, size ($\overline{\text{LSTRB}}$), read/write (R/$\overline{\text{W}}$), $\overline{\text{IRQ}}$, and $\overline{\text{XIRQ}}$. 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 ($\overline{\text{DBE}}$, $\overline{\text{LSTRB}}$, R/$\overline{\text{W}}$, $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$). A single control bit enables the pull-ups for all these pins which are configured as inputs.

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	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 $\overline{\text{XIRQ}}$ and $\overline{\text{IRQ}}$ 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	CGMTE	PIPOE	NECLK	LSTRE	RDWE	CALE	DBENE	
RESET:	0	0	0	0	0	0	0	0	Normal Expanded
RESET:	0	0	1	0	1	1	0	0	Special Expanded
RESET:	1	1	0	1	0	0	0	0	Peripheral
RESET:	1	0	0	1	0	0	0	0	Normal single chip
RESET:	0	0	1	0	1	1	0	0	Special single chip

Port E serves as general purpose I/O lines or as system and bus control signals. The PEAR register is used to choose between the general-purpose I/O functions and the alternate bus control functions. 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 normal expanded modes, the reset vector is located in external memory. The $\overline{\text{DBE}}$ and E clock are required for de-multiplexing address and data but $\overline{\text{LSTRB}}$ and R/$\overline{\text{W}}$ are only needed by the system when there are external writable resources. Therefore in normal expanded modes, the $\overline{\text{DBE}}$ and the E clock are configured for their alternate bus control functions 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, $\overline{\text{DBE}}$, IPIPE1, IPIPE0, E, $\overline{\text{LSTRB}}$, and R/$\overline{\text{W}}$ are configured as bus-control signals.

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

In peripheral mode, the PEAR register is not accessible for reads or writes. However, the CGMTE control bit is reset to one to configure PE6 as a test output for the CGM module.

NDBE — No Data Bus Enable

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

0 = PE7 is used for $\overline{\text{DBE}}$, external control of data enable on memories, or inverted E clock.

1 = PE7 is CAL function if CALE bit is set in PEAR register or general-purpose I/O otherwise.

The NDBE bit has no effect in Single Chip or Peripheral Modes and PE7 is defaulted to the CAL function if CALE bit is set in PEAR register or to an I/O otherwise.

CGMTE — Clock Generator Module Testing Enable

Normal: write never; Special: write anytime EXCEPT the first time. Read anytime.

0 = PE6 is general-purpose I/O or pipe output.

1 = PE6 is a test signal output from the CGM module (no effect in single chip or normal expanded modes). PIPOE = 1 overrides this function and forces PE6 to be a pipe status output signal.

PIPOE — Pipe Status Signal Output Enable

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

0 = PE[6:5] are general-purpose I/O (if CGMTE = 1, PE6 is a test output signal from the CGM module).

1 = PE[6:5] are outputs and indicate the state of the instruction queue (only effective in expanded modes).

NECLK — No External E Clock

Normal single chip: write once; special single chip: write anytime; all other modes: write never.

Read anytime. In peripheral mode, E is an input and in all other modes, E is an output.

0 = PE4 is the external E-clock pin subject to the following limitation: In single-chip modes, to get an E clock output signal, it is necessary to have ESTR = 0 in addition to NECLK = 0. A 16-bit write to PEAR and MODE registers 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 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.

CALE — Calibration Reference Enable

Read and write anytime.

0 = Calibration reference is disabled and PE7 is general purpose I/O in single chip or peripheral modes or if NDBE bit is set.

1 = Calibration reference is enabled on PE7 in single chip and peripheral modes or if NDBE bit is set.

DBENE — $\overline{\text{DBE}}$ or Inverted E Clock on PE7

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

DBENE controls which signal is output on PE7 when NDBE control bit is cleared. The inverted E clock output can be used to latch the address for de-multiplexing. It has the same behavior as the E clock, except it is inverted. Please note that in the case of idle expansion bus, the ‘not E clock’ signal could stay high for many cycles.

The DBENE bit has no effect in Single Chip or Peripheral Modes and PE7 is defaulted to the CAL function if CALE bit is set in PEAR register or to an I/O otherwise.

0 = PE7 pin used for $\overline{\text{DBE}}$ external control of data enable on memories in expanded modes when NDBE = 0

1 = PE7 pin used for inverted E clock output in expanded modes when NDBE = 0

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

	Bit 7	6	5	4	3	2	1	Bit 0
	PUPK	PUPJ	PUPH	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.

PUPK — Pull-Up Port K Enable

0 = Port K pull-ups are disabled.

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

PUPJ — Pull-Up or Pull-Down Port J Enable

0 = Port J resistive loads (pull-ups or pull-downs) are disabled.

1 = Enable resistive load devices (pull-ups or pull-downs) for all port J input pins.

PUPH — Pull-Up or Pull-Down Port H Enable

0 = Port H resistive loads (pull-ups or pull-downs) are disabled.

1 = Enable resistive load devices (pull-ups or pull-downs) for all port H input pins.

PUPE — Pull-Up Port E Enable

0 = Port E pull-ups on PE7, PE3, PE2, PE1 and PE0 are disabled.

1 = Enable pull-up devices for port E input pins PE7, PE3, PE2, PE1 and PE0.

When this bit is set port E input pins 7, 3, 2, 1 & 0 have an active pull-up device.

PUPB — Pull-Up Port B Enable

0 = Port B pull-ups are disabled.

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

PUPA — Pull-Up Port A Enable

0 = Port A pull-ups are disabled.

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

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

	Bit 7	6	5	4	3	2	1	Bit 0
	RDPK	RDPJ	RDPH	RDPE	0	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.

RDPK — Reduced Drive of Port K

0 = All port K output pins have full drive enabled.
1 = All port K output pins have reduced drive capability.

RDPJ — Reduced Drive of Port J

0 = All port J output pins have full drive enabled.
1 = All port J output pins have reduced drive capability.

RDPH — Reduced Drive of Port H

0 = All port H output pins have full drive enabled.
1 = All port H output pins have reduced drive capability.

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.

Flash EEPROM

Contents

Introduction

The four Flash EEPROM array modules 00FEE32K, 01FEE32K, 10FEE32K and 11FEE32K for the MC68HC912DT128A serve as electrically erasable and programmable, non-volatile ROM emulation memory. The modules 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 they can be used for static data which is read frequently. The Flash EEPROM module is ideal for program storage for single-chip applications allowing for field reprogramming.

Overview

Each 32K 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.

Programming is by aligned word. The Flash EEPROM module supports bulk erase only.

Each Flash EEPROM module has hardware interlocks which protect stored data from accidental corruption. An erase- and program-protected 8-Kbyte block for boot routines is located at the top of each 32-Kbyte array. Since boot programs must be available at all times, the only useful boot block is at $E000–$FFFF location. All paged boot blocks can be used as protected program space if desired.

Flash EEPROM Control Block

A 4-byte register block for each module controls the Flash EEPROM 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 and points to the 00FEE32K register block.

Flash EEPROM Arrays

After reset, a fixed 32K Flash EEPROM array, 11FEE32K, is located from addresses $4000 to $7FFF and from $C000 to $FFFF. The other three 32K Flash EEPROM arrays 00FEE32K, 01FEE32K and 10FEE32K, are mapped through a 16K byte program page window located from addresses $8000 to $BFFF. The page window has eight 16K byte pages. The last two pages also map the physical location of the fixed 32K Flash EEPROM array 11FEE32K. In expanded modes, the Flash EEPROM arrays are turned off. See **Operating Modes**.

Flash EEPROM Registers

Each 32K byte Flash EEPROM module has a set of registers. The register space $00F4-$00F7 is in a register space window of four pages. Each register page of four bytes maps the register space for each Flash module and each page is selected by the PPAGE register. See **Resource Mapping**

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 blocks are located at $E000–$FFFF and $A000–$BFFF for odd program pages for each Flash EEPROM module. Since boot programs must be available at all times, the only useful boot block is at $E000–$FFFF location. All paged boot blocks can be used as protected program space if desired.

0 = Enable erase and program of 8K byte boot block
1 = Disable erase and program of 8K byte boot block

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

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	FEESWAI	HVEN	0	ERAS	PGM
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.

HVEN — High-Voltage Enable

This bit enables the charge pump to supply high voltages for program and erase operations in the array. HVEN can only be set if either PGM or ERAS are set and the proper sequence for program or erase is followed.

0 = Disables high voltage to array and charge pump off

1 = Enables high voltage to array and charge pump on

ERAS — Erase Control

This bit configures the memory for erase operation. ERAS is interlocked with the PGM bit such that both bits cannot be equal to 1 or set to1 at the same time.

0 = Erase operation is not selected.

1 = Erase operation selected.

PGM — Program Control

This bit configures the memory for program operation. PGM is interlocked with the ERAS bit such that both bits cannot be equal to 1 or set to1 at the same time.

0 = Program operation is not selected.

1 = Program operation selected.

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.

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.

Normal Operation

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

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.

Programming is restricted to 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.

Programming and erasing of Flash locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed t_{FPGM} maximum (40µs).

Programming the Flash EEPROM

Programming the Flash EEPROM is done on a row basis. A row consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $XX80

and $XXC0. Use this step-by-step procedure to program a row of Flash memory.

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Write to any Flash address with any data within the row address range desired.
3. Wait for a time, t_{NVS} (min. 10µs).
4. Set the HVEN bit.
5. Wait for a time, t_{PGS} (min. 5µs).
6. Write to the Flash address with data to the word desired to be programmed. If BOOTP is asserted, an attempt to program an address in the boot block will be ignored.
7. Wait for a time, t_{FPGM} (min. 30µs).
8. Repeat step 6 and 7 until all the words within the row are programmed.
9. Clear the PGM bit.
10. Wait for a time, t_{NVH} (min. 5µs).
11. Clear the HVEN bit.
12. After time, t_{RCV} (min 1µs), the memory can be accessed in read mode again.

This program sequence is repeated throughout the memory until all data is programmed. For minimum overall programming time and least program disturb effect, the sequence should be part of an intelligent operation which iterates per row.

Erasing the Flash EEPROM

The following sequence demonstrates the recommended procedure for erasing any of the Flash EEPROM array.

1. Set the ERAS bit.
2. Write to any valid address in the Flash array. The data written and the address written are not important. The boot block will be

erased only if the control bit BOOTP is negated.

3. Wait for a time, t_{NVS}.
4. Set the HVEN bit.
5. Wait for a time, t_{ERAS}.
6. Clear the ERAS bit.
7. Wait for a time, t_{NVHL}.
8. Clear the HVEN bit.
9. After time, t_{RCV}, the memory can be accessed in read mode again.

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.

CAUTION: *It is advised not to enter stop or wait modes when program or erase operation of the Flash array is in progress.*

EEPROM

Contents

Introduction

The MC68HC912DT128A EEPROM nonvolatile memory is arranged in a 16-bit configuration. The EEPROM array may be read as either bytes, aligned words or misaligned words. Access times are 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 normal 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. 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.

EEPROM Selective Write More Zeros

The EEPROM can be programmed such that one or multiple bits are programmed (written to a logic “0”) at a time. However, the user should may never program one bit more than once before erasing the entire byte. In other words, the user is not allowed to over write a logic “0” with another “0’.

For some applications it may be advantageous to track more than 10k events with a single byte of EEPROM by programming one bit at a time. For that purpose, a special selective bit programming technique is available. An example is shown here.

Original state of byte = binary 1111:1111 (erased)

First event is recorded by programming bit position 0
Program write = binary 1111:1110; Result = binary 1111:1110

Second event is recorded by programming bit position 1
Program write = binary 1111:1101; Result = binary 1111:1100

Third event is recorded by programming bit position 2
Program write = binary 1111:1011; Result = binary 1111:1000

Fourth event is recorded by programming bit position 3
Program write = binary 1111:0111; Result = binary 1111:0000

Events five through eight are recorded in a similar fashion.

Note that none of the bit locations are actually programmed more than once although the byte was programmed eight times.

When this technique is utilized, a program / erase cycle is defined as multiple writes (up to eight) to a unique location followed by a single erase sequence.

EEPROM Programmer's Model

The EEPROM module consists of two separately addressable sections. The first is an eight-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 eight-byte register section starts at address $00EC and the EEPROM array is located from addresses $0800 to $0FFF. Registers $00EC-$00ED are reserved.

Read/write access to the memory array section can be enabled or disabled by the EEON control bit in the INITEE register ($0012). This feature allows the access of memory mapped resources that have lower priority than the EEPROM memory array. EEPROM control registers can be accessed regardless of the state of EEON. Any EEPROM erase or program operation already in progress will not be affected by the change of EEON state. For information on re-mapping the register block and EEPROM address space, refer to **Operating Modes**.

CAUTION: *It is strongly recommended to discontinue program/erase operations during WAIT (when EESWAI=1) or STOP modes since all program/erase activities will be terminated abruptly and considered unsuccessful.*

For lowest power consumption during WAIT mode, it is advised to turn off EEPGM.

The EEPROM module contains an extra word called SHADOW word which is loaded at reset into the EEMCR, EEDIVH and EEDIVL registers. To program the SHADOW word, when in special modes (SMODN=0), the NOSHW bit in EEMCR register must be cleared. Normal programming routines are used to program the SHADOW word which becomes accessible at address $0FC0-$0FC1 when NOSHW is cleared. At the next reset the SHADOW word data is loaded into the EEMCR, EEDIVH and EEDIVL registers. The SHADOW word can be protected from being programmed or erased by setting the SHPROT bit of EEPROT register.

A steady internal self-time clock is required to provide accurate counts to meet EEPROM program/erase requirements. This clock is generated via by a programmable 10-bit prescaler register. Automatic program/erase termination is also provided.

In ordinary situations, with crystal operating properly, the steady internal self-time clock is derived from the input clock source (EXTALi). The divider value is as in EEDIVH:EEDIVL. In limp-home mode, where the oscillator has malfunctioned or is unavailable, the self-time clock is derived from the PLL at f_{VCOMIN} (nominally 1 MHz), with a predefined divider value of $0023. Program/erase operation is not guaranteed in limp-home mode. The clock switching function is only applicable for permanent loss of crystal condition, so the program/erase will also not be guaranteed when the loss of crystal condition is intermittent.

It is strongly recommended that the clock monitor is enabled to ensure that the program/erase operation will be shutdown in the event of loss of crystal with a clock monitor reset, or switch to a limp-home mode clock. This will prevent unnecessary stress on the emulated EEPROM during oscillator failure.

EEPROM Control Registers

EEDIVH — EEPROM Modulus Divider **$00EE**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	0	EEDIV9	EEDIV8
RESET:	0	0	0	0	0	0	—(1)	—(1)

1. Loaded from SHADOW word.

EEDIVL — EEPROM Modulus Divider **$00EF**

	Bit 7	6	5	4	3	2	1	Bit 0
	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
RESET:	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1. Loaded from SHADOW word.

EEDIV[9:0] — Prescaler divider

Loaded from SHADOW word at reset.

Read anytime. Write once in normal modes (SMODN =1) if EELAT = 0 and anytime in special modes (SMODN =0) if EELAT = 0.

The prescaler divider is required to produce a self-time clock with a fixed frequency around 28.6 Khz for the range of oscillator frequencies. The divider is set so that the oscillator frequency can be divided by a divide factor that can produce a 35 µs ± 2µs timebase.

CAUTION: *An incorrect or uninitialized value on EEDIV can result in overstress of EEPROM array during program/erase operation. It is also strongly recommend not to program EEPROM with oscillator frequencies less than 250 Khz.*

The EEDIV value is determined by the following formula:

$$EEDIV = INT[EXTALi\ (hz) \times 35 \times 10^{-6} + 0.5]$$

NOTE: *INT[A] denotes the round down integer value of A. Program/erase cycles will not be activated when EEDIV = 0.*

Table 17 EEDIV Selection

Osc Freq.	Osc Period	Divide Factor	EEDIV
16 Mhz	62.5ns	560	$0230
8 Mhz	125ns	280	$0118
4 Mhz	250ns	140	$008C
2 Mhz	500ns	70	$0046
1 Mhz	1µs	35	$0023
500 Khz	2µs	18	$0012
250 Khz	4µs	9	$0009

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

	Bit 7	6	5	4	3	2	1	Bit 0
	NOBDML	NOSHW	RESERVED(1)		1	EESWAI	PROTLCK	DMY
RESET:	—(2)	—(2)	—(2)	—(2)	1	1	0	0

1. Bits 4 and 5 have test functions and should not be programmed.
2. Loaded from SHADOW word.

Bits[7:4] are loaded at reset from the EEPROM SHADOW word.

NOTE: *The bits 5 and 4 are reserved for test purposes. These locations in SHADOW word should not be programmed otherwise some locations of regular EEPROM array will not be more visible.*

NOBDML — Background Debug Mode Lockout Disable

0 = The BDM lockout is enabled.
1 = The BDM lockout is disabled.

Loaded from SHADOW word at reset.
Read anytime. Write anytime in special modes (SMODN=0).

NOSHW — SHADOW Word Disable

0 = The SHADOW word is enabled and accessible at address $0FC0-$0FC1.
1 = Regular EEPROM array at address $0FC0-$0FC1.

Loaded from SHADOW word at reset.
Read anytime. Write anytime in special modes (SMODN=0).

When NOSHW cleared, the regular EEPROM array bytes at address $0FC0 and $0FC1 are not visible. The SHADOW word is accessed instead for both read and program/erase operations. Bits[7:4] from the high byte of the SHADOW word, $0FC0, are loaded to EEMCR[7:4]. Bits[1:0] from the high byte of the SHADOW word, $0FC0,are loaded to EEDIVH[1:0]. Bits[7:0] from the low byte of the SHADOW word, $0FC1,are loaded to EEDIVL[7:0]. BULK program/erase only applies if SHADOW word is enabled.

NOTE: *Bit 6 from high byte of SHADOW word should not be programmed in order to have the full EEPROM array visible.*

EESWAI — EEPROM Stops in Wait Mode

0 = The module is not affected during WAIT mode
1 = The module ceases to be clocked during WAIT mode

Read and write anytime.

NOTE: *The EESWAI 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).

DMY— Dummy bit

Read and write anytime.

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

	Bit 7	6	5	4	3	2	1	Bit 0
	SHPROT	1	BPROT5	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.

SHPROT — SHADOW Word Protection

0 = The SHADOW word can be programmed and erased.

1 = The SHADOW word is protected from being programmed and erased.

BPROT[5:0] — EEPROM Block Protection

0 = Associated EEPROM block can be programmed and erased.

1 = Associated EEPROM block is protected from being programmed and erased.

Table 18 2K byte EEPROM Block Protection

Bit Name	Block Protected	Block Size
BPROT5	$0800 to $0BFF	1024 Bytes
BPROT4	$0C00 to $0DFF	512 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
	0	EREVTN	0	0	0	ETMSD	ETMR	ETMSE
RESET:	0	0	0	0	0	0	0	0

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

.

EEPROG — EEPROM Control **$00F3**

	Bit 7	6	5	4	3	2	1	Bit 0
	BULKP	0	AUTO	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.

AUTO — Automatic shutdown of program/erase operation.

EEPGM is cleared automatically after the program/erase cycles are finished when AUTO is set.

0 = Automatic clear of EEPGM is disabled.

1 = Automatic clear of EEPGM is enabled.

Read anytime. Write anytime if EEPGM = 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 19 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 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.
1 = EEPROM configuration for erasure.

Read anytime. Write anytime if EEPGM = 0.

Configures the EEPROM for erasure or programming.

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 except when EEPGM = 1 or EEDIV = 0.

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 EELAT and EEPGM are set simultaneously, EEPGM remains clear but EELAT is set.

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

Program/Erase Operation

A program or erase operation should follow the sequence below if AUTO bit is clear:

1. Write BYTE, ROW and ERASE to 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

If the AUTO bit is set, steps 4 and 5 can be replaced by a step to poll the EEPGM bit until it is cleared.

CAUTION: *The state machine will not start if an attempt is made to program or erase a protected location and therefore the EEPGM bit will never clear on that EEPROM operation. Check for protected status or use a software timeout to avoid a continuous loop whilst polling the EEPGM bit. If using a timeout, ensure steps 5 and 6 are still executed.*

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

Shadow Word Mapping

The shadow word is mapped to location $_FC0 and $_FC1 when the NOSHW bit in EEMCR register is zero. The value in the shadow word is loaded to the EEMCR, EEDIVH and EEDIVL after reset. Table 20 shows the mapping of each bit from shadow word to the registers.

Table 20 Shadow word mapping

Shadow word location	Register / Bit
$_FC0 bit 7	EEMCR / NOBDML
$_FC0, bit 6	EEMCR / NOSHW
$_FC0, bit 5	EEMCR / bit 5(1)
$_FC0, bit 4	EEMCR / bit 4(1)
$_FC0, bit 3:2	not mapped(2))
$_FC0, bit 1:0	EEDIVH / bit 1:0
$_FC1, bit 7:0	EEMCR / bit 7:0

1. Reserved for testing. Must be set to one in user application.

2. Reserved. Must be set to one in user application for future compatibility.

Programming EEDIVH and EEDIVL Registers

The EEDIVH and EEDIVL registers must be correctly set according to the oscillator frequency before any EEPROM location can be programmed or erased.

Normal mode

The EEDIVH and EEDIVL registers are write once in normal mode. Upon system reset, the application program is required to write the correct divider value to EEDIVH and EEDIVL registers based on the oscillator frequency. After the first write, the value in the EEDIVH and EEDIVL registers is locked from being overwritten until the next reset. The EEPROM is then ready for standard program/erase routines.

CAUTION: *Runaway code can possibly corrupt the EEDIVH and EEDIVL registers if they are not initialized (write once registers).*

Special mode

If an existing application code with EEPROM program/erase routines is fixed and the system is already operating at a known oscillator frequency, it is recommended to initialize the shadow word with the corresponding EEDIVH and EEDIVL values in special mode. The shadow word initializes EEDIVH and EEDIVL registers upon system reset to ensure software compatibility with existing code. Initializing the EEDIVH and EEDIVL registers in special modes (SMODN=0) is accomplished by the following steps.

1. Write correct divider value to EEDIVH and EEDIVL registers based on the oscillator frequency as per Table 17.
2. Remove the SHADOW word protection by clearing SHPROT bit in EEPROT register.
3. Clear NOSHW bit in EEMCR register to make the SHADOW word visible at $0FC0-$0FC1.
4. Write NOSHW bit in EEMCR register to make the SHADOW word visible at $0FC0-$0FC1.
5. Program bits 1 and 0 of the high byte of the SHADOW word and bits 7 to 0 of the low byte of the SHADOW word like a regular EEPROM location at address $0FC0 and $0FC1. Do not program other bits of the high byte of the SHADOW word (location $0FC0); otherwise some regular EEPROM array locations will not be visible. At the next reset, the SHADOW values are loaded into the EEDIVH and EEDIVL registers. They do not require further initialization as long as the oscillator frequency of the target application is not changed.
6. Protect the SHADOW word by setting SHPROT bit in EEPROT register.

EEPROM

Resets and Interrupts

Contents

Introduction

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.

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)

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 21 lists interrupt sources and vectors in default order of priority. Before masking an interrupt by clearing the corresponding local enable bit, it is required to set the I-bit to avoid an SWI.

Latching of Interrupts

$\overline{\text{XIRQ}}$ is always level triggered and $\overline{\text{IRQ}}$ can be selected as a level triggered interrupt. These level triggered interrupt pins should only be released during the appropriate interrupt service routine. Generally the interrupt service routine will handshake with the interrupting logic to release the pin. In this way, the MCU will never start the interrupt service sequence only to determine that there is no longer an interrupt source. In event that this does occur the trap vector will be taken.

If $\overline{\text{IRQ}}$ is selected as an edge triggered interrupt, the hold time of the level after the active edge is independent of when the interrupt is serviced. As long as the minimum hold time is met, the interrupt will be latched inside the MCU. In this case the IRQ edge interrupt latch is cleared automatically when the interrupt is serviced.

All of the remaining interrupts are latched by the MCU with a flag bit. These interrupt flags should be cleared during an interrupt service routine or when interrupts are masked by the I bit. By doing this, the MCU will never get an unknown interrupt source and take the trap vector.

Table 21 Interrupt Vector Map

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	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	SCI 1	I bit	SC1CR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0 or ATD1	I bit	ATDxCTL2 (ASCIE)	$D2
$FFD0, $FFD1	MSCAN 0 wake-up	I bit	C0RIER (WUPIE)	$D0
$FFCE, $FFCF	Key wake-up J or H	I bit	KWIEJ[7:0] and KWIEH[7:0]	$CE
$FFCC, $FFCD	Modulus down counter underflow	I bit	MCCTL (MCZI)	$CC
$FFCA, $FFCB	Pulse Accumulator B Overflow	I bit	PBCTL (PBOVI)	$CA
$FFC8, $FFC9	MSCAN 0 errors	I bit	C0RIER (RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE)	$C8
$FFC6, $FFC7	MSCAN 0 receive	I bit	C0RIER (RXFIE)	$C6
$FFC4, $FFC5	MSCAN 0 transmit	I bit	C0TCR (TXEIE[2:0])	$C4
$FFC2, $FFC3	CGM lock and limp home	I bit	PLLCR (LOCKIE, LHIE)	$C2
$FFC0, $FFC1	IIC Bus	I bit	IBCR (IBIE)	$C0
$FFBE, $FFBF	MSCAN 1 wake-up	I bit	C1RIER (WUPIE)	$BE

Table 21 Interrupt Vector Map

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFBC, $FFBD	MSCAN 1 errors	I bit	C1RIER (RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE)	$BC
$FFBA, $FFBB	MSCAN 1 receive	I bit	C1RIER (RXFIE)	$BA
$FFB8, $FFB9	MSCAN 1 transmit	I bit	C1TCR (TXEIE[2:0])	$B8
$FFB6, $FFB7(1)	MSCAN 2 wake-up	I bit	C2RIER (WUPIE)	$B6
$FFB4, $FFB5(1)	MSCAN 2 errors	I bit	C2RIER (RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE)	$B4
$FFB2, $FFB3(1)	MSCAN 2 receive	I bit	C2RIER (RXFIE)	$B2
$FFB0, $FFB1(1)	MSCAN 2 transmit	I bit	C2TCR (TXEIE[2:0])	$B0
$FF80–$FFAF	Reserved	I bit		$80–$AE

1. MC68HC912DT128A only

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 is disconnected from interrupt logic.

1 = External $\overline{\text{IRQ}}$ pin is 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 XCLK rate chosen.

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	PSEL6	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 un-implemented 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.

Interrupt test registers

These registers are used in special modes for testing the interrupt logic and priority without needing to know which modules and what functions are used to generate the interrupts.Each bit is used to force a specific interrupt vector by writing it to 1.Bits are named with B6 through F4 to indicate vectors $FFB6 through $FFF4. These bits are also used in special modes to view that an interrupt caused by a module has reached the interrupt module.

These registers can only be read in special modes (read in normal mode will return $00). Reading these registers at the same time as the interrupt is changing will cause an indeterminate value to be read. These registers can only be written in special mode.

ITST0 — Interrupt Test Register 0 **$0018**

	Bit 7	6	5	4	3	2	1	Bit 0
	ITE6	ITE8	ITEA	ITEC	ITEE	ITF0	ITF2	ITF4
RESET:	0	0	0	0	0	0	0	0

ITST1 — Interrupt Test Register 1 **$0019**

	Bit 7	6	5	4	3	2	1	Bit 0
	ITD6	ITD8	ITDA	ITDC	ITDE	ITE0	ITE2	ITE4
RESET:	0	0	0	0	0	0	0	0

ITST2 — Interrupt Test Register 2 **$001A**

	Bit 7	6	5	4	3	2	1	Bit 0
	ITC6	ITC8	ITCA	ITCC	ITCE	ITD0	ITD2	ITD4
RESET:	0	0	0	0	0	0	0	0

ITST3 — Interrupt Test Register 3 **$001B**

	Bit 7	6	5	4	3	2	1	Bit 0
	ITB6	ITB8	ITBA	ITBC	ITBE	ITC0	ITC2	ITC4
RESET:	0	0	0	0	0	0	0	0

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.

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.

It is important to use an external low voltage reset circuit (for example: MC34064 or MC33464) to prevent power transitions or corruption of RAM or EEPROM.

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 nine E-clock cycles after an internal device releases reset. When a reset condition is sensed, an internal circuit drives the $\overline{\text{RESET}}$ pin low and a clocked reset sequence controls when the MCU can begin normal processing. In the case of a clock monitor error, a 4096 cycle oscillator start-up delay is imposed before the reset recovery sequence starts (reset is driven low throughout this 4096 cycle delay). The internal reset recovery sequence then drives reset low for 16 to 17 cycles and releases the drive to allow reset to rise. Nine E-clock cycles later the reset pin 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 reset from being detected during an external reset, hold the reset pin low for at least 32 cycles. To prevent a clock monitor reset from being detected during an external reset, hold the reset pin low for at least 4096 + 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.

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. In addition, windowed COP operation can be selected. In this mode, a write to the COPRST register must occur in the last 25% of the selected period. A premature write will also reset the part.

Clock Monitor Reset

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

Effects of Reset

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

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.

Clock and Watchdog Control Logic

The COP watchdog system is enabled, with the CR[2:0] bits set for the longest 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.

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.

Parallel I/O

If the MCU comes out of reset in a single-chip mode, all ports are configured as general-purpose high-impedance inputs.

If the MCU comes out of reset in an expanded mode, port A and port B are used for the 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). Out of reset, port J, port H, port K, port IB, port P, port S, port T, port AD0 and port AD1 are all configured as general-purpose inputs.

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.

Memory

After reset, the internal register block is located from $0000 to $03FF, RAM is at $2000 to $3FFF, and EEPROM is located at $0800 to $0FFF. In single chip mode one 32-Kbyte FLASH EEPROM module is located from $4000 to $7FFF and $C000 to $FFFF, and the other three 32-Kbyte FLASH EEPROM modules are accessible through the program page window located from $8000 to $BFFF. The first 32-Kbyte FLASH EEPROM is also accessible through the program page window.

Other Resources

The enhanced capture timer (ECT), pulse width modulation timer (PWM), serial communications interfaces (SCI0 and SCI1), serial peripheral interface (SPI), inter-IC bus (IIC), Motorola Scalable CANs (MSCAN0 and MSCAN1) and analog-to-digital converters (ATD0 and ATD1) are off after reset.

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 22.

Table 22 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 interrupt is fetched.

I/O Ports With Key Wake-Up

Contents

Introduction

The offers 16 additional I/O ports with key wake-up capability.

The key wake-up feature of the MC68HC912DT128A issues an interrupt that will wake up the CPU when it is in the STOP or WAIT mode. Two ports are associated with the key wake-up function: port H and port J. Port H and port J wake-ups are triggered with a either a rising or falling signal edge. For each pin which has an interrupt enabled, there is a path to the interrupt request signal which has no clocked devices when the part is in stop mode. This allows an active edge to bring the part out of stop.

Digital filtering is included to prevent pulses shorter than a specified value from waking the part from STOP.

An interrupt is generated when a bit in the KWIFH or KWIFJ register and its corresponding KWIEH or KWIEJ bit are both set. All 16 bits/pins share the same interrupt vector. Key wake-ups can be used with the pins configured as inputs or outputs.

Default register addresses, as established after reset, are indicated in the following descriptions. For information on re-mapping the register block, refer to **Operating Modes**.

Key Wake-up and port Registers

PORTJ — Port J Register **$0028**

	Bit 7	6	5	4	3	2	1	Bit 0
PORT	PJ7	PJ6	PJ5	PJ4	PJ3	PJ2	PJ1	PJ0
KWU	KWJ7	KWJ6	KWJ5	KWJ4	KWJ3	KWJ2	KWJ1	KWJ0
RESET:	-	-	-	-	-	-	-	-

Read and write anytime.

PORTH — Port H Register **$0029**

	Bit 7	6	5	4	3	2	1	Bit 0
	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
KWU	KWH7	KWH6	KWH5	KWH4	KWH3	KWH2	KWH1	KWH0
RESET:	-	-	-	-	-	-	-	-

Read and write anytime.

DDRJ — Port J Data Direction Register **$002A**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDJ7	DDJ6	DDJ5	DDJ4	DDJ3	DDJ2	DDJ1	DDJ0
RESET:	0	0	0	0	0	0	0	0

Data direction register J is associated with port J and designates each pin as an input or output.

Read and write anytime

DDRJ[7:0] — Data Direction Port J

0 = Associated pin is an input

1 = Associated pin is an output

DDRH — Port H Data Direction Register **$002B**

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

Data direction register H is associated with port H and designates each pin as an input or output. Read and write anytime.

DDRH[7:0] — Data Direction Port H
0 = Associated pin is an input
1 = Associated pin is an output

KWIEJ — Key Wake-up Port J Interrupt Enable Register **$002C**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWIEJ7	KWIEJ6	KWIEJ5	KWIEJ4	KWIEJ3	KWIEJ2	KWIEJ1	KWIEJ0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

KWIEJ[7:0] — Key Wake-up Port J Interrupt Enables
0 = Interrupt for the associated bit is disabled
1 = Interrupt for the associated bit is enabled

KWIEH — Key Wake-up Port H Interrupt Enable Register **$002D**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWIEH7	KWIEH6	KWIEH5	KWIEH4	KWIEH3	KWIEH2	KWIEH1	KWIEH0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

KWIEH[7:0] — Key Wake-up Port H Interrupt Enables
0 = Interrupt for the associated bit is disabled
1 = Interrupt for the associated bit is enabled

KWIFJ — Key Wake-up Port J Flag Register **$002E**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWIFJ7	KWIFJ6	KWIFJ5	KWIFJ4	KWIFJ3	KWIFJ2	KWIFJ1	KWIFJ0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

Each flag is set by an active edge on its associated input pin. This could be a rising or falling edge based on the state of the KWPJ register. To clear the flag, write one to the corresponding bit in KWIFJ.

Initialize this register after initializing KWPJ so that illegal flags can be cleared.

KWIFJ[7:0] — Key Wake-up Port J Flags

0 = Active edge on the associated bit has not occurred

1 = Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set).

KWIFH — Key Wake-up Port H Flag Register **$002F**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWIFH7	KWIFH6	KWIFH5	KWIFH4	KWIFH3	KWIFH2	KWIFH1	KWIFH0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

Each flag is set by an active edge on its associated input pin. This could be a rising or falling edge based on the state of the KWPH register. To clear the flag, write one to the corresponding bit in KWIFH.

Initialize this register after initializing KWPH so that illegal flags can be cleared.

KWIFH[7:0] — Key Wake-up Port H Flags

0 = Active edge on the associated bit has not occurred

1 = Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set)

KWPJ — Key Wake-up Port J Polarity Register **$0030**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWPJ7	KWPJ6	KWPJ5	KWPJ4	KWPJ3	KWPJ2	KWPJ1	KWPJ0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime. It is best to clear the flags after initializing this register because changing the polarity of a bit can cause the associated flag to become set.

KWPJ[7:0] — Key Wake-up Port J Polarity Selects

0 = Falling edge on the associated port J pin sets the associated flag bit in the KWIFJ register and a resistive pull-up device is connected to associated port J input pin.

1 = Rising edge on the associated port J pin sets the associated flag bit in the KWIFJ register and a resistive pull-down device is connected to associated port J input pin.

KWPH — Key Wake-up Port H Polarity Register **$0031**

	Bit 7	6	5	4	3	2	1	Bit 0
	KWPH7	KWPH6	KWPH5	KWPH4	KWPH3	KWPH2	KWPH1	KWPH0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime. It is best to clear the flags after initializing this register because changing the polarity of a bit can cause the associated flag to become set.

KWPH[7:0] — Key Wake-up Port H Polarity Selects

0 = Falling edge on the associated port H pin sets the associated flag bit in the KWIFH register and a resistive pull-up device is connected to associated port H input pin.

1 = Rising edge on the associated port H pin sets the associated flag bit in the KWIFH register and a resistive pull-down device is connected to associated port H input pin.

Key Wake-Up Input Filter

The KWU input signals are filtered by a digital filter which is active only during STOP mode.

The purpose of the filter is to prevent single pulses shorter than a specified value from waking the part from STOP.

The filter is composed of an internal oscillator and a majority voting logic. The filter oscillator starts the oscillation by detecting a triggering edge on an input if the corresponding interrupt enable bit is set.

The majority voting logic takes three samples of an asserted input pin at each filter oscillator period and if two samples are taken at the triggering level, the filter recognizes a valid triggering level and sets the corresponding interrupt flag. In this way the majority voting logic rejects the short non-triggering state between two incoming triggering pulses. As the filter is shared with all KWU inputs, the filter considers any pulse coming from any input pin for which the corresponding interrupt enable bit is set.

The timing specification is given for a single pulse. The time interval between the triggering edges of two following pulses should be greater than the tKWSP in order to be considered as a single pulse by the filter. If this time interval is shorter than tKWSP, the majority voting logic may treat the two consecutive pulses as a single valid pulse.

The filter is shared by all the KWU pins. Hence any valid triggering level on any KWU pin is seen by the filter. The timing specification applies to the input of the filter.

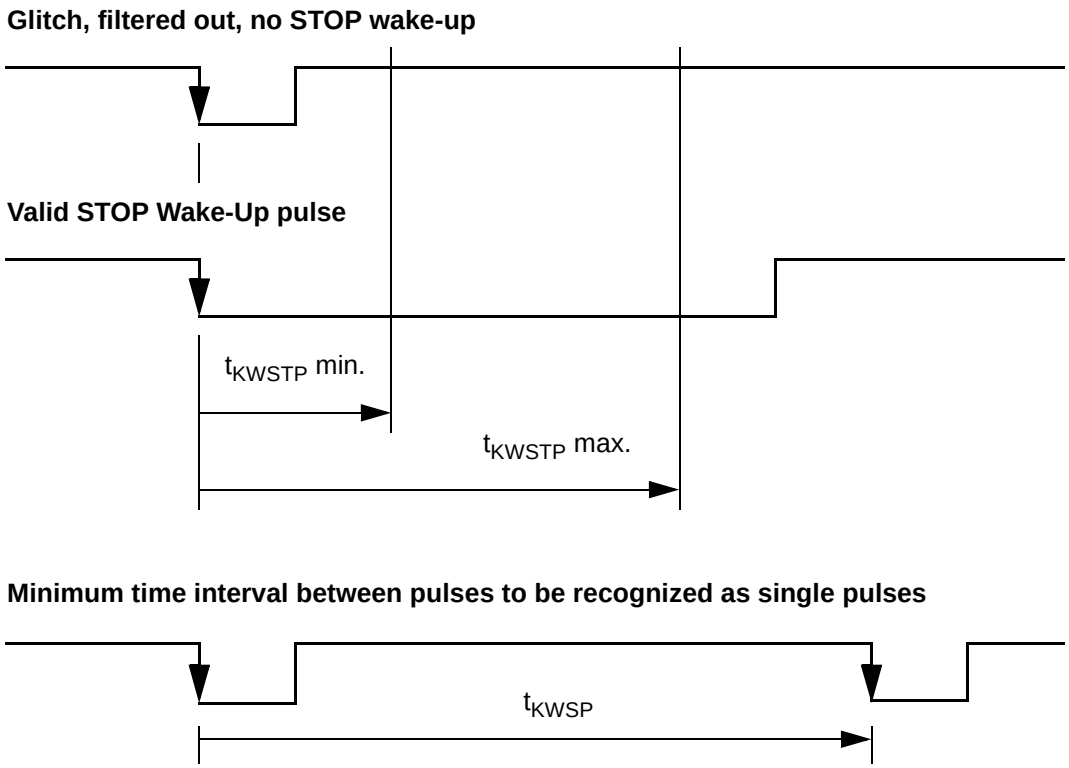


Figure 12 STOP Key Wake-up Filter

Clock Functions

Contents

Introduction

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 MC68HC912DT128A.

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 several types of internal clock signals derived from the primary clock signal:

TxCLK clocks are used by the CPU.

ECLK and PCLK are used by the bus interfaces, SPI, PWM, ATD0 and ATD1.

MCLK is either PCLK or XCLK, and drives on-chip modules such as SCI0, SCI1 and ECT.

XCLK drives on-chip modules such as RTI, COP and restart-from-stop delay time.

SLWCLK is used as a calibration output signal.

The MSCAN module is clocked by EXTALi or SYSCLK, under control of an MSCAN bit.

The clock monitor is clocked by EXTALi.

The BDM system is clocked by BCLK or ECLK, under control of a BDM bit.

A slow mode clock divider is included to deliver a lower clock frequency for the SCI baud rate generators, the ECT timer module, and the RTI and COP clocks. The slow clock bus frequencies divide the crystal frequency in a programmable range of 4 to 252, with steps of 4. This is very useful for low power operation.

See the Clock Divider Chains section for further details. Figure 13 shows some of the timing relationships.

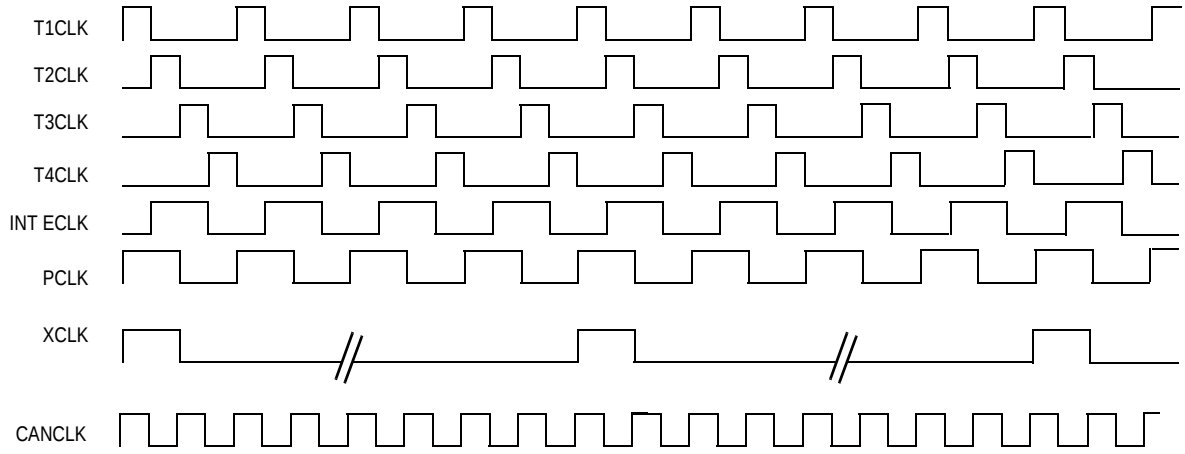


Figure 13 Internal Clock Relationships

Phase-Locked Loop (PLL)

The phase-locked loop (PLL) of the 68HC912D60A is designed for robust operation in an Automotive environment. The proposed PLL crystal or ceramic resonator reference of 0.5 to 8MHz is selected for the wide availability of components with good stability in the desired temperature range. Please refer to Figure 18 in section Clock Divider Chains for an overview of system clocks.

An oscillator design with reduced power consumption allows for slow wait operation with a typical power supply current lower than a milli-ampere. The PLL circuitry can be bypassed when the VDDPLL supply is at VSS level. In this case the oscillator output transistor has a stronger transconductance, for a crystal at twice the bus frequency. Refer to Figure 7 in **Pinout and Signal Descriptions**.

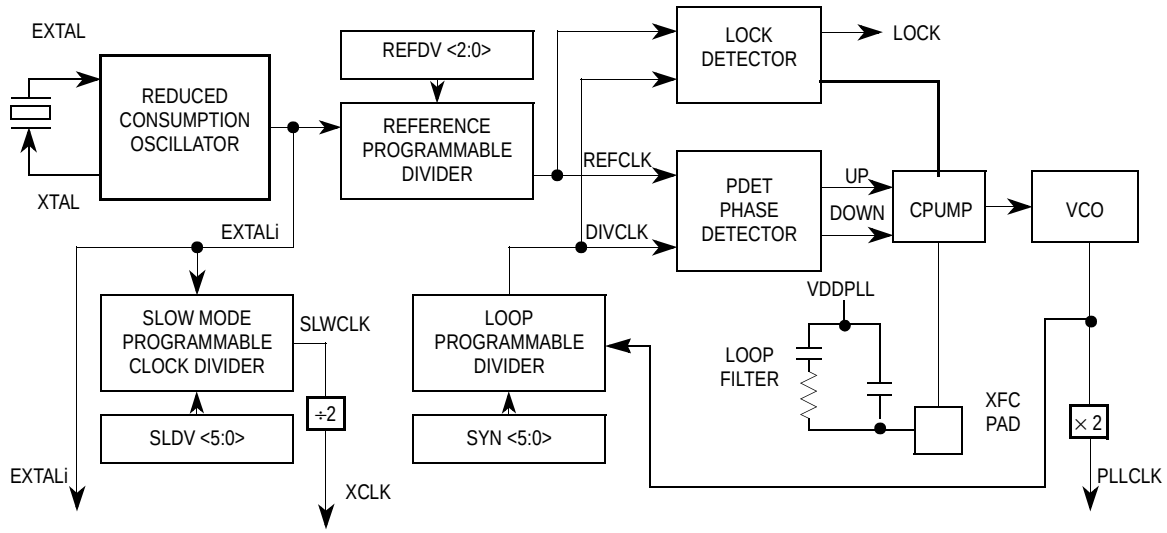


Figure 14 PLL Functional Diagram

The PLL may be used to run the MCU from a different time base than the incoming crystal value. It creates an integer multiple of a reference frequency. For increased flexibility, the crystal clock can be divided by values in a range of 1 – 8 (in unit steps) to generate the reference frequency. The PLL can multiply this reference clock in a range of 1 to 64. Although it is possible to set the divider to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.

If the PLL is selected, it will continue to run when in WAIT mode resulting in more power consumption than normal. To take full advantage of the reduced power consumption of WAIT mode, turn off the PLL before going into WAIT. Please note that in this case the PLL stabilization time applies.

The PLL operation is suspended in STOP mode. After STOP exit followed by the stabilization time, it resumes operation at the same frequency, provided the AUTO bit is set.

A passive external loop filter must be placed on the control line (XFC pad). The filter is a second-order, low-pass filter to eliminate the VCO input ripple. Values of components in the diagram are dependent upon the desired VCO operation. See XFC description.

Acquisition and Tracking Modes

The lock detector compares the frequencies of the VCO feedback clock, DIVCLK, and the final reference clock, REFCLK. Therefore, the speed of the lock detector is directly proportional to the final reference frequency. The circuit determines the mode of the PLL and the lock condition based on this comparison.

The PLL filter is manually or automatically configurable into one of two operating modes:

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. This mode can also be desired in harsh environments when the leakage levels on the filter pin (XFC) can overcome the tracking currents of the PLL charge pump. When in acquisition mode, the $\overline{\text{ACQ}}$ bit in the PLL control register is clear.
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. The PLL enters tracking mode when the VCO frequency is nearly correct. The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically. With an identical filtering time constant, the PLL bandwidth is larger in acquisition mode than in tracking by a ratio of about 3.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, PLLCLK, is safe to use as the source for the base clock, SYSCLK. If PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the PLLCLK clock is safe to use as the source

for the base clock. See Clock Divider Chains. If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application.

The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{\text{ACQ}}$ bit is a read-only indicator of the mode of the filter.
- The $\overline{\text{ACQ}}$ bit is set when the VCO frequency is within a certain tolerance, Δ_{trk}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{unt}. See 19 Electrical Characteristics.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{unl}. See 19 Electrical Characteristics.
- CPU interrupts can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

The PLL also can operate in manual mode (AUTO = 0). All LOCK features described above are active in this mode, only the bandwidth control is disabled. Manual mode is used mainly for systems operating under harsh conditions (e.g.uncoated PCBs in automotive environments). When this is the case, the PLL is likely to remain in acquisition mode. The following conditions apply when in manual mode:

- $\overline{\text{ACQ}}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{\text{ACQ}}$ bit must be clear.
- In case tracking is desired (ACQ = 1), the software must wait a given time, tacq, after turning on the PLL by setting PLLON in the PLL control register. This is to avoid switching to tracking mode too early while the XFC voltage level is still too far away from its quiescent value corresponding to the target frequency. This operation would be very detrimental to the stabilization time.

Limp-Home and Fast STOP Recovery modes

If the crystal frequency is not available due to a crystal failure or a long crystal start-up time, the MCU system clock can be supplied by the VCO at its minimum operating frequency, f_{VCOMIN}. This mode of operation is called Limp-Home Mode and is only available when the VDDPLL supply voltage is at VDD level (i.e. power supply for the PLL module is present). Upon power-up, the ability of the system to start in Limp-Home Mode is restricted to normal MCU modes only.

The Clock Monitor circuit (see section Clock Monitor) can detect the loss of EXTALi, the external clock input signal, regardless of whether this signal is used as the source for MCU clocks or as the PLL reference clock. The clock monitor control bits, CME and FCME, are used to enable or disable external clock detection.

A missing external clock may occur in the three following instances:

- During normal clock operation.
- At Power-On Reset.
- In the STOP exit sequence

Clock Loss during Normal Operation

The ‘no limp-home mode’ bit, NOLHM, determines how the MCU responds to an external clock loss in this case.

With limp home mode disabled (NOLHM bit set) and the clock monitor enabled (CME or FCME bits set), on a loss of clock the MCU is reset via the clock monitor reset vector. This is the same behavior as standard M68HC12 circuits without PLL or operation with VDDPLL at VSS level.

With limp home mode enabled (NOLHM bit cleared) and the clock monitor enabled (CME or FCME bits set), on a loss of clock, the PLL VCO clock at its minimum frequency, f_{VCOMIN}, is provided as the system clock, allowing the MCU to continue operating.

The MCU is said to be operating in **“limp-home” mode** with the forced VCO clock as the system clock. PLLON and BCSP (‘bus clock select PLL’) signals are forced high and the MCS (‘module clock select’) signal

is forced low. The LHOME flag in the PLLFLG register is set to indicate that the MCU is running in limp-home mode. A change of this flag sets the limp-home interrupt flag, LHIF, and if enabled by the LHIE bit, the limp-home mode interrupt is requested. The Clock Monitor is enabled irrespective of CME and FCME bit settings. Module clocks to the RTI & COP (XCLK), BDM (BCLK) and ECT & SCI (MCLK) are forced to be PCLK (at f_{VCOMIN}) and ECLK is also equal to f_{VCOMIN}. MSCAN clock select is unaffected.

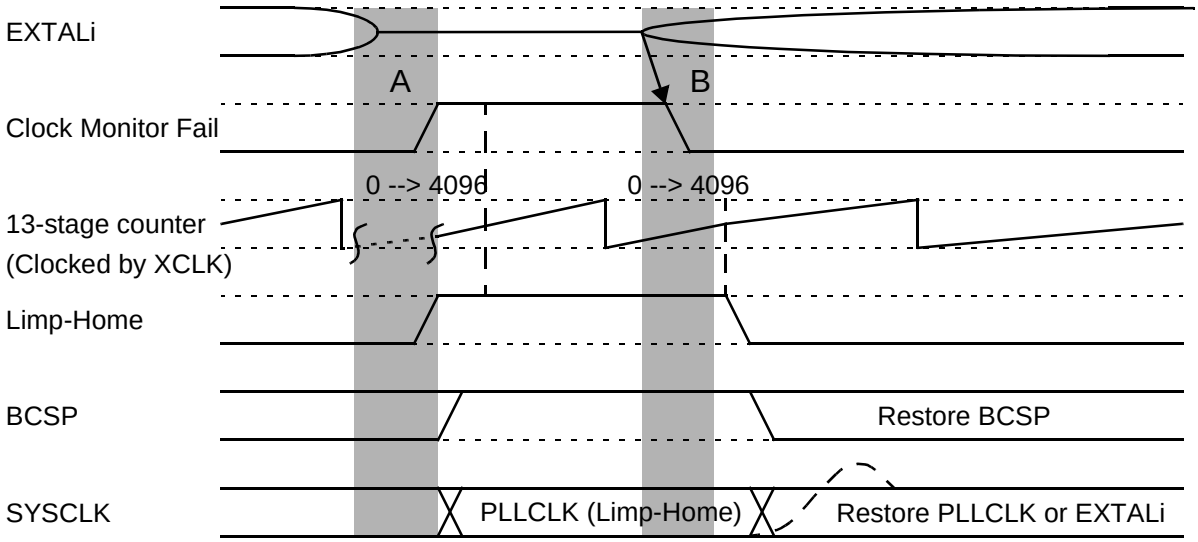


Figure 15 Clock Loss during Normal Operation

The clock monitor is polled each time the 13-stage free running counter reaches a count of 4096 XCLK cycles i.e. mid-count, hence the clock status gets checked once every 8192 XCLK cycles. When the presence of an external clock is detected, the MCU exits limp-home mode, clearing the LHOME flag and setting the limp-home interrupt flag. Upon leaving limp-home mode, BCSP and MCS signals are restored to their values before the clock loss. All clocks return to their normal settings and Clock Monitor control is returned to the CME & FCME bits. If AUTO and BCSP bits were set before the clock loss (selecting the PLL to provide a system clock) the SYSCLK ramps-up and the PLL locks at the previously selected frequency. To prevent PLL operation when the external clock

frequency comes back, software should clear the BCSP bit while running in limp-home mode.

The two shaded regions **A** and **B** in Figure 15 present a of code run away due to incorrect clocks on SYSCLK if the MCU is clocked by EXTALi and the PLL is not used.

In region **A**, there is a delay between the loss of clock and its detection by the clock monitor. When the EXTALi clock signal is disturbed, the clock generation circuitry may receive an out of spec signal and drive the CPU with irregular clocks. This may lead to code runaway.

In region **B**, as the 13-stage counter is free running, the count of 4096 may be reached when the amplitude of the EXTALi clock has not stabilized. In this case, an improper EXTALi is sent to the clock generation circuitry when limp-home mode is exited. This may also cause code runaway.

If the MCU is clocked by the PLL, the risk of code runaway is very low, but it can still occur under certain conditions due to irregular clocks from the clock source appearing on the SYSCLK.

CAUTION: *The COP watch dog should always be enabled in order to reset the MCU in case of a code runaway situation.*

NOTE: *It is always advisable to take additional precautions within the application software to trap such situations.*

No Clock at Power-On Reset

The voltage level on VDDPLL determines how the MCU responds to an external clock loss in this case.

With the VDDPLL supply voltage at VDD level, any reset sets the Clock Monitor Enable bit (CME) and the PLLON bit and clears the NOLHM bit. Therefore, if the MCU is powered up without an external clock, limp-home mode is entered provided the MCU is in a normal mode of operation.

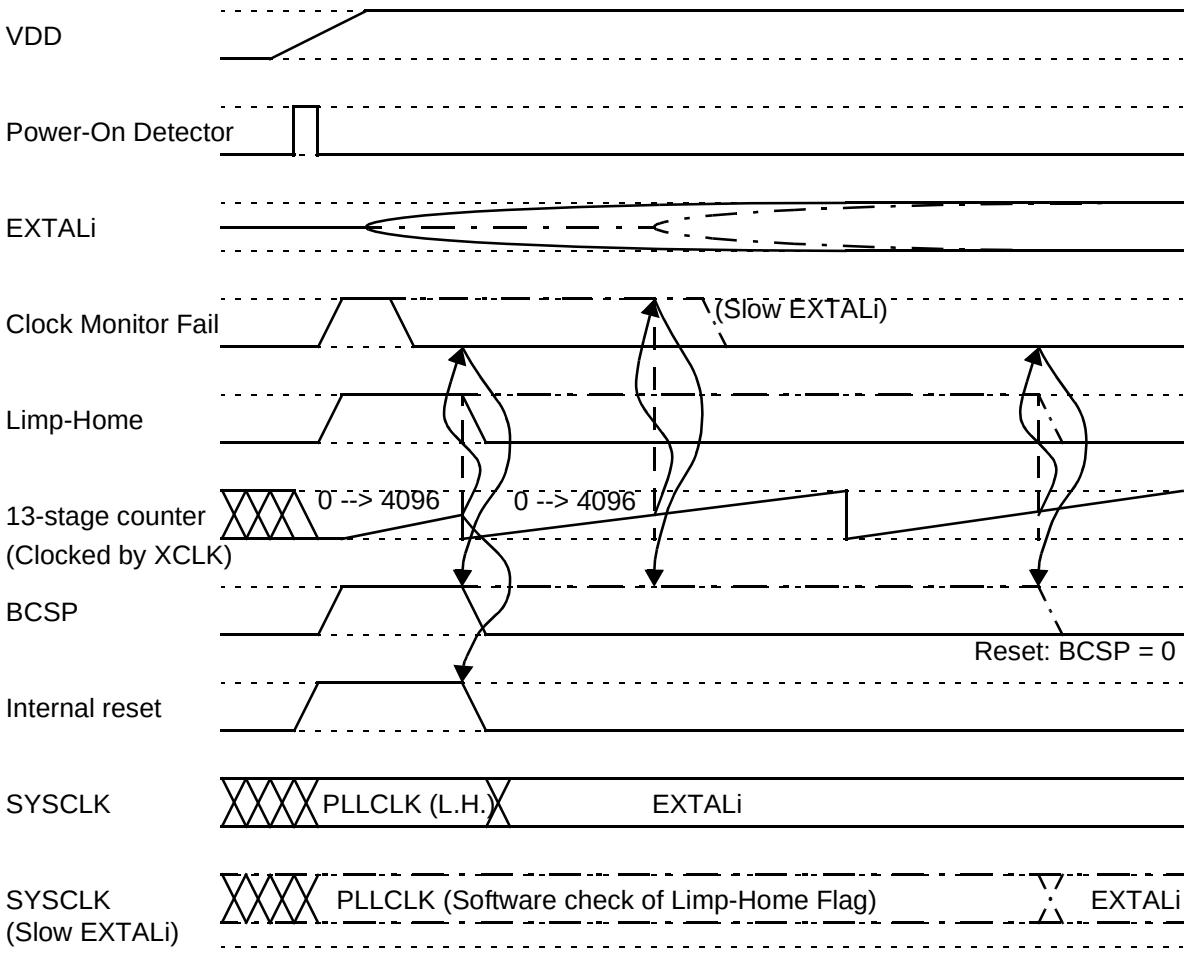


Figure 16 No Clock at Power-On Reset

During this power up sequence, after the POR pulse falling edge, the VCO supplies the limp-home clock frequency to the 13-stage counter, as the BCSP output is forced high and MCS is forced low. XCLK, BCLK and MCLK are forced to be PCLK, which is supplied by the VCO at f_{VCOMIN}. The initial period taken for the 13-stage counter to reach 4096 defines the internal reset period.

If the clock monitor indicates the presence of an external clock during the internal reset period, limp-home mode is de-asserted and the 13-stage counter is then driven by EXTALi clock. After the 13-stage counter reaches a count of 4096 XCLK cycles, the internal reset is released, the 13-stage counter is reset and the MCU exits reset normally using EXTALi clock.

However, if the crystal start-up time is longer than the initial count of 4096 XCLK cycles, or in the absence of an external clock, the MCU will leave the reset state in limp-home mode. The LHOME flag is set and LHIF limp-home interrupt request is set, to indicate it is not operating at the desired frequency. Then after yet another 4096 XCLK cycles followed regularly by 8192 XCLK cycles (corresponding to the 13-stage counter timing out), a check of the clock monitor status is performed. When the presence of an external clock is detected limp-home mode is exited generating a limp-home interrupt if enabled.

CAUTION: *The clock monitor circuit can be misled by the EXTALi clock into reporting a good signal before it has fully stabilised. Under these conditions improper EXTALi clock cycles can occur on SYSCLK. This may lead to a code runaway. To ensure that this situation does not occur, the external Reset period should be longer than the oscillator stabilisation time - this is an application dependent parameter.*

With the VDDPLL supply voltage at VSS level, the PLL module and hence limp-home mode are disabled, the device will remain effectively in a static state whilst there is no activity on EXTALi. The internal reset period and MCU operation will execute only on EXTALi clock.

NOTE: *The external clock signal must stabilise within the initial 4096 reset counter cycles.*

STOP Exit and Fast STOP Recovery

Stop mode is entered when a STOP instruction is executed. Recovery from STOP depends primarily on the state of the three status bits NOLHM, CME & DLY.

The DLY bit controls the duration of the waiting period between the actual exit for some key blocks (e.g. clock monitor, clock generators) and the effective exit from stop for all the rest of the MCU. DLY=1 enables the 13-stage counter to generate a 4096 count delay. DLY=0 selects no delay. As the XCLK is derived from the slow mode divider, the value in the SLOW register modifies the actual delay time.

NOTE: ***DLY=0 is only recommended when there is a good signal available at the EXTAL pin (e.g. an external square wave source).***

STOP mode is exited with an external reset, an external interrupt from IRQ or XIRQ, a Key Wake-Up interrupt from port J or port H, or an MSCAN Wake-Up interrupt.

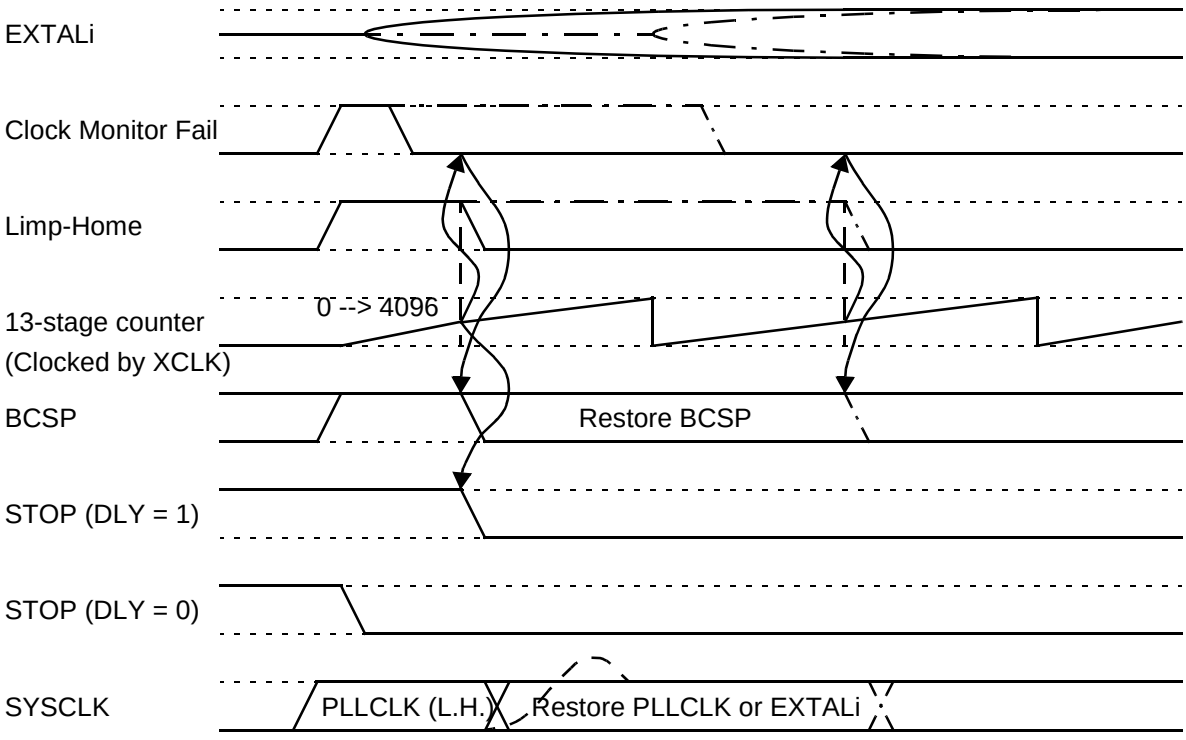


Figure 17 STOP Exit and Fast STOP Recovery

STOP exit without Limp Home mode, clock monitor disabled

(NOLHM=1, CME=0, DLY=X)

If Limp home mode is disabled (V_{DDPLL}=V_{SS} or NOLHM bit set) and the CME (or FCME) bit is cleared, the MCU goes into STOP mode when a STOP instruction is executed.

If EXTALi clock is present then exit from STOP will occur normally using this clock. Under this condition, DLY should always be set to allow the crystal to stabilise and minimise the risk of code runaway. With DLY=1 execution resumes after a delay of 4096 XCLK cycles.

NOTE: *The external clock signal should stabilise within the 4096 reset counter cycles.* ***Use of DLY=0 is not recommended due to this requirement.***

Executing the STOP instruction without Limp Home mode, clock monitor enabled

(NOLHM=1, CME=1, DLY=X)

If the NOLHM bit and the CME (or FCME) bits are set, a clock monitor failure is detected when a STOP instruction is executed and the MCU resets via the clock monitor reset vector.

STOP exit in Limp Home mode with Delay

(NOLHM=0, CME=X, DLY=1)

If the NOLHM bit is cleared, then the CME (or FCME) bit is masked when a STOP instruction is executed to prevent a clock monitor failure. When coming out of STOP mode, the MCU goes into limp-home mode where CME and FCME signals are asserted.

When using a crystal oscillator, a normal STOP exit sequence requires the DLY bit to be set to allow for the crystal stabilization period.

With the 13-stage counter clocked by the VCO (at f_{VCOMIN}), following a delay of 4096 XCLK cycles at the limp-home frequency, if the clock monitor indicates the presence of an external clock, the limp-home mode is de-asserted and the MCU exits STOP normally using EXTALi clock. Where the crystal start-up time is longer than the initial count of 4096 XCLK cycles, or in the absence of an external clock, the MCU recovers from STOP following the 4096 count in limp-home mode with both the LHOME flag set and the LHIF limp-home interrupt request set to indicate

it is not operating at the desired frequency. Each time the 13-stage counter reaches a count of 4096 XCLK cycles, a check of the clock monitor status is performed.

When the presence of an external clock is detected, limp-home mode is exited and the LHOME flag is cleared. This sets the limp-home interrupt flag and if enabled by the LHIE bit, the limp-home mode interrupt is requested.

CAUTION: *The clock monitor circuit can be misled by EXTALi clock into reporting a good signal before it has fully stabilised. Under these conditions, improper EXTALi clock cycles can occur on SYSCLK. This may lead to a code runaway.*

STOP exit in Limp Home mode without Delay (Fast Stop Recovery)

(NOLHM=0, CME=X, DLY=0)

Fast STOP recovery refers to any exit from STOP using DLY=0.

If the NOLHM bit is cleared, then the CME (or FCME) bit is masked when a STOP instruction is executed to prevent a clock monitor failure. When coming out of STOP mode, the MCU goes into limp-home mode where CME and FCME signals are asserted.

When using a crystal oscillator, it is possible to exit STOP with the DLY bit cleared. In this case, STOP is de-asserted without delay and the MCU will execute software in limp-home mode, giving the crystal oscillator time to stablise.

CAUTION: *This mode is not recommended since the risk of the clock monitor detecting incorrect clocks is high.*

Each time the 13-stage counter reaches a count of 4096 XCLK cycles (every 8192 cycles), a check of the clock monitor status is performed. If the clock monitor indicates the presence of an external clock limp-home mode is de-asserted, the LHOME flag is cleared and the limp-home interrupt flag is set. Upon leaving limp-home mode, BCSP and MCS are restored to their values before the loss of clock, and all clocks return to their previous frequencies. If AUTO and BCSP were set before the clock loss, the SYSCLK ramps-up and the PLL locks at the previously selected frequency.

To prevent PLL operation when the external clock frequency comes back, the software should clear the BCSP bit while running in limp-home mode.

When using an external clock, i.e. a square wave source, it is possible to exit STOP with the DLY bit cleared. In this case the LHOME flag is never set and STOP is de-asserted without delay.

Pseudo-STOP

Pseudo-STOP is a low power mode similar to STOP where the external oscillator is allowed to run (at reduced amplitude) whilst the rest of the part is in STOP. This increases the current consumption over STOP mode by the amount of current in the oscillator, but reduces wear and mechanical stress on the crystal.

If the PSTP bit in the PLLCR register is set, the MCU goes into Pseudo-STOP mode when a STOP instruction is executed.

Pseudo-STOP mode is exited the same as STOP with an external reset, an external interrupt from IRQ or XIRQ, a Key Wake-Up interrupt from port J or port H, or an MSCAN Wake-Up interrupt.

The effect of the DLY bit is the same as noted above in STOP Exit and Fast STOP Recovery.

Pseudo-STOP exit in Limp Home mode with Delay

(NOLHM=0, CME=X, DLY=1)

When coming out of Pseudo-STOP mode with the NOLHM bit cleared and the DLY bit set, the MCU goes into limp-home mode (regardless of the state of the CME or FCME bits).

The VCO supplies the limp-home clock frequency to the 13-stage counter (XCLK). The BCSP output is forced high and MCS is forced low. After the 13-stage counter reaches a count of 4096 XCLK cycles, a check of the clock monitor is performed and as the crystal oscillator was kept running due to the Pseudo-stop mode, the MCU exits STOP normally, using the EXTALi clock. In the case where a crystal failure occurred during pseudo-stop, then the MCU exits STOP using the limp home clock (f_{VCOMIN}) with both the LHOME flag set and the LHIF limp-home interrupt request set to indicate it is not operating at the desired frequency. Each time the 13-stage counter reaches a count of 4096 XCLK cycles, a check of the clock monitor is performed. If the clock monitor indicates the presence of an external clock, limp-home mode is de-asserted, the LHOME flag is cleared and the LHIF limp-home interrupt request is set to indicate a return to normal operation using EXTALi clock.

Pseudo-STOP exit in Limp Home mode without Delay (Fast Stop Recovery)

(NOLHM=0, CME=X, DLY=0)

If Pseudo-STOP is exited with the NOLHM bit set to 0 and the DLY bit is cleared then the exit from Pseudo-STOP is accomplished without delay as in Fast STOP recovery.

CAUTION: *Where Pseudo-STOP recovers using the Limp Home Clock the VCO - which has been held in STOP - must be restarted in order to supply the limp home frequency. This restart, which occurs at a high frequency and ramps toward the limp home frequency, is almost immediately supplied to the CPU before it may have reached the steady state frequency. It is possible that the initial clock frequency may be high enough to cause the CPU to function incorrectly with a resultant risk of code runaway.*

Pseudo-STOP exit without Limp Home mode, clock monitor enabled

(NOLHM=1, CME=1, DLY=X)

If the NOLHM bit is set and the CME (or FCME) bits are set, a clock monitor failure is detected when a STOP instruction is executed and the MCU resets via the clock monitor reset vector.

Pseudo-STOP exit without Limp Home mode, clock monitor disabled

(NOLHM=1, CME=0, DLY=1)

If NOLHM is set to 1 and the CME and FCME bits are cleared, the limp home clock is not used. In this mode, crystal activity is the only method by which the device may recover from Pseudo-STOP. The device will start execution with the EXTALi clock following 4096 XCLK cycles.

(NOLHM=1, CME=0, DLY=0)

If NOLHM is set to 1 and the CME and FCME bits are cleared, the limp home clock is not used. In this mode, crystal activity is the only method by which the device may recover from Pseudo-STOP. The device will start execution with the EXTALi clock following 16 XCLK cycles.

CAUTION: *Due to switching of the clock this configuration is not recommended.*

Summary of STOP and pseudo-STOP Mode Exit Conditions

Table 23 and Table 24 summarise the exit conditions from STOP and pseudo-STOP modes using Interrupt, Key-interrupt and XIRQ.

A short RESET pulse should not be used to exit stop or pseudo-STOP mode because Limp Home mode is automatically entered after RESET (when V_{DDPLL}=V_{DD}). The RESET wakeup pulse must be longer than the oscillator startup time (as in power on reset) in order to remove the risk of code runaway.

. .

Table 23 Summary of STOP Mode Exit Conditions

Mode	Conditions	Summary
STOP exit without Limp Home mode, clock monitor disabled	NOLHM=1 CME=0 DLY=X	Oscillator must be stable within 4096 XCLK cycles. XCLK can be modified by SLOW divider register. Use of DLY=0 only recommended with external clock.
Executing the STOP instruction without Limp Home mode, clock monitor enabled	NOLHM=1 CME=1 DLY=X	When a STOP instruction is executed the MCU resets via the clock monitor reset vector.
STOP exit in Limp Home mode with Delay	NOLHM=0 CME=X DLY=1	Oscillator must be stable within 4096 f_{VCOMIN} cycles or there is a possibility of code runaway as the clock monitor circuit can be misled by EXTALi clock into reporting a good signal before it has fully stabilised
STOP exit in Limp Home mode without Delay (Fast Stop Recovery)	NOLHM=0 CME=X DLY=0	This mode is only recommended for use with an external clock source.

Table 24 Summary of Pseudo STOP Mode Exit Conditions

Mode	Conditions	Summary
Pseudo-STOP exit in Limp Home mode with Delay	NOLHM=0 CME=X DLY=1	CPU exits stop in limp home mode and oscillator running. If the oscillator fails during pseudo-STOP and then recovers there is a possibility of code runaway as the clock monitor circuit can be misled by EXTALi clock into reporting a good signal before it has fully stabilised
Pseudo-STOP exit in Limp Home mode without Delay (Fast Stop Recovery)	NOLHM=0 CME=X DLY=0	This mode is not recommended as it is possible that the initial VCO clock frequency may be high enough to cause code runaway.
Pseudo-STOP exit without Limp Home mode, clock monitor enabled	NOLHM=1 CME=1 DLY=X	When a STOP instruction is executed the MCU resets via the clock monitor reset vector.
Pseudo-STOP exit without Limp Home mode, clock monitor disabled, with Delay	NOLHM=1 CME=0 DLY=1	Oscillator starts operation following 4096 XCLK cycles (actual controlled by SLOW mode divider).
Pseudo-STOP exit without Limp Home mode, clock monitor disabled, without Delay	NOLHM=1 CME=0 DLY=0	This mode is only recommended for use with an external clock source.

PLL Register Descriptions

SYNR — Synthesizer Register **$0038**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0
RESET:	0	0	0	0	0	0	0	0

Read anytime, write anytime, except when BCSP = 1 (PLL selected as bus clock).

If the PLL is on, the count in the loop divider (SYNR) register effectively multiplies up the bus frequency from the PLL reference frequency by SYNR + 1. Internally, SYSCLK runs at twice the bus frequency. Caution should be used not to exceed the maximum rated operating frequency for the CPU.

REFDV — Reference Divider Register **$0039**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	REFDV2	REFDV1	REFDV0
RESET:	0	0	0	0	0	0	0	0

Read anytime, write anytime, except when BCSP = 1.

The reference divider bits provides a finer granularity for the PLL multiplier steps. The reference frequency is divided by REFDV + 1.

CGTFLG — Clock Generator Test Register **$003A**

	Bit 7	6	5	4	3	2	1	Bit 0
	TSTOUT7	TSTOUT6	TSTOUT5	TSTOUT4	TSTOUT3	TSTOUT2	TSTOUT1	TSTOUT0
RESET:	0	0	0	0	0	0	0	0

Always reads zero, except in test modes.

PLLFLG — PLL Flags **$003B**

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

Read anytime, refer to each bit for write conditions.

LOCKIF — PLL Lock Interrupt Flag

0 = No change in LOCK bit.

1 = LOCK condition has changed, either from a locked state to an unlocked state or vice versa.

To clear the flag, write one to this bit in PLLFLG. Cleared in limp-home mode.

LOCK — Locked Phase Lock Loop Circuit

Regardless of the bandwidth control mode (automatic or manual):

0 = PLL VCO is not within the desired tolerance of the target frequency.

1 = After the phase lock loop circuit is turned on, indicates the PLL VCO is within the desired tolerance of the target frequency.

Write has no effect on LOCK bit. This bit is cleared in limp-home mode as the lock detector cannot operate without the reference frequency.

LHIF — Limp-Home Interrupt Flag

0 = No change in LHOME bit.

1 = LHOME condition has changed, either entered or exited limp-home mode.

To clear the flag, write one to this bit in PLLFLG.

LHOME — Limp-Home Mode Status

0 = MCU is operating normally, with EXTALi clock available for generating clocks or as PLL reference.

1 = Loss of reference clock. CGM delivers PLL VCO limp-home frequency to the MCU.

For Limp-Home mode, see Limp-Home and Fast STOP Recovery modes.

PLLCR — PLL Control Register **$003C**

	Bit 7	6	5	4	3	2	1	Bit 0
	LOCKIE	PLLON	AUTO	$\overline{\text{ACQ}}$	0	PSTP	LHIE	NOLHM
RESET:	0	__(1)	1	0	0	0	0	__(2)

1. Set when VDDPLL power supply is high. Forced to 0 when VDDPLL is low.
2. Cleared when VDDPLL power supply is high. Forced to 1 when VDDPLL is low.

Read and write anytime. Exceptions are listed below for each bit.

LOCKIE — PLL LOCK Interrupt Enable

0 = PLL LOCK interrupt is disabled
1 = PLL LOCK interrupt is enabled

Forced to 0 when VDDPLL=0.

PLLON — Phase Lock Loop On

0 = Turns the PLL off.
1 = Turns on the phase lock loop circuit. If AUTO is set, the PLL will lock automatically.

Cannot be cleared when BCSP = 1 (PLL selected as bus clock). Forced to 0 when VDDPLL is at VSS level. In limp-home mode, the output of PLLON is forced to 1, but the PLLON bit reads the latched value.

AUTO — Automatic Bandwidth Control

0 = Automatic Mode Control is disabled and the PLL is under software control, using $\overline{\text{ACQ}}$ bit.
1 = Automatic Mode Control is enabled. $\overline{\text{ACQ}}$ bit is read only.

Automatic bandwidth control selects either the high bandwidth (acquisition) mode or the low bandwidth (tracking) mode depending on how close to the desired frequency the VCO is running. See **Electrical Characteristics**.

$\overline{ACQ}$ — Not in Acquisition

If AUTO = 1 ($\overline{ACQ}$ is Read Only)

0 = PLL VCO is not within the desired tolerance of the target frequency. The loop filter is in high bandwidth, acquisition mode.

1 = After the phase lock loop circuit is turned on, indicates the PLL VCO is within the desired tolerance of the target frequency. The loop filter is in low bandwidth, tracking mode.

If AUTO = 0

0 = High bandwidth PLL loop selected

1 = Low bandwidth PLL loop selected

PSTP — Pseudo-STOP Enable

0 = Pseudo-STOP oscillator mode is disabled

1 = Pseudo-STOP oscillator mode is enabled

In Pseudo-STOP mode, the oscillator is still running while the MCU is maintained in STOP mode. This allows for a faster STOP recovery and reduces the mechanical stress and aging of the resonator in case frequent STOP conditions at the expense of a slightly increased power consumption.

LHIE — Limp-Home Interrupt Enable

0 = Limp-Home interrupt is disabled

1 = Limp-Home interrupt is enabled

Forced to 0 when VDDPLL is at VSS level.

NOLHM —No Limp-Home Mode

0 = Loss of reference clock forces the MCU in limp-home mode.

1 = Loss of reference clock causes standard Clock Monitor reset.

Read anytime; Normal modes: write once; Special modes: write anytime. Forced to 1 when VDDPLL is at VSS level.

CLKSEL — Clock Generator Clock select Register **$003D**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	BCSP	BCSS	0	0	MCS	0	0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime. Exceptions are listed below for each bit.

BCSP and BCSS bits determine the clock used by the main system including the CPU and buses.

BCSP — Bus Clock Select PLL

0 = SYSCLK is derived from the crystal clock or from SLWCLK.
1 = SYSCLK source is the PLL.

Cannot be set when PLLON = 0. In limp-home mode, the output of BCSP is forced to 1, but the BCSP bit reads the latched value.

BCSS — Bus Clock Select Slow

0 = SYSCLK is derived from the crystal clock EXTALi.
1 = SYSCLK source is the Slow clock SLWCLK.

This bit has no effect when BCSP is set.

MCS — Module Clock Select

0 = M clock is the same as PCLK.
1 = M clock is derived from Slow clock SLWCLK.

This bit determines the clock used by the ECT module and the baud rate generators of the SCIs. In limp-home mode, the output of MCS is forced to 0, but the MCS bit reads the latched value.

SLOW — Slow mode Divider Register **$003E**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	SLDV5	SLDV4	SLDV3	SLDV2	SLDV1	SLDV0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime.

A write to this register changes the SLWCLK frequency with minimum delay (less than one SLWCLK cycle), thus allowing immediate tune-up of the performance versus power consumption for the modules using this clock. The frequency divide ratio is 2 times (SLOW), hence the divide range is 2 to 126 (not on first pass products). When SLOW = 0, the divider is bypassed. The generation of E, P and M clocks further divides SLWCLK by 2. Hence, the final ratio of Bus to EXTALi Frequency is programmable to 2, 4, 8, 12, 16, 20, ..., 252, by steps of 4. SLWCLK is a 50% duty cycle signal.

System Clock Frequency Formulae

See Figure 18:

SLWCLK = EXTALi / (2 x SLOW) SLOW = 1,2,..63

SLWCLK = EXTALi SLOW = 0

PLLCLK = 2 x EXTALi x (SYNR + 1) / (REFDV + 1)

ECLK = SYSCLK / 2

XCLK = SLWCLK / 2

PCLK = SYSCLK / 2

BCLK[1] = EXTALi / 2

Boolean equations:

SYSCLK = (BCSP & PLLCLK) | ($\overline{\text{BCSP}}$ & $\overline{\text{BCSS}}$ & EXTALi) | ($\overline{\text{BCSP}}$ & BCSS & SLWCLK)

MCLK = (PCLK & $\overline{\text{MCS}}$) | (XCLK & MCS)

MSCAN system = (EXTALi & $\overline{\text{CLKSRC}}$) | (SYSCLK & CLKSRC)

BDM system = (BCLK & $\overline{\text{CLKSW}}$) | (ECLK & CLKSW)

NOTE: *During limp-home mode PCLK, ECLK, BCLK, MCLK and XCLK are supplied by VCO (PLLCLK).*

1. If SYSCLK is slower than EXTALi (BCSS=1, BCSP=0, SLOW>0), BCLK becomes ECLK.

Clock Divider Chains

Figure 18, Figure 19, Figure 20, and Figure 21 summarize the clock divider chains for the various peripherals on the 68HC912D60A.

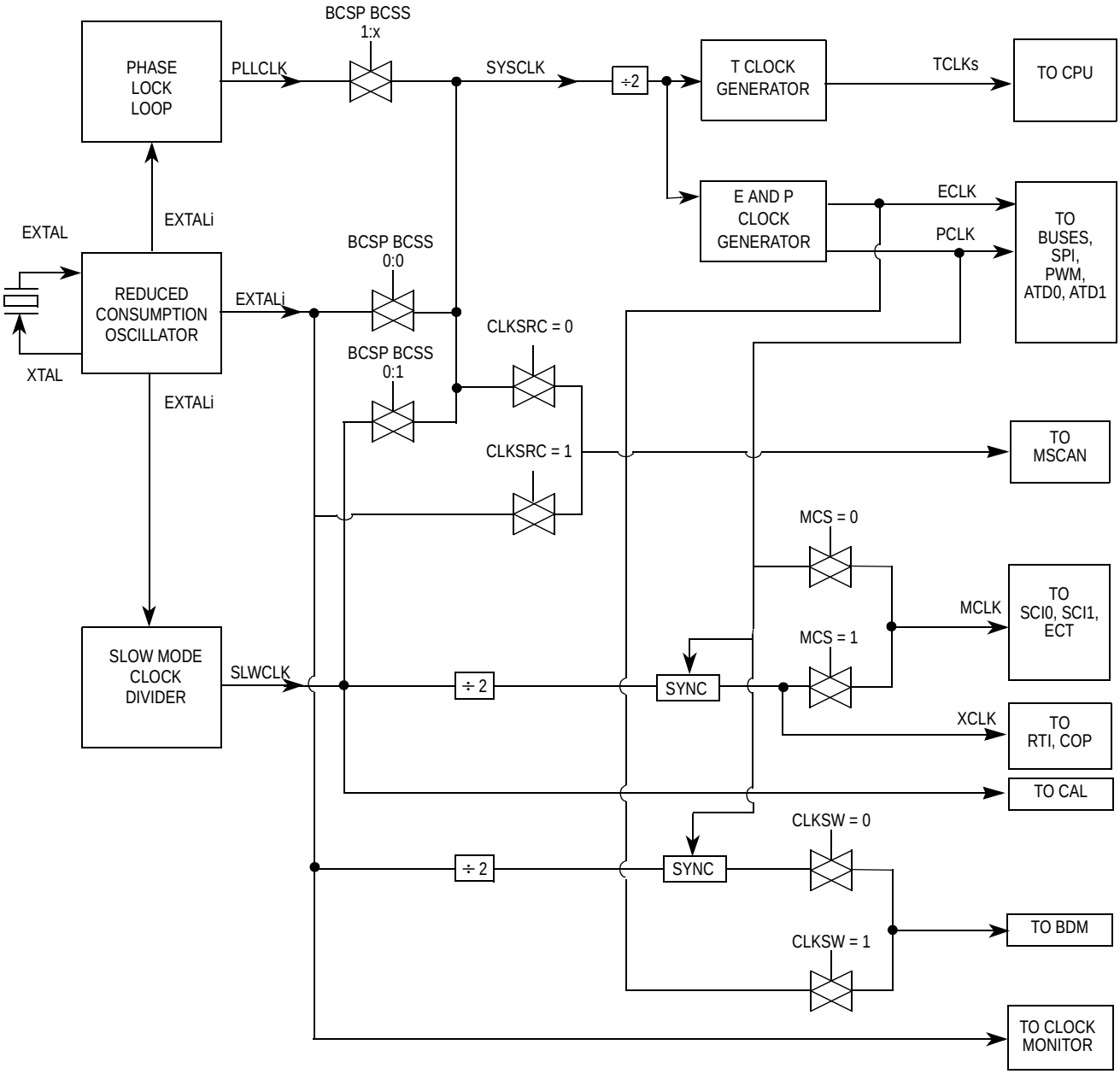


Figure 18 Clock Generation Chain

Bus clock select bits BCSP and BCSS in the clock select register (CLKSEL) determine which clock drives SYSCLK for the main system including the CPU and buses. BCSS has no effect if BCSP is set. During the transition, the clock select output will be held low and all CPU activity will cease until the transition is complete.

The Module Clock Select bit MCS determines the clock used by the ECT module and the baud rate generators of the SCIs. In limp-home mode, the output of MCS is forced to 0, but the MCS bit reads the latched value. It allows normal operation of the serial and timer subsystems at a fixed reference frequency while allowing the CPU to operate at a higher, variable frequency.

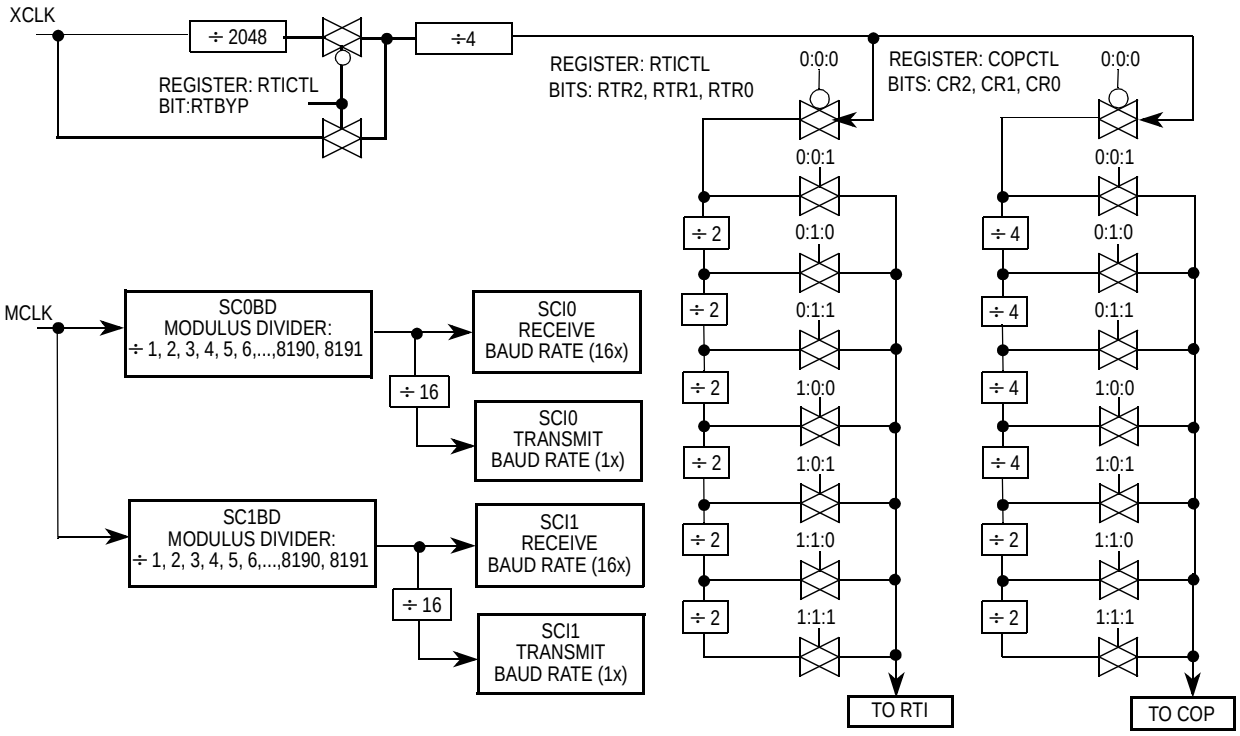


Figure 19 Clock Chain for SCI0, SCI1, RTI, COP

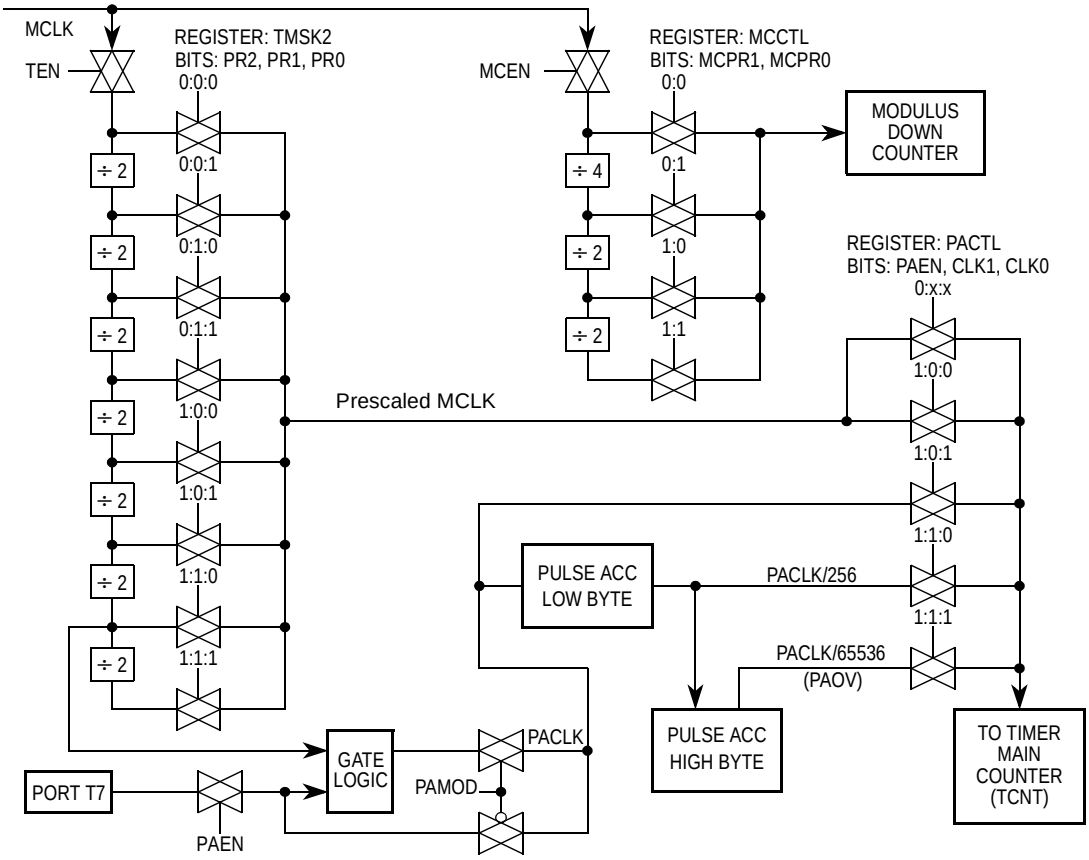


Figure 20 Clock Chain for ECT

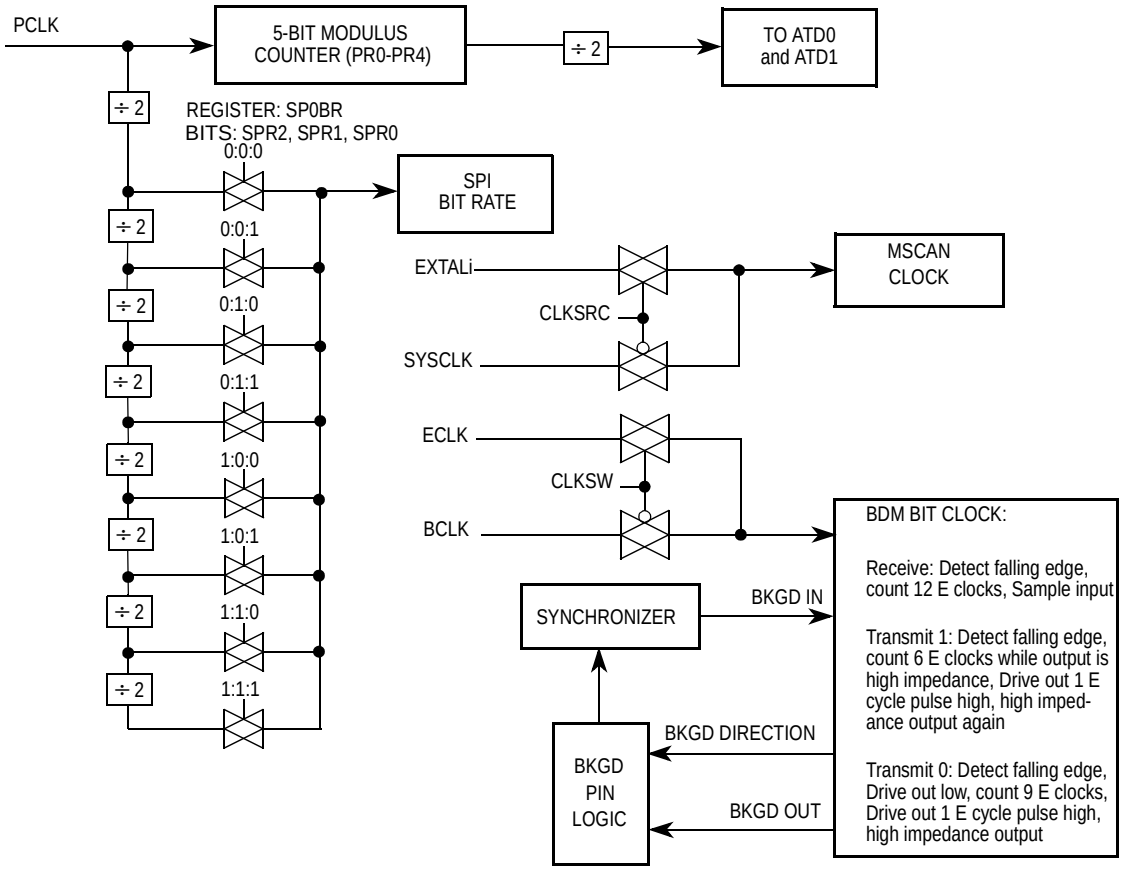


Figure 21 Clock Chain for MSCAN, SPI, ATD0, ATD1 and BDM

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. 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, the part is reset.

In addition, windowed COP operation can be selected. In this mode, writes to the COPRST register must occur in the last 25% of the selected period. A premature write will also reset the part.

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.

Clock Monitor

The clock monitor circuit is based on an internal resistor-capacitor (RC) time delay. If no EXTALi 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 EXTALi clock.

Clock monitor time-outs are shown in Table 25. The corresponding EXTALi clock period with an ideal 50% duty cycle is twice this time-out value.

Table 25 Clock Monitor Time-Outs

Supply	Range
5 V +/– 10%	2–20 µS

Clock Function Registers

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

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

	Bit 7	6	5	4	3	2	1	Bit 0
	RTIE	RSWAI	RSBCK	Reserved	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 when 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 XCLK divided by 2^{13}, when bypassed becomes XCLK 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 XCLK.

Table 26 Real Time Interrupt Rates

RTR2	RTR1	RTR0	Divide X By:	Time-Out Period X = 125 KHz	Time-Out Period X = 500 KHz	Time-Out Period X = 2.0 MHz	Time-Out Period X = 8.0 MHz
0	0	0	OFF	OFF	OFF	OFF	OFF
0	0	1	2^{13}	65.536 ms	16.384 ms	4.096 ms	1.024 ms
0	1	0	2^{14}	131.72 ms	32.768 ms	8.196 ms	2.048 ms
0	1	1	2^{15}	263.44 ms	65.536 ms	16.384 ms	4.096 ms
1	0	0	2^{16}	526.88 ms	131.72 ms	32.768 ms	8.196 ms
1	0	1	2^{17}	1.05 s	263.44 ms	65.536 ms	16.384 ms
1	1	0	2^{18}	2.11 s	526.88 ms	131.72 ms	32.768 ms
1	1	1	2^{19}	4.22 s	1.05 s	263.44 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	FCMCOP	WCOP	DISR	CR2	CR1	CR0	
RESET:	0/1	0	0	0	0	1	1	1	Normal
RESET:	0/1	0	0	0	1	1	1	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 or limp-home mode. See Limp-Home and Fast STOP Recovery modes.

On reset

CME is 1 if VDDPLL is high
CME is 0 if VDDPLL is low.

NOTE: *The VDDPLL-dependent reset operation is not implemented on first pass products.*
In this case the state of CME on reset is 0.

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 or limp-home mode.

See Limp-Home and Fast STOP Recovery modes.

FCMCOP — Force Clock Monitor Reset or COP Watchdog 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 = A clock monitor failure reset or a COP failure reset is forced depending on the state of CME and if COP is enabled.

CME	COP enabled	Forced reset
0	0	none
0	1	COP failure
1	0	Clock monitor failure
1	1	Both(1)

1. Highest priority interrupt vector is serviced.

WCOP — Window COP mode

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

0 = Normal COP operation

1 = Window COP operation

When set, a write to the COPRST register must occur in the last 25% of the selected period. A premature write will also reset the part. As long as all writes occur during this window, $55 can be written as often as desired. Once $AA is written the time-out logic restarts and the user must wait until the next window before writing to COPRST. Please note, there is a fixed time uncertainty about the exact COP counter state when reset, as the initial prescale clock divider in the RTI section is not cleared when the COP counter is cleared. This means the effective window is reduced by this uncertainty. Table 27 below shows the exact duration of this window for the seven available COP rates.

Table 27 COP Watchdog Rates

CR2	CR1	CR0	Divide XCLK by	8.0 MHz XCLK Time-out	Window COP enabled:		
					Window start (1)	Window end	Effective Window (2)
0	0	0	OFF	OFF	OFF	OFF	OFF
0	0	1	2^{13}	1.024 ms -0/+0.256 ms	0.768 ms	0.768 ms	0 % (3)
0	1	0	2^{15}	4.096 ms -0/+0.256 ms	3.072 ms	3.840 ms	18.8 %
0	1	1	2^{17}	16.384 ms -0/+0.256 ms	12.288 ms	16.128 ms	23.4 %
1	0	0	2^{19}	65.536 ms -0/+1.024 ms	49.152 ms	64.512 ms	23.4 %
1	0	1	2^{21}	262.144 ms -0/+1.024 ms	196.608 ms	261.120 ms	24.6 %
1	1	0	2^{22}	524.288 ms -0/+1.024 ms	393.216 ms	523.264 ms	24.8 %
1	1	1	2^{23}	1.048576 ms -0/+1.024 ms	786.432 ms	1.047552 ms	24.9 %

1. Time for writing $55 following previous COP restart of time-out logic due to writing $AA.
2. Please refer to WCOP bit description above.
3. Window COP cannot be used at this rate.

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

These bits select the COP time-out rate. The clock used for this module is the XCLK.

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

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.

Pulse-Width Modulator

Contents

Introduction

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 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 rolls over or the channel is disabled. If the channel is not enabled, then writes to the period and/or duty register will go directly to the latches as well as the buffer, thus ensuring that the PWM output will always be either the old waveform or the new waveform, not some variation in between.

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. 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.

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.

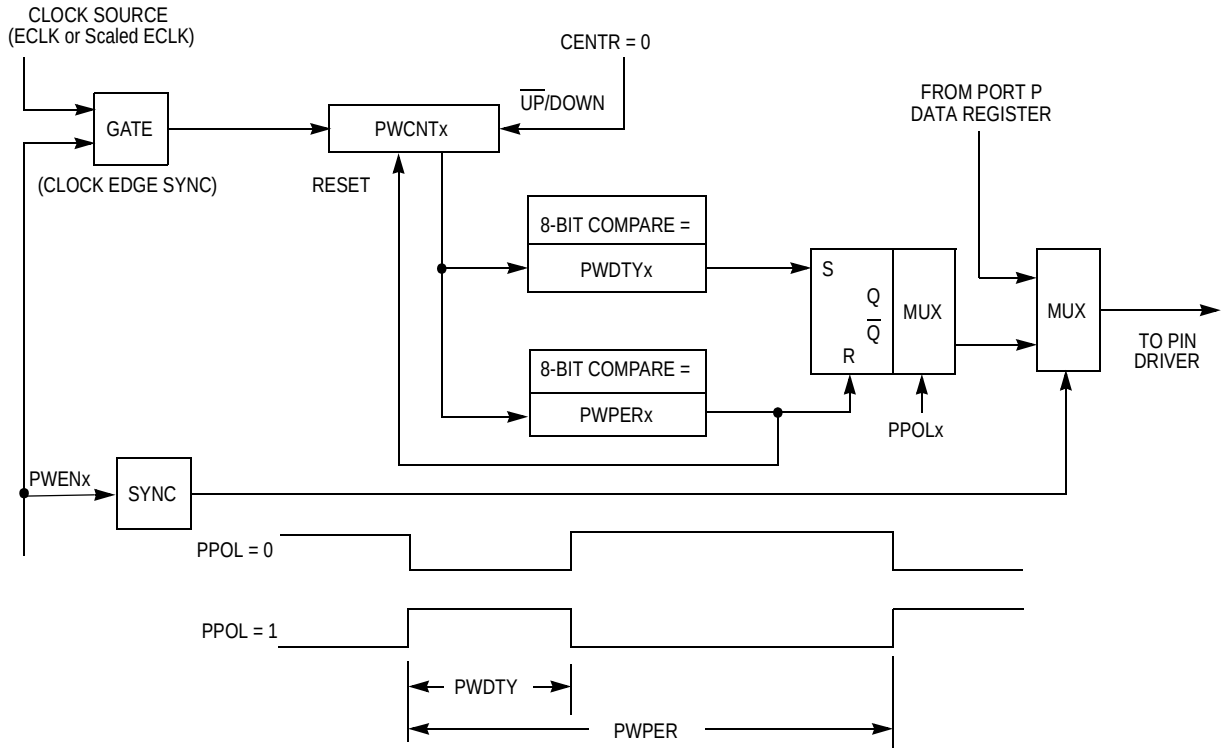


Figure 22 Block Diagram of PWM Left-Aligned Output Channel

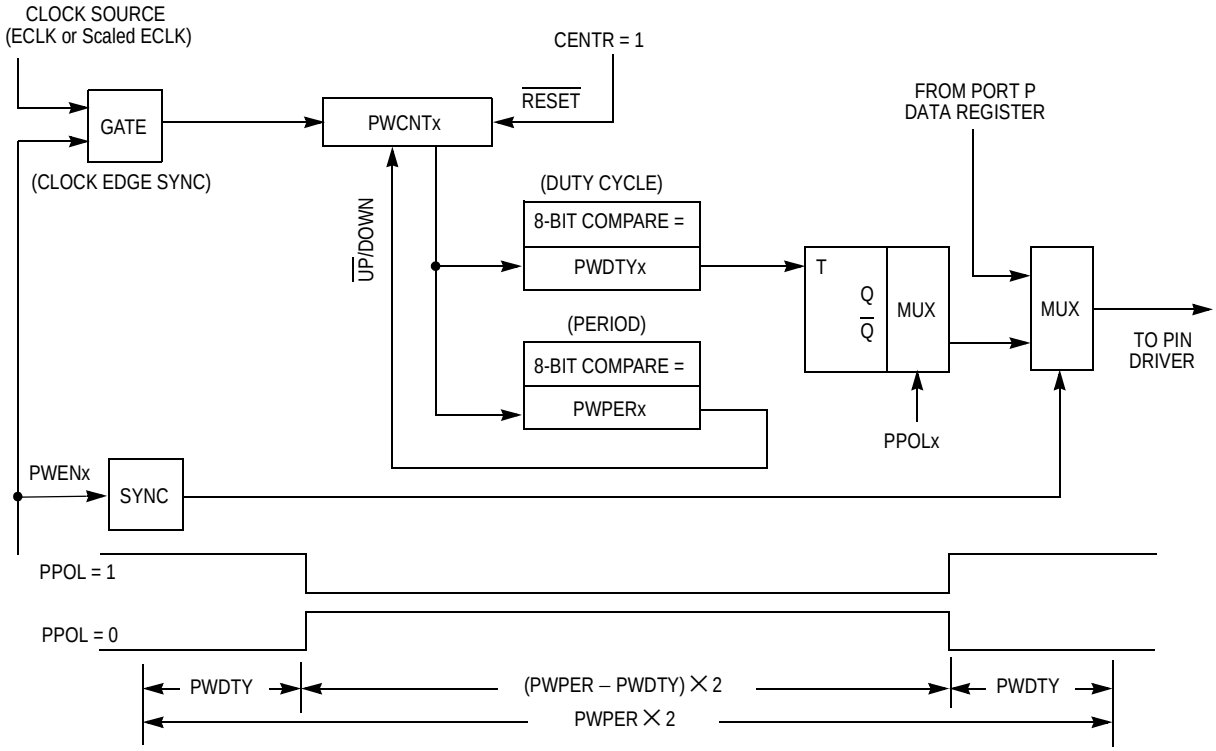


Figure 23 Block Diagram of PWM Center-Aligned Output Channel

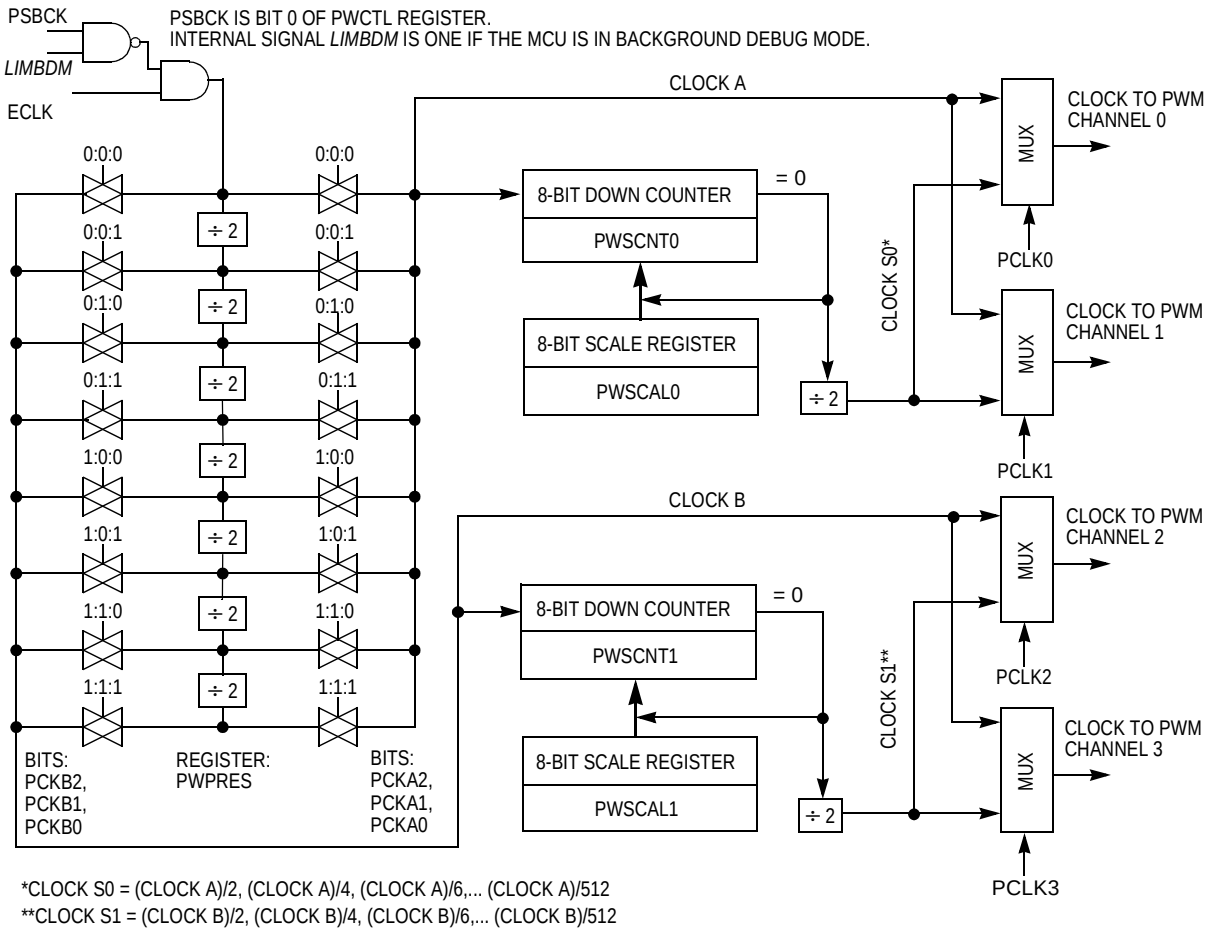


Figure 24 PWM Clock Sources

PWM Register Descriptions

PWCLK — PWM Clocks and Concatenate **$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. Channel 3 output pin becomes a general purpose I/O.

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 determine the clock source. Channel 1 output pin becomes a general purpose I/O.

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

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 28.

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 28.

Table 28 Clock A and Clock B Prescaler

PCKA2 (PCKB2)	PCKA1 (PCKB1)	PCKA0 (PCKB0)	Value of Clock A (B)
0	0	0	P
0	0	1	P ÷ 2
0	1	0	P ÷ 4
0	1	1	P ÷ 8
1	0	0	P ÷ 16
1	0	1	P ÷ 32
1	1	0	P ÷ 64
1	1	1	P ÷ 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.

If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse may occur during the transition.

The following four bits apply in left-aligned mode only:

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.

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 zero and left alignment is selected, the duty registers contain a count of the low time. If the polarity bit is one, the duty registers contain a count of the high 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

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 + 1 and dividing again by two. When PWSCAL0 = $FF, clock A is divided by 256 then divided by two to generate clock S0.

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 + 1 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.

The PWM counters are not reset when PWM channels are disabled. The counters must be reset prior to a new enable.

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. To avoid a truncated PWM period, write to a counter while the counter is disabled. In left-aligned output mode, resetting the counter and starting the waveform output is controlled by a match between the period register and the value in the counter. In center-aligned output mode the counters operate as up/down counters, where a match in period changes the counter direction. The duty register changes the state of the output during the period to determine the duty.

When a channel is enabled, the associated PWM counter starts at the count in the PWCNTx register using the clock selected for that channel.

In special mode, when DISCP = 1 and configured for left-aligned output, a match of period does not reset the associated PWM counter.

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.

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. 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 × (PWPER + 1) (CENTR = 0)
Period = Channel-Clock-Period × (2 × PWPER) (CENTR = 1)

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. When the duty value is equal to the counter value, the output changes state. If the register is written while the channel is enabled, the new value is held in a buffer until the counter rolls over or the channel is disabled. Reading this register returns the most recent value written.

If the duty register is greater than or equal to the value in the period register, there will be no duty change in state. If the duty register is set to $FF the output will always be in the state which would normally be the state opposite the PPOLx value.

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

PWCTL — PWM Control Register **$0054**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	PSWAI	CENTR	RDPP	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

To avoid irregularities in the PWM output mode, write the CENTR bit only when PWM channels are disabled.

0 = PWM channels operate in left-aligned output mode

1 = PWM channels operate in center-aligned output mode

RDPP — 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.

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 (SMODN = 0). These bits are available only in special mode and are reset in normal mode.

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

PORTP — 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:	–	–	–	–	–	–	–	–

PORTP can be read anytime.

PWM functions share port P pins 3 to 0 and take precedence over the general-purpose port when enabled.

When configured as input, a read will return the pin level. For port bits 7 to 4 it will read zero because there are no available external pins.

When configured as output, a read will return the latched output data. For port bits 7 to 4 it will read the last value written. A write will drive associated pins only if configured for output and the corresponding PWM channel is not enabled.
After reset, all pins are general-purpose, high-impedance inputs.

DDRP — 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

DDRP determines pin direction of port P when used for general-purpose I/O.

DDRP[7:4] — This bits served as memory locations since there are no corresponding port pins.

DDRP[3:0] — Data Direction Port P pin 0-3
0 = I/O pin configured as high impedance input
1 = I/O pin configured for output.

PWM Boundary Cases

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

Table 29 PWM Left-Aligned 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

Table 30 PWM Center-Aligned Boundary Conditions

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

Enhanced Capture Timer

Contents

Introduction

The HC12 Enhanced Capture Timer module has the features of the HC12 Standard Timer module enhanced by additional features in order to enlarge the field of applications, in particular for automotive ABS applications.

The additional features permit the operation of this timer module in a mode similar to the Input Control Timer implemented on MC68HC11NB4.

These additional features are:

- 16-Bit Buffer Register for four Input Capture (IC) channels.
- Four 8-Bit Pulse Accumulators with 8-bit buffer registers associated with the four buffered IC channels. Configurable also as two 16-Bit Pulse Accumulators.
- 16-Bit Modulus Down-Counter with 4-bit Prescaler.
- Four user selectable Delay Counters for input noise immunity increase.
- Main Timer Prescaler extended to 7-bit.

This design specification describes the standard timer as well as the additional features.

The basic timer consists of a 16-bit, software-programmable counter driven by a prescaler. This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

A full access for the counter registers or the input capture/output compare registers should take place in one clock cycle. Accessing high byte and low byte separately for all of these registers may not yield the same result as accessing them in one word.

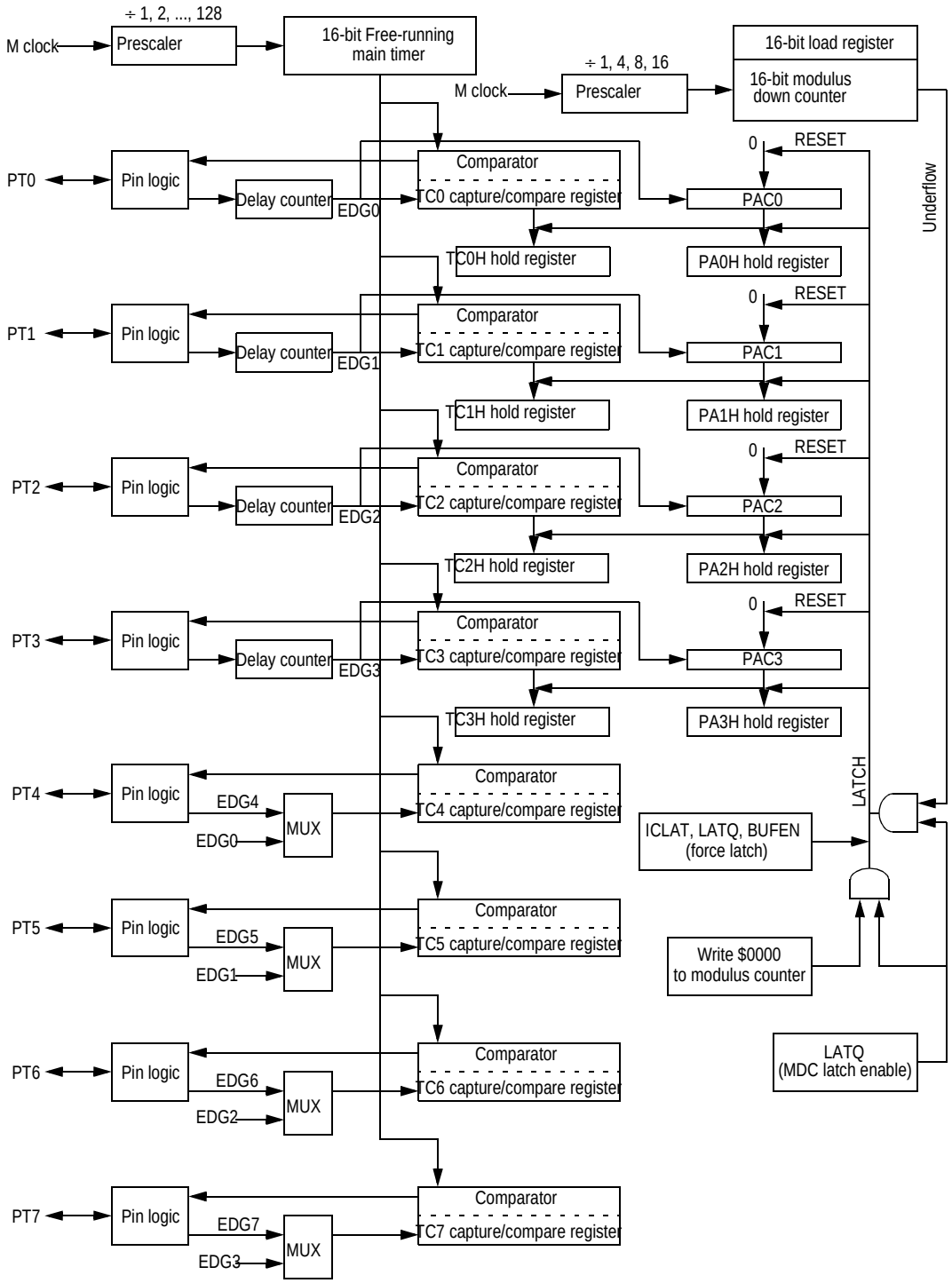


Figure 25 Timer Block Diagram in Latch Mode

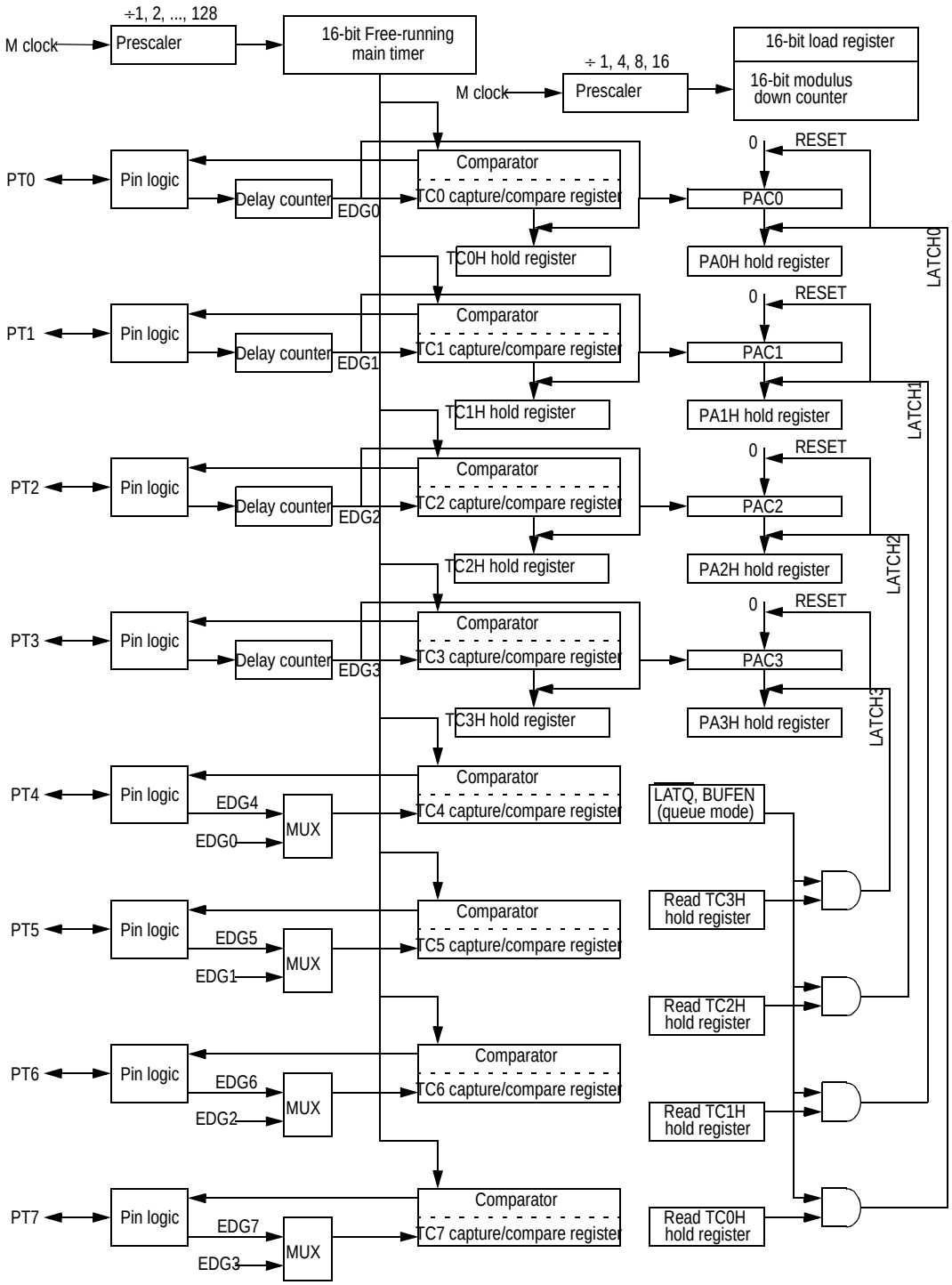


Figure 26 Timer Block Diagram in Queue Mode

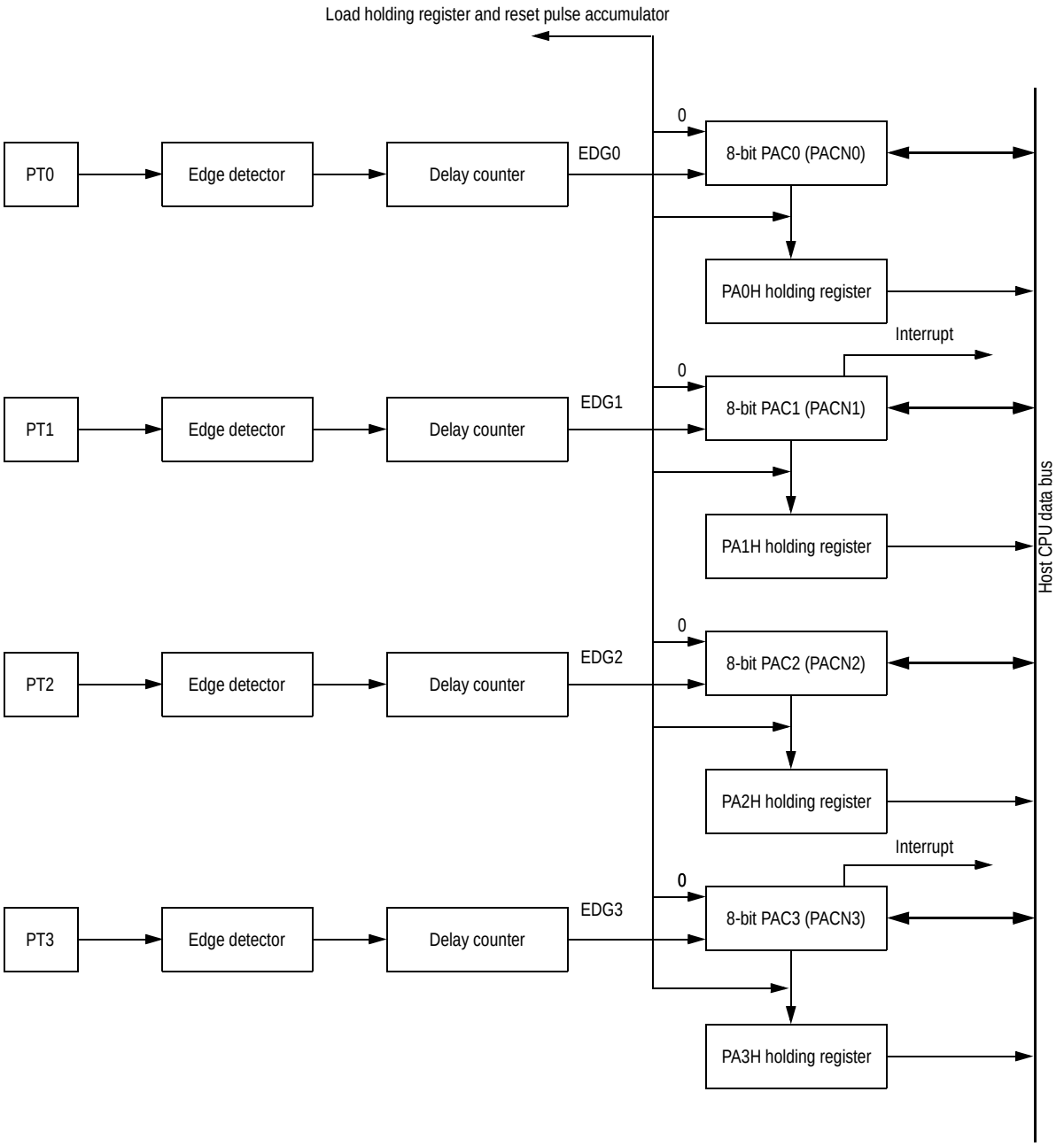


Figure 27 8-Bit Pulse Accumulators Block Diagram

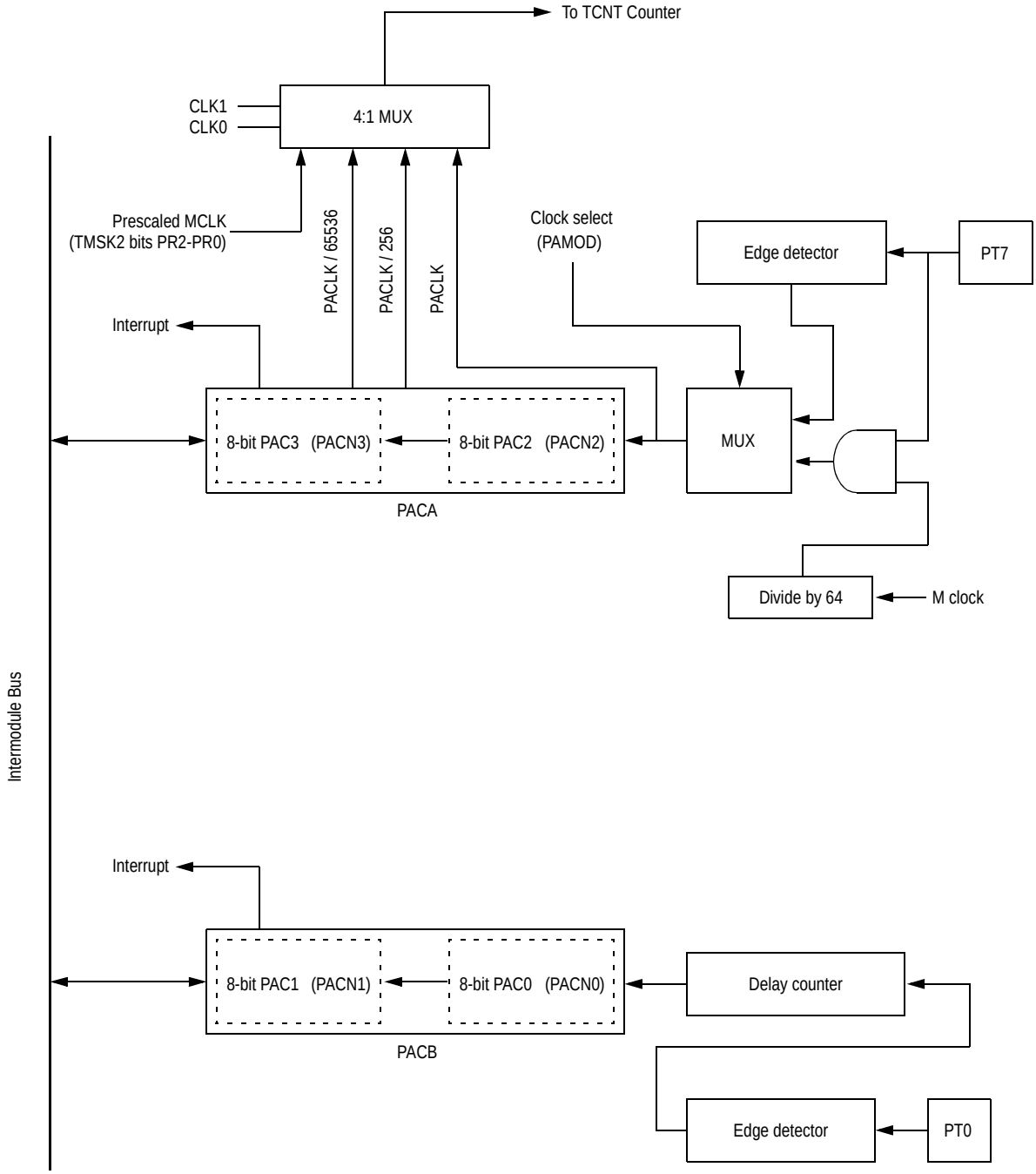


Figure 28 16-Bit Pulse Accumulators Block Diagram

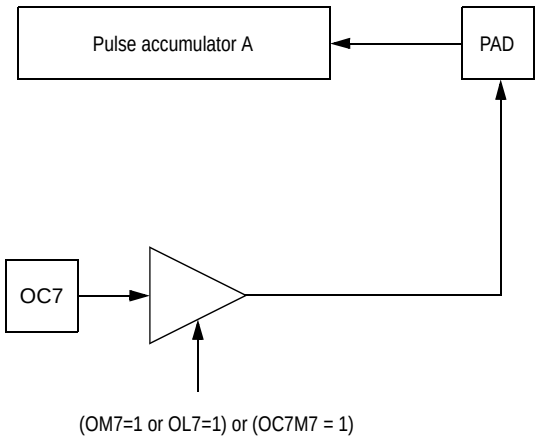


Figure 29 Block Diagram for Port7 with Output compare / Pulse Accumulator A

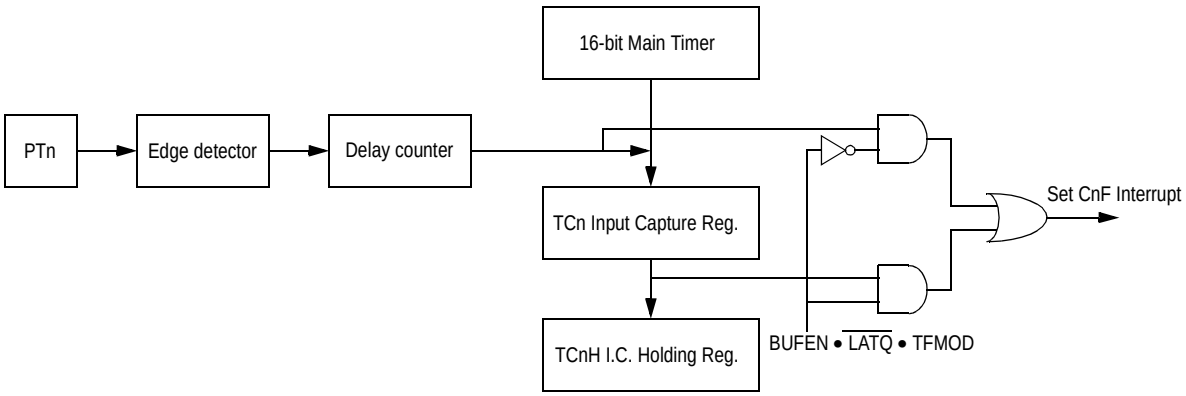


Figure 30 C3F-C0F Interrupt Flag Setting

Enhanced Capture Timer Modes of Operation

The Enhanced Capture Timer has 8 Input Capture, Output Compare (IC/OC) channels same as on the HC12 standard timer (timer channels TC0 to TC7). When channels are selected as input capture by selecting the IOSx bit in TIOS register, they are called Input Capture (IC) channels.

Four IC channels are the same as on the standard timer with one capture register which memorizes the timer value captured by an action on the associated input pin.

Four other IC channels, in addition to the capture register, have also one buffer called holding register. This permits to memorize two different timer values without generation of any interrupt.

Four 8-bit pulse accumulators are associated with the four buffered IC channels. Each pulse accumulator has a holding register to memorize their value by an action on its external input. Each pair of pulse accumulators can be used as a 16-bit pulse accumulator.

The 16-bit modulus down-counter can control the transfer of the IC registers contents and the pulse accumulators to the respective holding registers for a given period, every time the count reaches zero.

The modulus down-counter can also be used as a stand-alone time base with periodic interrupt capability.

IC Channels

The IC channels are composed of four standard IC registers and four buffered IC channels.

An **IC register** is **empty** when it has been read or latched into the holding register.

A **holding register** is **empty** when it has been read.

Non-Buffered IC Channels

The main timer value is memorized in the IC register by a valid input pin transition. If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value.

If the corresponding NOVWx bit of the ICOVW register is set, the capture register cannot be written unless it is empty.

This will prevent the captured value to be overwritten until it is read.

Buffered IC Channels

There are two modes of operations for the buffered IC channels.

- IC Latch Mode:

When enabled (LATQ=1), the main timer value is memorized in the IC register by a valid input pin transition.

The value of the buffered IC register is latched to its holding register by the Modulus counter for a given period when the count reaches zero, by a write $0000 to the modulus counter or by a write to ICLAT in the MCCTL register.

If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. In case of latching, the contents of its holding register are overwritten.

If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see IC Channels). This will prevent the captured value to be overwritten until it is read or latched in the holding register.

- IC queue mode:

When enabled (LATQ=0), the main timer value is memorized in the IC register by a valid input pin transition.

If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.

If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see IC Channels).

In queue mode, reads of holding register will latch the corresponding pulse accumulator value to its holding register.

Pulse Accumulators

There are four 8-bit pulse accumulators with four 8-bit holding registers associated with the four IC buffered channels. A pulse accumulator counts the number of active edges at the input of its channel.

The user can prevent 8-bit pulse accumulators counting further than $FF by PACMX control bit in ICSYS ($AB). In this case a value of $FF means that 255 counts or more have occurred.

Each pair of pulse accumulators can be used as a 16-bit pulse accumulator.

There are two modes of operation for the pulse accumulators.

Pulse Accumulator latch mode

The value of the pulse accumulator is transferred to its holding register when the modulus down-counter reaches zero, a write $0000 to the modulus counter or when the force latch control bit ICLAT is written.

At the same time the pulse accumulator is cleared.

Pulse Accumulator queue mode

When queue mode is enabled, reads of an input capture holding register will transfer the contents of the associated pulse accumulator to its holding register.

At the same time the pulse accumulator is cleared.

Modulus Down-Counter

The modulus down-counter can be used as a time base to generate a periodic interrupt. It can also be used to latch the values of the IC registers and the pulse accumulators to their holding registers.

The action of latching can be programmed to be periodic or only once.

Timer Register Descriptions

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 PORTT 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 PORTT 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. At successful OC7, for eachbit that is set in OC7M, the corresponding data bit in OC7D is stored to the corresponding bit of the timer port.

NOTE: *OC7M has priority over output action on timer port enabled by OMn and OLn bits in TCTL1 and TCTL2. If an OC7M bit is set, it prevents the action of corresponding OM and OL bits on the selected timer port.*

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

The 16-bit main timer is an up counter.

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				
RESET:	0	0	0	0	0	0	0	0

Read or write anytime.

TEN — Timer Enable

0 = Disables the main 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 Module Stops While in Wait

0 = Allows the timer module to continue running during wait.

1 = Disables the timer module when the MCU is in the wait mode. Timer interrupts cannot be used to get the MCU out of wait.

TSWAI also affects pulse accumulators and modulus down counters.

TSBCK — Timer and Modulus Counter Stop While in Background Mode

0 = Allows the timer and modulus counter to continue running while in background mode.

1 = Disables the timer and modulus counter whenever the MCU is in background mode. This is useful for emulation.

TBSCK does not stop the pulse accumulator.

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, $85) clears the TOF flag. Any access to the PACN3 and PACN2 registers ($A2, $A3) clears the PAOVF and PAIF flags in the PAFLG register ($A1). Any access to the PACN1 and PACN0 registers ($A4, $A5) clears the PBOVF flag in the PBFLG register ($B1). Any access to the MCCNT register ($B6, $B7) clears the MCZF flag in the MCFLG register ($A7). 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
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.

NOTE: *To enable output action by OMn and OLn bits on timer port, the corresponding bit in OC7M should be cleared.*

Table 31 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

To operate the 16-bit pulse accumulators A and B (PACA and PACB) independently of input capture or output compare 7 and 0 respectively the user must set the corresponding bits IOSn = 1, OMn = 0 and OLn = 0. OC7M7 or OC7M0 in the OC7M register must also be cleared.

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 2 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

Read or write anytime.

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 Port Pull-Up Resistor Enable

This enable bit controls pull-up resistors on the timer port pins when the pins are configured as inputs.

0 = Disable pull-up resistor function
1 = Enable pull-up resistor function

RDPT — Timer Port Drive Reduction

This bit reduces the effective output driver size which can reduce power supply current and generated noise depending upon pin loading.

0 = Normal output drive capability
1 = Enable output drive reduction function

TCRE — Timer Counter Reset Enable

This bit allows the timer counter to be reset by a successful output compare 7 event. This mode of operation is similar to an up-counting modulus counter.

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 be set when TCNT is reset from $FFFF to $0000.

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 main timer counter.

Table 32 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	64
1	1	1	128

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

TFLG1 — Main 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.

Use of the TFMOD bit in the ICSYS register ($AB) in conjunction with the use of the ICOVW register ($AA) allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture.

Read anytime. Write used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a zero will not affect 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 — Main 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 (set bits 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 — 16-Bit Pulse Accumulator A 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

16-Bit Pulse Accumulator A (PACA) is formed by cascading the 8-bit pulse accumulators PAC3 and PAC2.

When PAEN is set, the PACA is enabled. The PACA shares the input pin with IC7.

Read: any time
Write: any time

PAEN — Pulse Accumulator A System Enable

0 = 16-Bit Pulse Accumulator A system disabled. 8-bit PAC3 and PAC2 can be enabled when their related enable bits in ICPACR ($A8) are set.
Pulse Accumulator Input Edge Flag (PAIF) function is disabled.

1 = Pulse Accumulator A system enabled. The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled, the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA.
PA3EN and PA2EN control bits in ICPACR ($A8) have no effect.
Pulse Accumulator Input Edge Flag (PAIF) function is enabled.

PAEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled.

PAMOD — Pulse Accumulator Mode

0 = event counter mode

1 = gated time accumulation mode

PEDGE — Pulse Accumulator Edge Control

For PAMOD bit = 0 (event counter mode).

0 = falling edges on PT7 pin cause the count to be incremented

1 = rising edges on PT7 pin cause the count to be incremented

For PAMOD bit = 1 (gated time accumulation mode).

0 = PT7 input pin high enables M divided by 64 clock to Pulse Accumulator and the trailing falling edge on PT7 sets the PAIF flag.

1 = PT7 input pin low enables M divided by 64 clock to Pulse Accumulator and the trailing rising edge on PT7 sets the PAIF flag.

PAMOD	PEDGE	Pin Action
0	0	Falling edge
0	1	Rising edge
1	0	Div. by 64 clock enabled with pin high level
1	1	Div. by 64 clock enabled with pin low level

If the timer is not active (TEN = 0 in TSCR), there is no divide-by-64 since the E÷64 clock is generated by the timer prescaler.

CLK1, CLK0 — Clock Select Bits

CLK1	CLK0	Clock Source
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 A 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 A 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 the 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 A Overflow Flag

Set when the 16-bit pulse accumulator A overflows from $FFFF to $0000,or when 8-bit pulse accumulator 3 (PAC3) overflows from $FF to $00.

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 PT7 input pin. In event mode the event edge triggers PAIF and in gated time accumulation mode the trailing edge of the gate signal at the PT7 input pin triggers PAIF.

This bit is cleared by a write to the PAFLG register with bit 0 set. Any access to the PACN3, PACN2 registers will clear all the flags in this register when TFFCA bit in register TSCR($86) is set.

PACN3, PACN2 — Pulse Accumulators Count Registers **$00A2, $00A3**

	BIT 7	6	5	4	3	2	1	BIT 0	
$00A2	BIt 7	6	5	4	3	2	1	Bit 0	PACN3 (hi)
$00A3	Bit 7	6	5	4	3	2	1	Bit 0	PACN2 (lo)
RESET:	0	0	0	0	0	0	0	0	

Read or write any time.

The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled (PAEN=1 in PACTL, $A0) the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA.

When PACN3 overflows from $FF to $00, the Interrupt flag PAOVF in PAFLG ($A1) is set.

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.

PACN1, PACN0 — Pulse Accumulators Count Registers **$00A4, $00A5**

	BIT 7	6	5	4	3	2	1	BIT 0	
$00A4	BIt 7	6	5	4	3	2	1	Bit 0	PACN1 (hi)
$00A5	Bit 7	6	5	4	3	2	1	Bit 0	PACN0 (lo)
RESET:	0	0	0	0	0	0	0	0	

Read or write any time.

The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator. When PACB in enabled, (PBEN=1 in PBCTL, $B0) the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB.

When PACN1 overflows from $FF to $00, the Interrupt flag PBOVF in PBFLG ($B1) is set.

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.

MCCTL — 16-Bit Modulus Down-Counter Control Register **$00A6**

	BIT 7	6	5	4	3	2	1	BIT 0
	MCZI	MODMC	RDMCL	ICLAT	FLMC	MCEN	MCPR1	MCPR0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

MCZI — Modulus Counter Underflow Interrupt Enable
- 0 = Modulus counter interrupt is disabled.
- 1 = Modulus counter interrupt is enabled.

MODMC — Modulus Mode Enable
- 0 = The counter counts once from the value written to it and will stop at $0000.
- 1 = Modulus mode is enabled. When the counter reaches $0000, the counter is loaded with the latest value written to the modulus count register.

NOTE: *For proper operation, the MCEN bit should be cleared before modifying the MODMC bit in order to reset the modulus counter to $FF.*

RDMCL — Read Modulus Down-Counter Load
- 0 = Reads of the modulus count register will return the present value of the count register.
- 1 = Reads of the modulus count register will return the contents of the load register.

ICLAT — Input Capture Force Latch Action

When input capture latch mode is enabled (LATQ and BUFEN bit in ICSYS ($AB) are set), a write one to this bit immediately forces the contents of the input capture registers TC0 to TC3 and their corresponding 8-bit pulse accumulators to be latched into the associated holding registers. The pulse accumulators will be automatically cleared when the latch action occurs.

Writing zero to this bit has no effect. Read of this bit will return always zero.

FLMC — Force Load Register into the Modulus Counter Count Register

This bit is active only when the modulus down-counter is enabled (MCEN=1).

A write one into this bit loads the load register into the modulus counter count register. This also resets the modulus counter prescaler. Write zero to this bit has no effect.

When MODMC=0, counter starts counting and stops at $0000.

Read of this bit will return always zero.

MCEN — Modulus Down-Counter Enable

0 = Modulus counter disabled.
1 = Modulus counter is enabled.

When MCEN=0, the counter is preset to $FFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled.

MCPR1, MCPR0 — Modulus Counter Prescaler select

These two bits specify the division rate of the modulus counter prescaler.

The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs.

MCPR1	MCPR0	Prescaler division rate
0	0	1
0	1	4
1	0	8
1	1	16

MCFLG — 16-Bit Modulus Down-Counter FLAG Register **$00A7**

	BIT 7	6	5	4	3	2	1	BIT 0
	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
RESET:	0	0	0	0	0	0	0	0

Read: any time
Write: Only for clearing bit 7

MCZF — Modulus Counter Underflow Interrupt Flag

The flag is set when the modulus down-counter reaches $0000.

A write one to this bit clears the flag. Write zero has no effect.

Any access to the MCCNT register will clear the MCZF flag in this register when TFFCA bit in register TSCR($86) is set.

POLF3 – POLF0 — First Input Capture Polarity Status

This are read only bits. Write to these bits has no effect.

Each status bit gives the polarity of the first edge which has caused an input capture to occur after capture latch has been read.

Each POLFx corresponds to a timer PORTx input.

0 = The first input capture has been caused by a falling edge.
1 = The first input capture has been caused by a rising edge.

ICPAR — Input Control Pulse Accumulators Register **$00A8**

	BIT 7	6	5	4	3	2	1	BIT 0
	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
RESET:	0	0	0	0	0	0	0	0

The 8-bit pulse accumulators PAC3 and PAC2 can be enabled only if PAEN in PATCL ($A0) is cleared. If PAEN is set, PA3EN and PA2EN have no effect.

The 8-bit pulse accumulators PAC1 and PAC0 can be enabled only if PBEN in PBTCL ($B0) is cleared. If PBEN is set, PA1EN and PA0EN have no effect.

Read or write any time.

PAxEN — 8-Bit Pulse Accumulator ‘x’ Enable
0 = 8-Bit Pulse Accumulator is disabled.
1 = 8-Bit Pulse Accumulator is enabled.

DLYCT — Delay Counter Control Register **$00A9**

	BIT 7	6	5	4	3	2	1	BIT 0
	0	0	0	0	0	0	DLY1	DLY0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

If enabled, after detection of a valid edge on input capture pin, the delay counter counts the pre-selected number of M clock (module clock) cycles, then it will generate a pulse on its output. The pulse is generated only if the level of input signal, after the preset delay, is the opposite of the level before the transition.This will avoid reaction to narrow input pulses.

After counting, the counter will be cleared automatically.

Delay between two active edges of the input signal period should be longer than the selected counter delay.

DLYx — Delay Counter Select

DLY1	DLY0	Delay
0	0	Disabled (bypassed)
0	1	256M clock cycles
1	0	512M clock cycles
1	1	1024 M clock cycles

ICOVW — Input Control Overwrite Register **$00AA**

	BIT 7	6	5	4	3	2	1	BIT 0
	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

An IC register is empty when it has been read or latched into the holding register.

A holding register is empty when it has been read.

NOVWx — No Input Capture Overwrite

0 = The contents of the related capture register or holding register can be overwritten when a new input capture or latch occurs.

1 = The related capture register or holding register cannot be written by an event unless they are empty (see IC Channels). This will prevent the captured value to be overwritten until it is read or latched in the holding register.

ICSYS — Input Control System Control Register **$00AB**

	BIT 7	6	5	4	3	2	1	BIT 0
	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
RESET:	0	0	0	0	0	0	0	0

Read: any time
Write: May be written once (SMODN=1). Writes are always permitted when SMODN=0.

SHxy — Share Input action of Input Capture Channels x and y

0 = Normal operation

1 = The channel input ‘x’ causes the same action on the channel ‘y’. The port pin ‘x’ and the corresponding edge detector is used to be active on the channel ‘y’.

TFMOD — Timer Flag-setting Mode

Use of the TFMOD bit in the ICSYS register ($AB) in conjunction with the use of the ICOVW register ($AA) allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture.

By setting TFMOD in queue mode, when NOVW bit is set and the corresponding capture and holding registers are emptied, an input capture event will first update the related input capture register with

the main timer contents. At the next event the TCn data is transferred to the TCnH register, The TCn is updated and the CnF interrupt flag is set. See Figure 30.

In all other input capture cases the interrupt flag is set by a valid external event on PTn.

0 = The timer flags C3F–C0F in TFLG1 ($8E) are set when a valid input capture transition on the corresponding port pin occurs.

1 = If in queue mode (BUFEN=1 and LATQ=0), the timer flags C3F–C0F in TFLG1 ($8E) are set only when a latch on the corresponding holding register occurs.
If the queue mode is not engaged, the timer flags C3F–C0F are set the same way as for TFMOD=0.

PACMX — 8-Bit Pulse Accumulators Maximum Count

0 = Normal operation. When the 8-bit pulse accumulator has reached the value $FF, with the next active edge, it will be incremented to $00.

1 = When the 8-bit pulse accumulator has reached the value $FF, it will not be incremented further. The value $FF indicates a count of 255 or more.

BUFEN — IC Buffer Enable

0 = Input Capture and pulse accumulator holding registers are disabled.

1 = Input Capture and pulse accumulator holding registers are enabled. The latching mode is defined by LATQ control bit. Write one into ICLAT bit in MCCTL ($A6), when LATQ is set will produce latching of input capture and pulse accumulators registers into their holding registers.

LATQ — Input Control Latch or Queue Mode Enable

The BUFEN control bit should be set in order to enable the IC and pulse accumulators holding registers. Otherwise LATQ latching modes are disabled.

Write one into ICLAT bit in MCCTL ($A6), when LATQ and BUFEN are set will produce latching of input capture and pulse accumulators registers into their holding registers.

0 = Queue Mode of Input Capture is enabled.
The main timer value is memorized in the IC register by a valid input pin transition.
With a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.

1 = Latch Mode is enabled. Latching function occurs when modulus down-counter reaches zero or a zero is written into the count register MCCNT (see Buffered IC Channels).
With a latching event the contents of IC registers and 8-bit pulse accumulators are transferred to their holding registers.
8-bit pulse accumulators are cleared.

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

	BIT 7	6	5	4	3	2	1	BIT 0
	0	0	0	0	0	0	TCBYP	0
RESET:	0	0	0	0	0	0	0	0

Read: any time
Write: only in special mode (SMOD = 1).

TCBYP — Main Timer Divider Chain Bypass

0 = Normal operation

1 = For testing only. 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.
When the high byte of timer counter TCNT ($84) overflows from $FF to $00, the TOF flag in TFLG2 ($8F) will be set.

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

	BIT 7	6	5	4	3	2	1	BIT 0
PORT	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
RESET:	-	-	-	-	-	-	-	-

Read: any time (inputs return pin level; outputs return data register contents)

Write: data stored in an internal latch (drives pins only if configured for output)

Since the Output Compare 7 shares the pin with Pulse Accumulator input, the only way for Pulse accumulator to receive an independent input from Output Compare 7 is setting both OM7 & OL7 to be zero, and also OC7M7 in OC7M register to be zero.

OC7 is still able to reset the counter if enabled while PT7 is used as input to Pulse Accumulator.

PORTT can be read anytime. When configured as an input, a read will return the pin level. When configured as an 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 the width of 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 any time.

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 line associated with an enabled output compare. In these cases the data direction bits will not be changed, but have no effect 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.

PBCTL — 16-Bit Pulse Accumulator B Control Register **$00B0**

	BIT 7	6	5	4	3	2	1	BIT 0
	0	PBEN	0	0	0	0	PBOVI	0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

16-Bit Pulse Accumulator B (PACB) is formed by cascading the 8-bit pulse accumulators PAC1 and PAC0.

When PBEN is set, the PACB is enabled. The PACB shares the input pin with IC0.

PBEN — Pulse Accumulator B System Enable

0 = 16-bit Pulse Accumulator system disabled. 8-bit PAC1 and PAC0 can be enabled when their related enable bits in ICPACR ($A8) are set.

1 = Pulse Accumulator B system enabled. The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator. When PACB in enabled, the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB.
PA1EN and PA0EN control bits in ICPACR ($A8) have no effect.

PBEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled.

PBOVI — Pulse Accumulator B Overflow Interrupt enable

0 = interrupt inhibited

1 = interrupt requested if PBOVF is set

PBFLG — Pulse Accumulator B Flag Register **$00B1**

	BIT 7	6	5	4	3	2	1	BIT 0
	0	0	0	0	0	0	PBOVF	0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

PBOVF — Pulse Accumulator B Overflow Flag

This bit is set when the 16-bit pulse accumulator B overflows from $FFFF to $0000, or when 8-bit pulse accumulator 1 (PAC1) overflows from $FF to $00.

This bit is cleared by a write to the PBFLG register with bit 1 set.

Any access to the PACN1 and PACN0 registers will clear the PBOVF flag in this register when TFFCA bit in register TSCR($86) is set.

PA3H–PA0H — 8-Bit Pulse Accumulators Holding Registers **$00B2–$00B5**

	BIT 7	6	5	4	3	2	1	BIT 0	
$00B2	BIt 7	6	5	4	3	2	1	Bit 0	PA3H
$00B3	Bit 7	6	5	4	3	2	1	Bit 0	PA2H
$00B4	BIt 7	6	5	4	3	2	1	Bit 0	PA1H
$00B5	Bit 7	6	5	4	3	2	1	Bit 0	PA0H
RESET:	0	0	0	0	0	0	0	0	

Read: any time
Write: has no effect.

These registers are used to latch the value of the corresponding pulse accumulator when the related bits in register ICPAR ($A8) are enabled (see Pulse Accumulators).

MCCNT — Modulus Down-Counter Count Register **$00B6, $00B7**

	BIT 7	6	5	4	3	2	1	BIT 0	
$00B6	BIt 15	14	13	12	11	10	9	Bit 8	MCCNTH
$00B7	Bit 7	6	5	4	3	2	1	Bit 0	MCCNTL
RESET:	1	1	1	1	1	1	1	1	

Read or write any time.

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 different result than accessing them as a word.

If the RDMCL bit in MCCTL register is cleared, reads of the MCCNT register will return the present value of the count register. If the RDMCL

bit is set, reads of the MCCNT will return the contents of the load register.

If a $0000 is written into MCCNT and modulus counter while LATQ and BUFEN in ICSYS ($AB) register are set, the input capture and pulse accumulator registers will be latched.

With a $0000 write to the MCCNT, the modulus counter will stay at zero and does not set the MCZF flag in MCFLG register.

If modulus mode is enabled (MODMC=1), a write to this address will update the load register with the value written to it. The count register will not be updated with the new value until the next counter underflow.

The FLMC bit in MCCTL ($A6) can be used to immediately update the count register with the new value if an immediate load is desired.

If modulus mode is not enabled (MODMC=0), a write to this address will clear the prescaler and will immediately update the counter register with the value written to it and down-counts once to $0000.

TC0H — Timer Input Capture Holding Register 0 **$00B8–$00B9**

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

TC1H — Timer Input Capture Holding Register 1 **$00BA–$00BB**

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

TC2H — Timer Input Capture Holding Register 2 **$00BC–$00BD**

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

TC3H — Timer Input Capture Holding Register 3 **$00BE–$00BF**

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

Read: any time
Write: has no effect.

These registers are used to latch the value of the input capture registers TC0 – TC3. The corresponding IOSx bits in TIOS ($80) should be cleared (see IC Channels).

Timer and Modulus Counter Operation in Different Modes

STOP: Timer and modulus counter are off since clocks are stopped.

BGDM: Timer and modulus counter keep on running, unless TSBCK (REG$86, bit5) is set to one.

WAIT: Counters keep on running, unless TSWAI in TSCR ($86) is set to one.

NORMAL: Timer and modulus counter keep on running, unless TEN in TSCR($86) respectively MCEN in MCCTL ($A6) are cleared.

TEN=0: All 16-bit timer operations are stopped, can only access the registers.

MCEN=0: Modulus counter is stopped.

PAEN=1: 16-bit Pulse Accumulator A is active.

PAEN=0: 8-Bit Pulse Accumulators 3 and 2 can be enabled. (see ICPAR)

PBEN=1: 16-bit Pulse Accumulator B is active.

PBEN=0: 8-Bit Pulse Accumulators 1 and 0 can be enabled. (see ICPAR)

Multiple Serial Interface

Contents

Introduction

The multiple serial interface (MSI) module consists of three independent serial I/O sub-systems: two serial communication interfaces (SCI0 and SCI1) and the serial peripheral interface (SPI). Each serial pin shares function with the general-purpose port pins of port S. The SCI subsystems are NRZ type systems that are compatible with standard RS-232 systems. These SCI systems have a new 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.

Block diagram

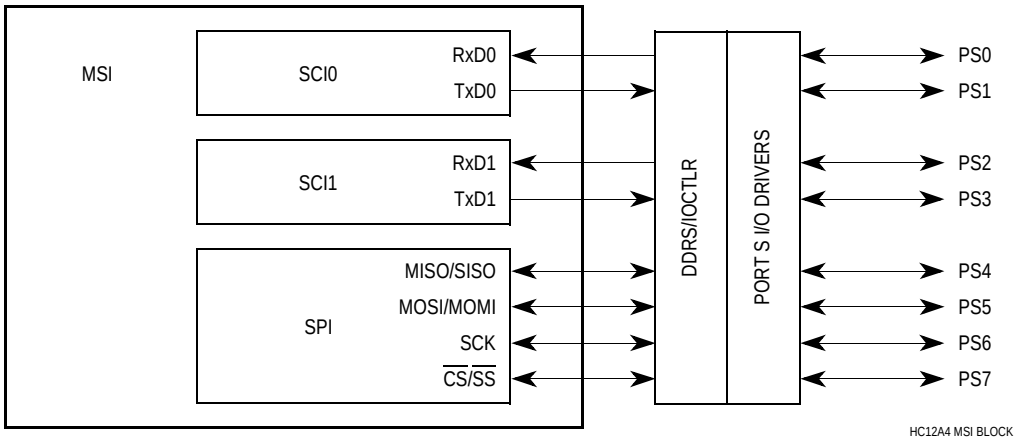


Figure 31 Multiple Serial Interface Block Diagram

Serial Communication Interface (SCI)

Two serial communication interfaces are available on the MC68HC912DT128A. These are NRZ format (one start, eight or nine data, and one stop bit) asynchronous communication systems with independent internal baud rate generation circuitry and SCI transmitters and receivers. They 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 for each SCI provide the external interface for the transmitted data (TXD) and the received data (RXD).

For a faster wake-up out of WAIT mode by a received SCI message, both SCI have the capability of sending a receiver interrupt, if enabled, when RAF (receiver active flag) is set. For compatibility with other

M68HC12 products, this feature is active only in WAIT mode and is disabled when VDDPLL supply is at V_{SS} level.

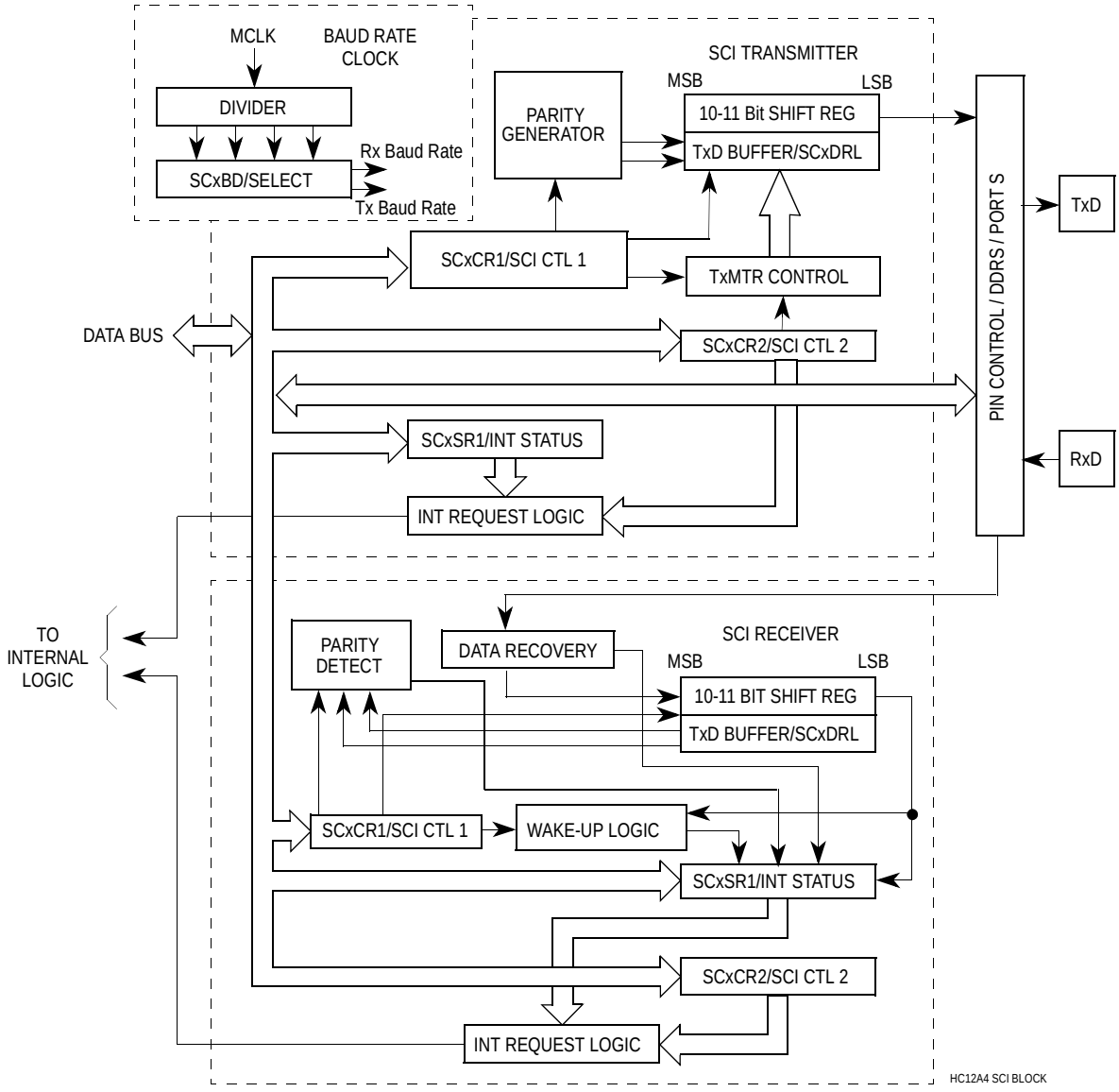


Figure 32 Serial Communications Interface Block Diagram

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.

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 M Clock.

Table 33 Baud Rate Generation

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

SCI 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. Both SCI have identical control registers mapped in two blocks of eight bytes.

SC0BDH/SC1BDH — SCI Baud Rate Control Register **$00C0/$00C8**

	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/SC1BDL — SCI Baud Rate Control Register **$00C1/$00C9**

	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	

SCxBDH and SCxBDL 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 TE or RE bit in SCxCR2 register is set for the first time after reset, and/or the baud rate generator is disabled when SBR[12:0] = 0.*

BTST — Reserved for test function

BSPL — Reserved for test function

BRLD — Reserved for test function

SC0CR1/SC1CR1 — SCI Control Register 1 **$00C2/$00CA**

	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 34.

WOMS — Wired-Or Mode for Serial Pins

This bit controls the two pins (TXD and RXD) associated with the SCIx 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 DDRS0/2) which is the single wire case when using the SCI. WOMS bit still affects general-purpose output on TXD and RXD pins when SCIx 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 34 Loop Mode Functions

LOOPS	RSRC	DDRS1(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	1	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 — Wake-up 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/SC1CR2 — SCI Control Register 2 **$00C3/$00CB**

	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, RAF interrupt in WAIT mode disabled

1 = SCI interrupt will be requested whenever the RDRF or OR status flag is set, or when RAF is set while in WAIT mode with VDDPLL high.

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/bit 3) 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 only 10 (or 11) zeros and then revert to mark idle or sending data.

SC0SR1/SC1SR1 — SCI Status Register 1 **$00C4/$00CC**

	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 SCxSR1 (RDRF, IDLE, OR, NF, FE, and PF) are all cleared by a read of the SCxSR1 register followed by a read of the transmit/receive data register low byte. However, only those bits which were set when SCxSR1 was read will be cleared by the subsequent read of the transmit/receive data register low byte. The transmit related bits in SCxSR1 (TDRE and TC) are cleared by a read of the SCxSR1 register followed by a write to the transmit/receive data register low byte.

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 SCxSR1 is read before writing to the transmit data register. Reset sets this bit.

0 = SCxDR 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 SCxSR1 with TC set and then writing to SCxDR.

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 SCxDR. Clear the RDRF flag by reading SCxSR1 with RDRF set and then reading SCxDR.

0 = SCxDR empty
1 = SCxDR 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 SCxSR1 with FE set and then reading SCxDR.

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 SCxCR1.

0 = Parity correct
1 = Incorrect parity detected

SC0SR2/SC1SR2 — SCI Status Register 2 **$00C5/$00CD**

	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

If enabled with RIE = 1, RAF set generates an interrupt when VDDPLL is high while in WAIT mode.

SC0DRH/SC1DRH — SCI Data Register High **$00C6/$00CE**

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

SC0DRL/SC1DRL — SCI Data Register Low **$00C7/$00CF**

	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:	—	—	—	—	—	—	—	—

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). SCxDRL:SCxDRH form the 9-bit data word for the SCI. If the SCI is being used with a 7- or 8-bit data word, only SCxDRL 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.

Serial Peripheral Interface (SPI)

The serial peripheral interface allows the MC68HC912DT128A to communicate synchronously with peripheral devices and other microprocessors. The SPI system in the MC68HC912DT128A 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.

SPI Baud Rate Generation

The E Clock is input to a divider series and the resulting SPI clock rate may be selected to be E 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.

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.

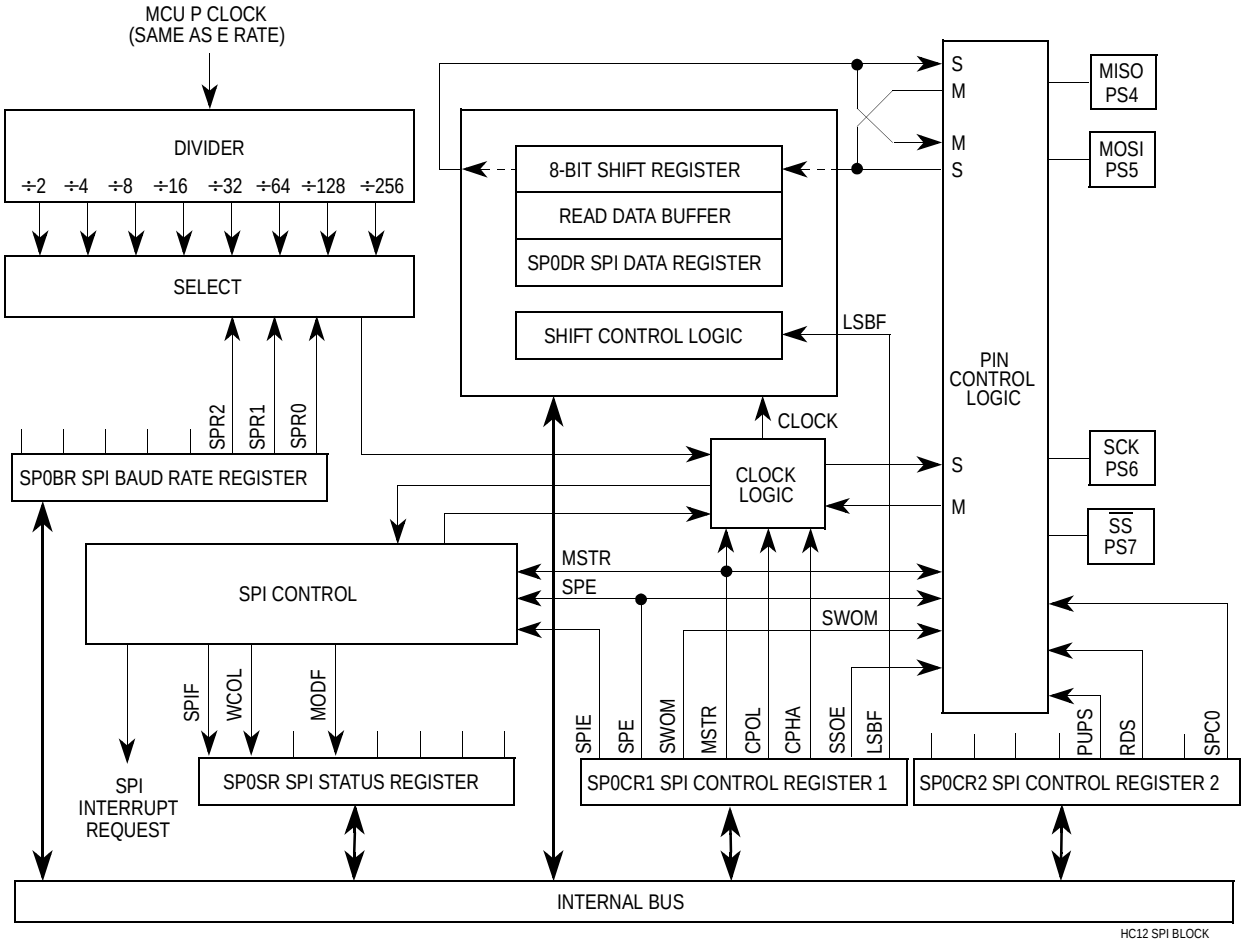


Figure 33 Serial Peripheral Interface Block Diagram

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.

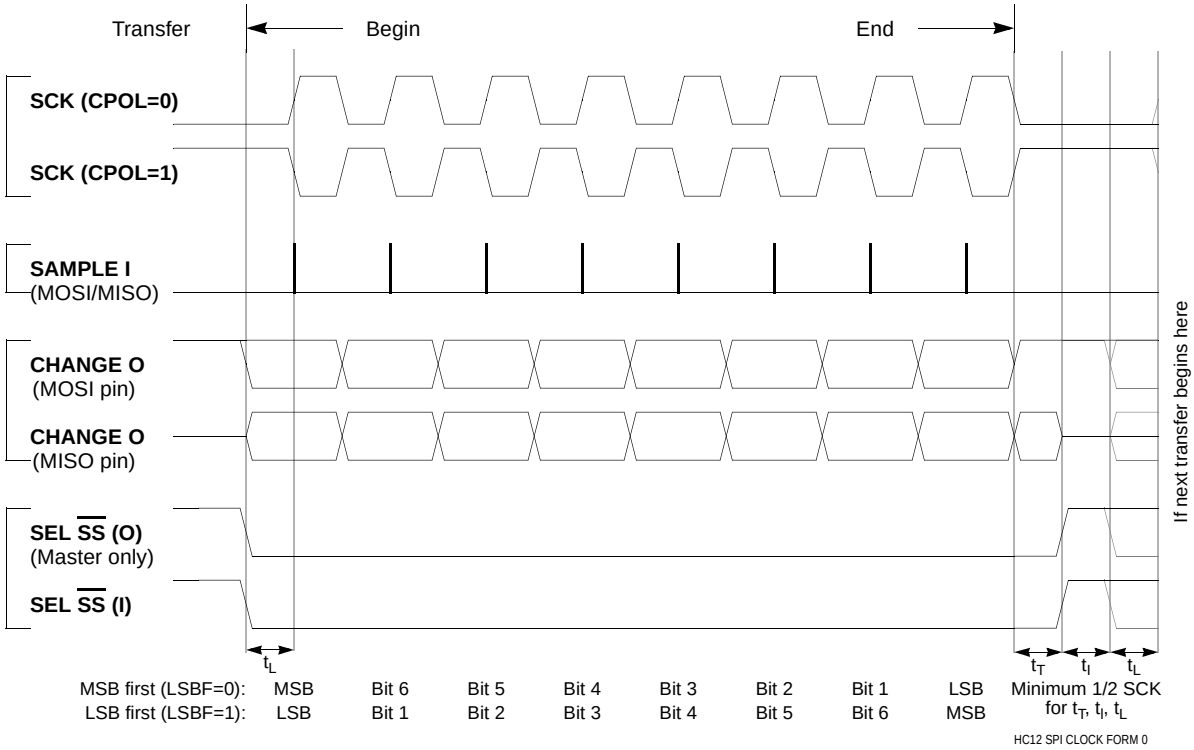


Figure 34 SPI Clock Format 0 (CPHA = 0)

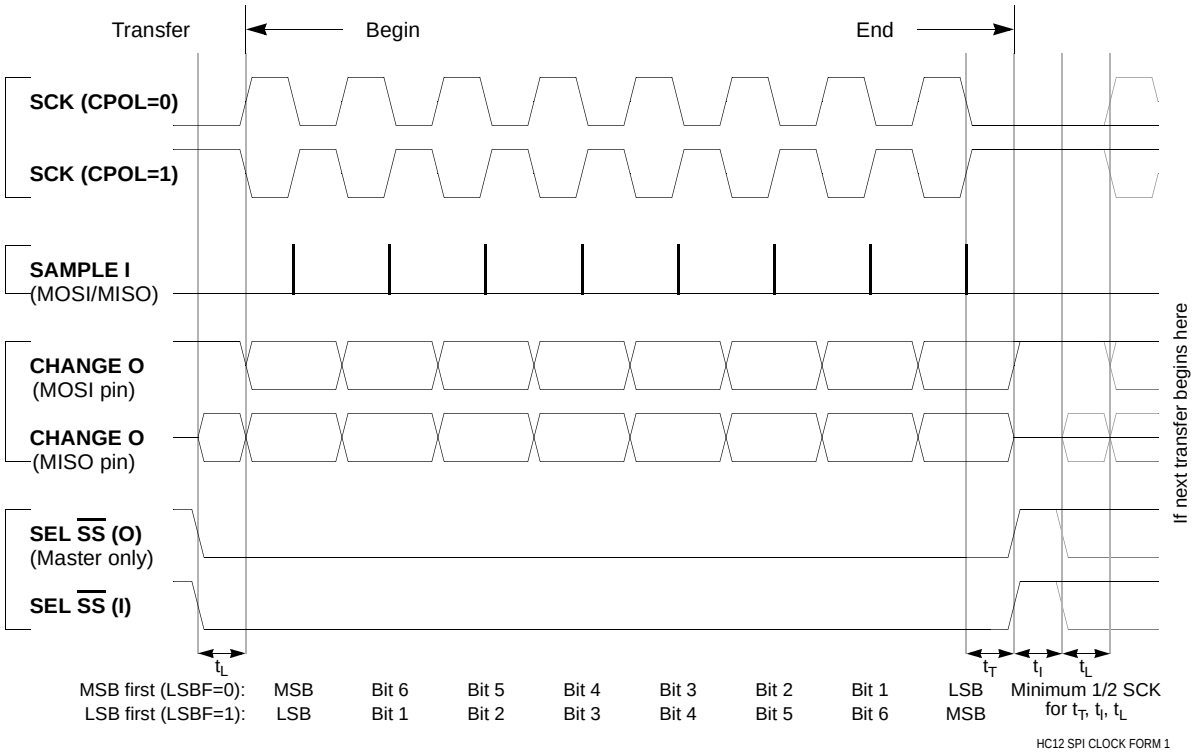


Figure 35 SPI Clock Format 1 (CPHA = 1)

$\overline{SS}$ Output

Available in master mode only, $\overline{SS}$ output is enabled with the SSOE bit in the SP0CR1 register if the corresponding DDRS 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.

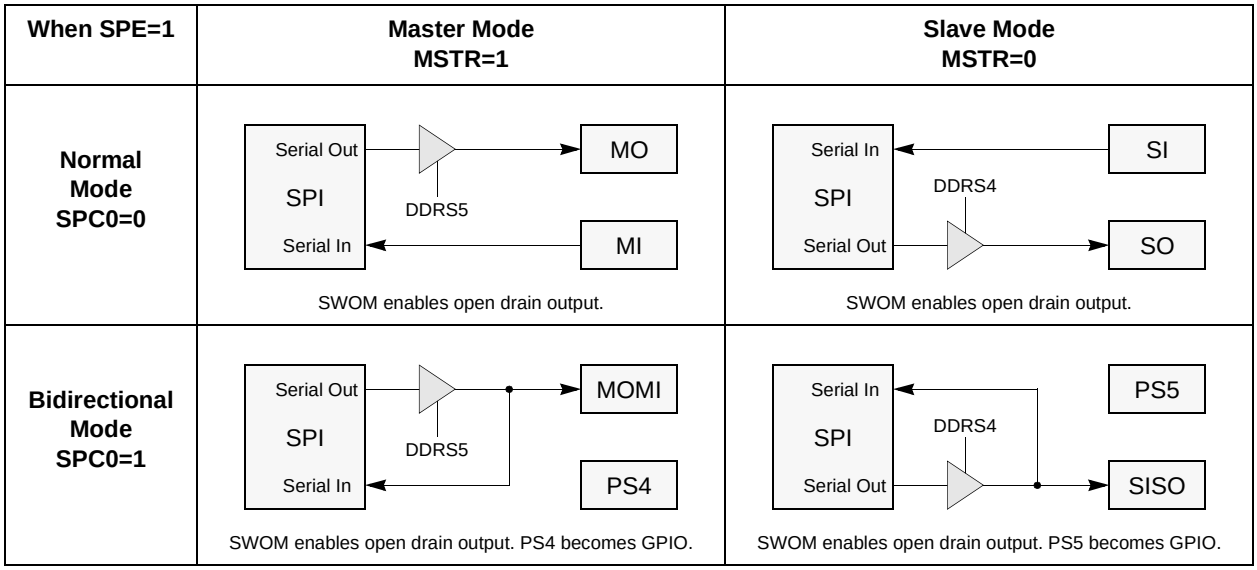
Table 35 $\overline{SS}$ Output Selection

DDRS7	SSOE	Master Mode	Slave Mode
0	0	SS Input with MODF Feature	SS Input
0	1	Reserved	SS Input
1	0	General-Purpose Output	SS Input
1	1	SS Output	SS Input

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 36 Normal Mode and Bidirectional Mode



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. For more information refer to **Operating Modes**.

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

0 = SPI interrupts are inhibited

1 = Hardware interrupt sequence is requested each time the SPIF or MODF status flag is set

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

When MODF is set, MSTR always reads zero. SP0CR1 must be written as part of a mode fault recovery sequence.

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 34 and Figure 35.

SSOE — Slave Select Output Enable

The $\overline{SS}$ output feature is enabled only in the master mode by asserting the SSOE and DDRS7.

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	PUPS	RDPS	SSWAI	SPC0
RESET:	0	0	0	0	1	0	0	0

Read or write anytime.

PUPS — Pull-Up Port S Enable

0 = No internal pull-ups on port S

1 = All port S input pins have an active pull-up device. If a pin is programmed as output, the pull-up device becomes inactive

RDPS — Reduce Drive of Port S

0 = Port S output drivers operate normally

1 = All port S output pins have reduced drive capability for lower power and less noise

SSWAI — Serial Interface Stop in WAIT mode

0 = Serial interface clock operates normally

1 = Halt serial interface clock generation in WAIT mode

SPC0 — Serial Pin Control 0

This bit decides serial pin configurations with MSTR control bit.

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

1. The serial pin control 0 bit enables bidirectional configurations.
2. Slave output is enabled if DDRS4 = 1, SS = 0 and MSTR = 0. (#1, #3)
3. Master output is enabled if DDRS5 = 1 and MSTR = 1. (#2, #4)
4. SCK output is enabled if DDRS6 = 1 and MSTR = 1. (#2, #4)
5. SS output is enabled if DDRS7 = 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 36 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

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.

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
MSI	$\overline{SS}$ $\overline{CS}$	SCK	MOSI MOMI	MISO SISO	TXD1	RXD1	TXD0	RXD0
RESET:	-	-	-	-	-	-	-	-

Read anytime (inputs return pin level; outputs return pin driver input level). Write data stored in internal latch (drives pins only if configured for output). 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 (SPI and SCI0/1).

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

DDS2, DDS0 — Data Direction for Port S Bit 2 and 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.

DDS3, DDS1 — Data Direction for Port S Bit 3 and 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[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, DDRS7 has no meaning or effect; the PS7 pin is dedicated as the $\overline{SS}$ input. In SPI master mode, DDRS7 determines whether PS7 is an error detect input to the SPI or a general-purpose or slave select output line.

NOTE: *If mode fault error occurs bits 5, 6 and 7 are forced to zero.*

Inter-IC Bus

Contents

Introduction

The Inter-IC Bus (IIC or I^2C) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. Being a two-wire device, the IIC minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400pF.

IIC Features

The IIC module has the following key features:

- Compatible with I^2C Bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Eight-bit general purpose I/O port

A block diagram of the IIC module is shown in Figure 37.

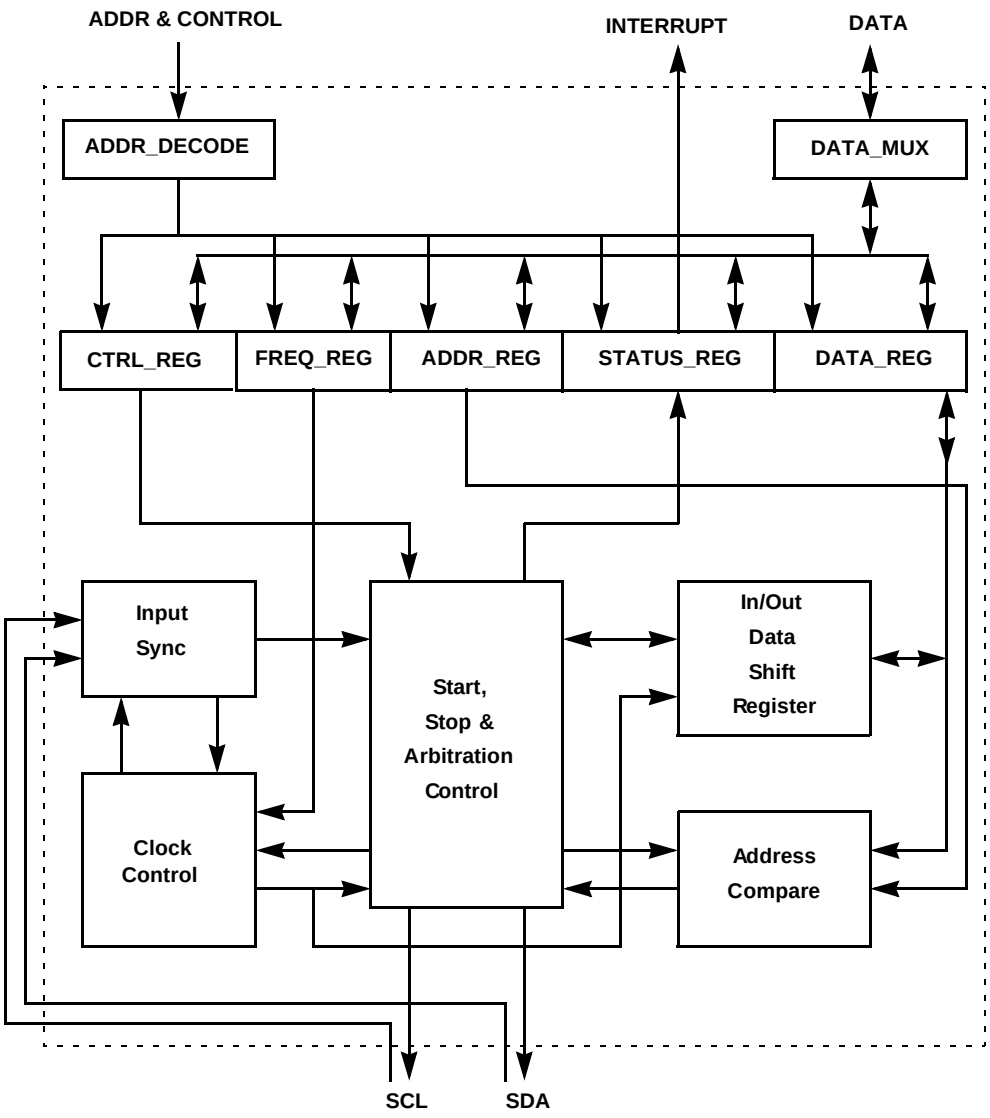


Figure 37 IIC Block Diagram

IIC System Configuration

The IIC system uses a Serial Data line (SDA) and a Serial Clock Line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. Logic “and” function is exercised on both lines with external pull-up resistors, the value of these resistors is system dependent.

IIC Protocol

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 38.

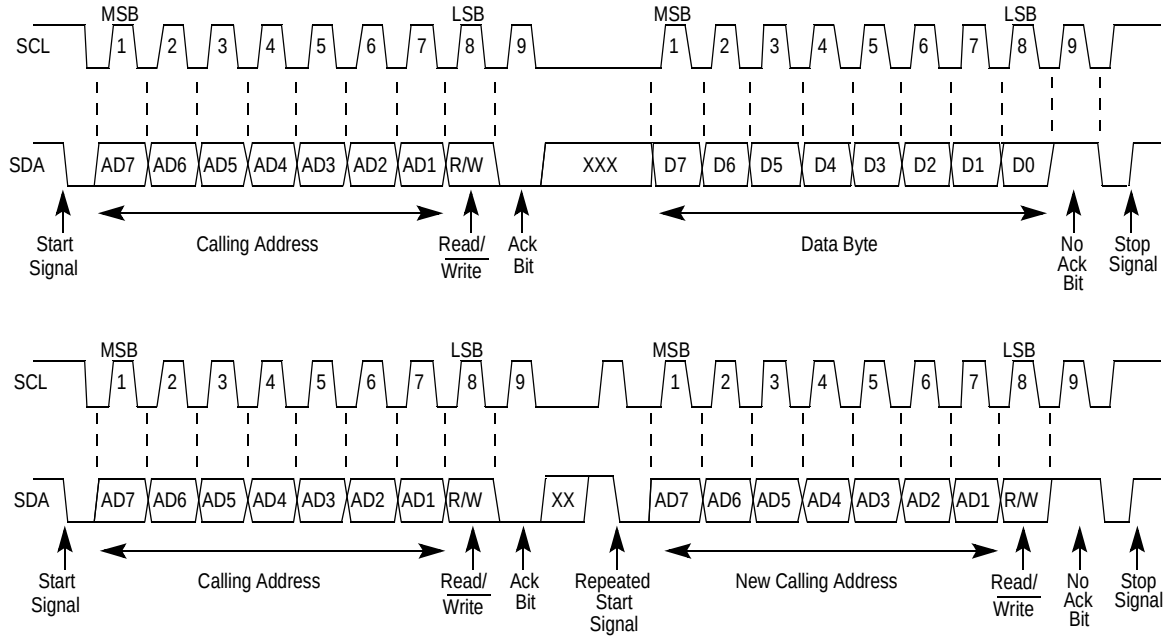


Figure 38 IIC Transmission Signals

START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 38, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and wakes up all slaves.

Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 38).

Slave address - No two slaves in the system may have the same address. If the IIC is master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time. If however arbitration is lost during an address cycle the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ bit sent by the calling master.

NOTE: *All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.*

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 38. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge

bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.

STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical "1" (see Figure 38).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

Repeated START Signal

As shown in Figure 38, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

Arbitration Procedure

IIC is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition

from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

Clock Synchronization

Since wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 39). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

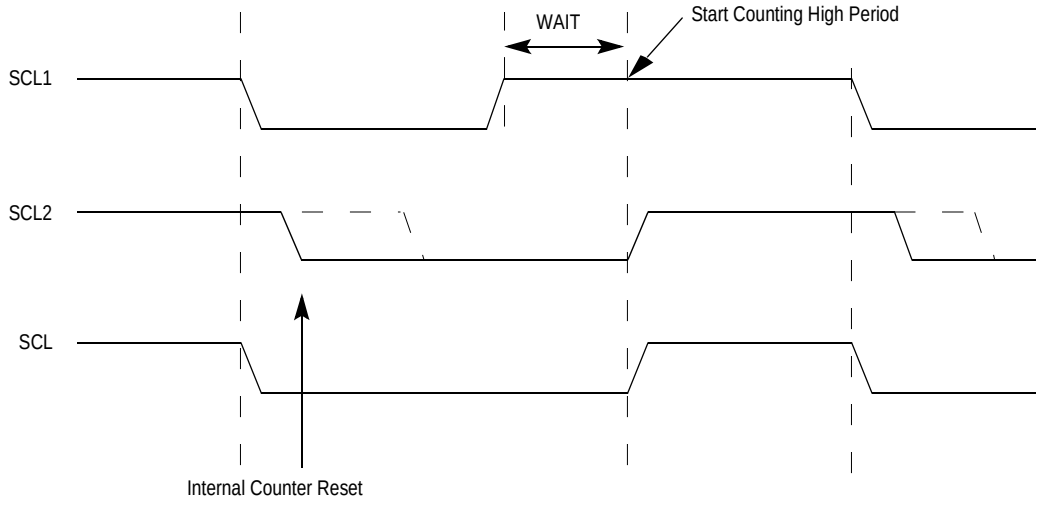


Figure 39 IIC Clock Synchronization

Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of

one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

Clock Stretching The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

IIC Register Descriptions

.

IBAD — IIC Bus Address Register **$00E0**

	Bit 7	6	5	4	3	2	1	Bit 0
	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime

This register contains the address the IIC will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer

ADR7–ADR1 — Slave Address

Bit 1 to bit 7 contain the specific slave address to be used by the IIC module.

The default mode of IIC is slave mode for an address match on the bus.

IBFD — IIC Bus Frequency Divider Register **$00E1**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime

IBC5–IBC0 — IIC Bus Clock Rate 5–0

This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescaled shift register - IBC5-3 select the prescaler divider and IBC2-0 select the shift register tap point. The IBC bits are decoded to give the Tap and Prescale values as shown in Table 37.

NOTE: *At 8 MHz system bus frequency, the IIC bus frequency will slow down by as much as 5%. However, the communications rate of the IIC system will be automatically adjusted to a slower rate.*

Table 37 IIC Tap and Prescale Values

IBC2-0 (bin)	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

IBC5-3 (bin)	scl2tap (clocks)	tap2tap (clocks)
000	4	1
001	4	2
010	6	4
011	6	8
100	14	16
101	30	32
110	62	64
111	126	128

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 37, all subsequent tap points are separated by 2^{IBC5-3} as shown in the tap2tap column in Table 37. The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

The serial bit clock frequency is equal to the CPU clock frequency divided by the divider shown in Table 3. The equation used to generate the divider values from the IBFD bits is:

SCL Divider = 2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Figure 3. The equation used to generate the SDA Hold value from the IBFD bits is:

SDA Hold = scl2tap + [(SDA_Tap - 1) x tap2tap] + 3

Table 3 IIC Divider and SDA Hold values

IBC5-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)	IBC5-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)
00	20	7	20	160	17
01	22	7	21	192	17
02	24	8	22	224	33
03	26	8	23	256	33
04	28	9	24	288	49
05	30	9	25	320	49
06	34	10	26	384	65
07	40	10	27	480	65
08	28	7	28	320	33
09	32	7	29	384	33
0A	36	9	2A	448	65
0B	40	9	2B	512	65
0C	44	11	2C	576	97
0D	48	11	2D	640	97
0E	56	13	2E	768	129
0F	68	13	2F	960	129
10	48	9	30	640	65
11	56	9	31	768	65
12	64	13	32	896	129
13	72	13	33	1024	129
14	80	17	34	1152	193
15	88	17	35	1280	193
16	104	21	36	1536	257

Table 3 IIC Divider and SDA Hold values

IBC5-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)
17	128	21
18	80	9
19	96	9
1A	112	17
1B	128	17
1C	144	25
1D	160	25
1E	192	33
1F	240	33

IBC5-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)
37	1920	257
38	1280	129
39	1536	129
3A	1792	257
3B	2048	257
3C	2304	385
3D	2560	385
3E	3072	513
3F	3840	513

IBCR — IIC Bus Control Register **$00E2**

	Bit 7	6	5	4	3	2	1	Bit 0
	IBEN	IBIE	MS/$\overline{SL}$	Tx/$\overline{Rx}$	TXAK	RSTA	0	IBSWAI
RESET:	0	0	0	0	0	0	0	0

Read and write anytime

IBEN — IIC Bus Enable

This bit controls the software reset of the entire IIC module.

0 = The module is reset and disabled. This is the power-on reset situation. When low the IIC system is held in reset but registers can still be accessed.

1 = The IIC system is enabled. This bit must be set before any other IBCR bits have any effect.

If the IIC module is enabled in the middle of a byte transfer the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the IIC module losing arbitration, after which bus operation would return to normal.

NOTE: *To prevent glitches from appearing on the SDA & SCL lines during reset of the IIC module, set PORTIB bit 6 & 7 to 1 before clearing the IBEN bit.*

IBIE — IIC Bus Interrupt Enable

0 = Interrupts from the IIC module are disabled. Note that this does not clear any currently pending interrupt condition.

1 = Interrupts from the IIC module are enabled. An IIC interrupt occurs provided the IBIF bit in the status register is also set.

MS/$\overline{SL}$ — Master/Slave mode select bit

Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. MS/$\overline{SL}$ is cleared without generating a STOP signal when the master loses arbitration.

0 = Slave Mode

1 = Master Mode

Tx/$\overline{Rx}$ — Transmit/Receive mode select bit

This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.

0 = Receive

1 = Transmit

TXAK — Transmit Acknowledge enable

This bit specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. Note that values written to this bit are only used when the IIC is a receiver, not a transmitter.

0 = An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data

1 = No acknowledge signal response is sent (i.e., acknowledge bit = 1)

RSTA — Repeat Start

Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.

1 = Generate repeat start cycle

IBSWAI — IIC Stop in WAIT mode

0 = IIC module operates normally

1 = Halt clock generation of IIC module in WAIT mode

IBSR — IIC Bus Status Register **$00E3**

	Bit 7	6	5	4	3	2	1	Bit 0
	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
RESET:	1	0	0	0	0	0	0	0

This status register is read-only with exception of bit 1 (IBIF) and bit 4 (IBAL), which are software clearable

TCF — Data transferring bit

While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer.

0 = Transfer in progress

1 = Transfer complete

IAAS — Addressed as a slave bit

When its own specific address (IIC Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set its Tx/$\overline{\text{Rx}}$ mode accordingly. Writing to the IIC Bus Control Register clears this bit.

0 = Not addressed

1 = Addressed as a slave

IBB — IIC Bus busy bit

This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, it is cleared.

0 = Bus is idle

1 = Bus is busy

NOTE: *If, after trying to generate a START signal and neither the IBB nor IBAL bits are set after several cycles, the IIC should be disabled and re-enabled with IBEN bit.*

IBAL — Arbitration Lost

The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances:

1. SDA sampled as low when the master drives a high during an address or data transmit cycle.
2. SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
3. A start cycle is attempted when the bus is busy.
4. A repeated start cycle is requested in slave mode.
5. A stop condition is detected when the master did not request it.

This bit must be cleared by software, by writing a one to it.

NOTE: *If, after trying to generate a START signal and neither the IBB nor IBAL bits are set after several cycles, the IIC should be disabled and re-enabled with IBEN bit.*

SRW — Slave Read/Write

When IAAS is set this bit indicates the value of the R/W command bit of the calling address sent from the master.

CAUTION: *This bit is only valid when the IIC is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated.*

Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.

0 = Slave receive, master writing to slave
1 = Slave transmit, master reading from slave

IBIF — IIC Bus Interrupt Flag

The IBIF bit is set when an interrupt is pending, which will cause a processor interrupt request provided IBIE is set. IBIF is set when one of the following events occurs:

1. Complete one byte transfer (set at the falling edge of the 9th clock).
2. Receive a calling address that matches its own specific address in slave receive mode.
3. Arbitration lost.

This bit must be cleared by software, writing a one to it, in the interrupt routine.

RXAK — Received Acknowledge

The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock.

0 = Acknowledge received
1 = No acknowledge received

.

IBDR — IIC Bus Data I/O Register **$00E4**

	Bit 7	6	5	4	3	2	1	Bit 0	
	D7	D6	D5	D4	D3	D2	D1	D0	High
RESET:	0	0	0	0	0	0	0	0	

Read and write anytime

In master transmit mode, when data is written to the IBDR a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

NOTE: *In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{SL}$ is used for the address transfer and should comprise of the calling address (in position D7-D1) concatenated with the required R/$\overline{W}$ bit (in position D0).*

IBPURD — Pull-Up and Reduced Drive for Port IB **$00E5**

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

Read and write anytime

RDPIB - Reduced Drive of Port IB

0 = All port IB output pins have full drive enabled.

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

PUPIB - Pull-Up Port IB Enable

0 = Port IB pull-ups are disabled.

1 = Enable pull-up devices for port IB input pins [7:6]. Pull-ups for port IB input pins [5:0] are always enabled.

PORTIB — Port Data IB Register **$00E6**

	Bit 7	6	5	4	3	2	1	Bit 0
	PIB7	PIB6	PIB5	PIB4	PIB3	PIB2	PIB1	PIB0
IIC	SCL	SDA	-	-	-	-	-	-
RESET:	-	-	-	-	-	-	-	-

Read and write anytime.

IIC functions SCL and SDA share port IB pins 7 and 6 and take precedence over the general-purpose port when IIC is enabled. The SCL and SDA output buffers behave as open-drain outputs.

When port is configured as input, a read will return the pin level. Port bits 5 through 0 have internal pull ups when configured as inputs so they will read ones.

When configured as output, a read will return the latched output data. Port bits 5 through 0 will read the last value written. A write will drive associated pins only if configured for output and IIC is not enabled.

Port bits 5 through 0 do not have available external pins for MC68HC912DT128A.

DDRIB — Data Direction for Port IB Register **$00E7**

	Bit 7	6	5	4	3	2	1	Bit 0
	DDRIB7	DDRIB6	DDRIB5	DDRIB4	DDRIB3	DDRIB2	DDRIB1	DDRIB0
RESET:	0	0	0	0	0	0	0	0

Read and write anytime

DDRIB[7:2]— Port IB [7:2] Data direction

Each bit determines the primary direction for each pin configured as general-purpose I/O.

0 = Associated pin is a high-impedance input.

1 = Associated pin is an output.

DDRIB[5:0] — These bits served as memory locations since there are no corresponding external port pins for MC68HC912DT128A.

IIC Programming Examples

Initialization Sequence

Reset will put the IIC Bus Control Register to its default status. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the Frequency Divider Register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the IIC Bus Address Register (IBAD) to define its slave address.
3. Set the IBEN bit of the IIC Bus Control Register (IBCR) to enable the IIC interface system.
4. Modify the bits of the IIC Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.

Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is

connected to a multi-master bus system, the state of the IIC Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period it may be necessary to wait until the IIC is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of a program which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
CHFLAG    BRSET   IBSR,#$20,*    ;WAIT FOR IBB FLAG TO CLEAR
TXSTART   BSET    IBCR,#$30      ;SET TRANSMIT AND MASTER MODE
                                 ;i.e. GENERATE START CONDITION
          MOVB    CALLING,IBDR   ;TRANSMIT THE CALLING
                                 ;ADDRESS, D0=R/W
IBFREE    BRCLR   IBSR,#$20,*    ;WAIT FOR IBB FLAG TO SET
```

Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The IIC Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. Software must clear the IBIF bit in the interrupt routine first. The TCF bit will be cleared by reading from the IIC Bus Data I/O Register (IBDR) in receive mode or writing to IBDR in transmit mode.

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If

master receive mode is required, indicated by R/$\overline{W}$ bit in IBDR, then the Tx/$\overline{Rx}$ bit should be toggled at this stage.

During slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/$\overline{Rx}$ bit is programmed accordingly. For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/$\overline{Rx}$ bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine (see Figure 40).

```
ISR        BCLR    IBSR,#$02          ;CLEAR THE IBIF FLAG
           BRCLR   IBCR,#$20,SLAVE    ;BRANCH IF IN SLAVE MODE
           BRCLR   IBCR,#$10,RECEIVE  ;BRANCH IF IN RECEIVE MODE
           BRSET   IBSR,#$01,END      ;IF NO ACK, END OF TRANSMISSION
TRANSMIT   MOVB    DATABUF,IBDR       ;TRANSMIT NEXT BYTE OF DATA
```

Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
MASTX    TST     TXCNT           ;GET VALUE FROM THE
                                 ;TRANSMITING COUNTER
         BEQ     END             ;END IF NO MORE DATA
         BRSET   IBSR,#$01,END   ;END IF NO ACK
         MOVB    DATABUF,IBDR    ;TRANSMIT NEXT BYTE OF DATA
         DEC     TXCNT           ;DECREASE THE TXCNT
         BRA     EMASTX          ;EXIT
END      BCLR    IBCR,#$20       ;GENERATE A STOP CONDITION
EMASTX   RTI                     ;RETURN FROM INTERRUPT
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP

signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

```
MASR     DEC     RXCNT        ;DECREASE THE RXCNT
         BEQ     ENMASR       ;LAST BYTE TO BE READ
         MOVB    RXCNT,D1     ;CHECK SECOND LAST BYTE
         DEC     D1           ;TO BE READ
         BNE     NXMAR        ;NOT LAST OR SECOND LAST
LAMAR    BSET    IBCR,#$08    ;SECOND LAST, DISABLE ACK
                              ;TRANSMITTING
         BRA     NXMAR
ENMASR   BCLR    IBCR,#$20    ;LAST ONE, GENERATE 'STOP' SIGNAL
NXMAR    MOVB    IBDR,RXBUF   ;READ DATA AND STORE
         RTI
```

Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
RESTART   BSET   IBCR,#$04       ANOTHER START (RESTART)
          MOVB   CALLING,IBDR    ;TRANSMIT THE CALLING ADDRESS
                                 ;D0=R/W
```

Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received (see Figure 40). If IAAS is set, software should set the transmit/receive mode select bit (Tx/$\overline{Rx}$ bit of IBCR) according to the R/$\overline{W}$ command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means

an ’end of data’ signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/$\overline{SL}$=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/$\overline{SL}$ bit from 1 to 0 without generating STOP condition; generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

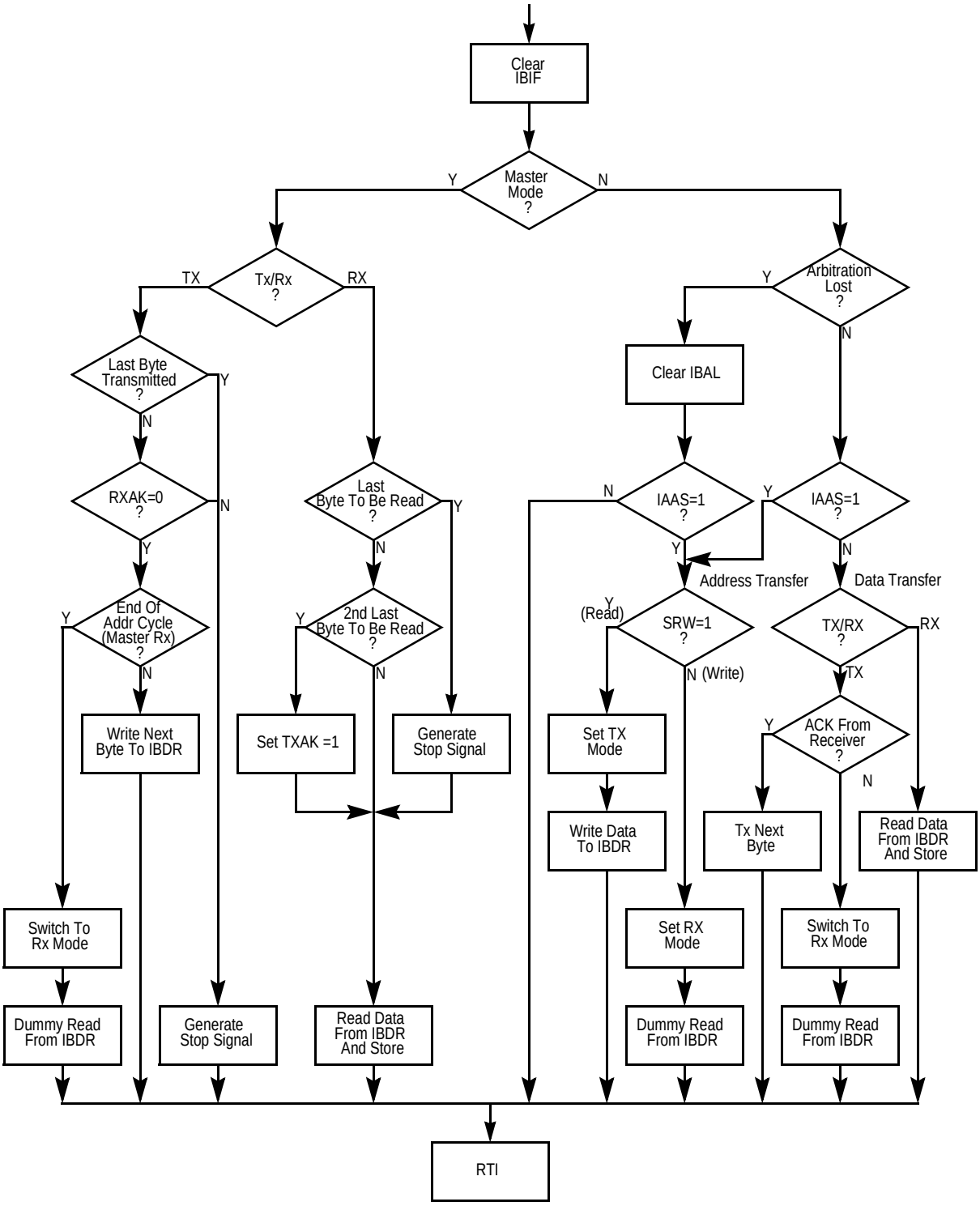


Figure 40 Flow-Chart of Typical IIC Interrupt Routine

MSCAN Controller

Contents

Introduction

The MC68HC912DT128A has three identical msCAN12 modules, identified as CAN0, CAN1 and CAN2. The MC68HC912DG128A has two: CAN0 and CAN1. The information to follow describes one msCAN unless specifically noted and register locations specifically relate to CAN0. CAN1 registers are located 512 bytes from CAN0, and on the MC68HC912DT128A, CAN2 registers are located 256 bytes from CAN0.

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 behavior and simplifies the application software.

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 msCAN12 Port CAN Control Register (PCTLCAN) and msCAN12 Port CAN Data Direction Register (DDRCAN)).

A typical CAN system with msCAN12 is shown in Figure 41 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.

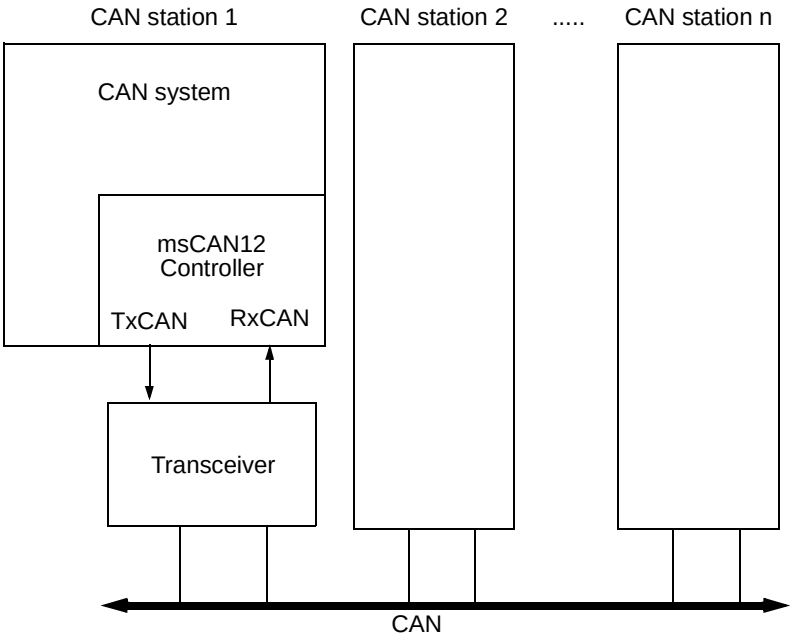


Figure 41 The CAN System

Message Storage

msCAN12 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

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 such that if more than one message is ready to be sent, the highest priority message will be sent out first.

Above behavior 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 prioritizing which the msCAN12 implements with the local priority concept described below.

Receive Structures

The received messages are stored in a two stage input FIFO. The two message buffers are mapped using a ping-pong arrangement into a single memory area (see Figure 42). While the background receive buffer (RxBG) is exclusively associated to the msCAN12, the foreground receive buffer (RxFG) is addressed 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 Programmer's Model of Message Storage).

The receiver full flag (RXF) in the msCAN12 receiver flag register (CRFLG) (see msCAN12 Receiver Flag Register (CRFLG)) 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, it copies the

content of RxBG into RxFG[1], sets the RXF flag, and emits a receive interrupt to the CPU[2]. 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.

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 condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and a further correctly received message with accepted identifier is received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. As long as both buffers remain filled, the msCAN12 is able to transmit messages but it will discard all incoming messages.

NOTE: *The 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.*

1. Only if the RXF flag is not set.

2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

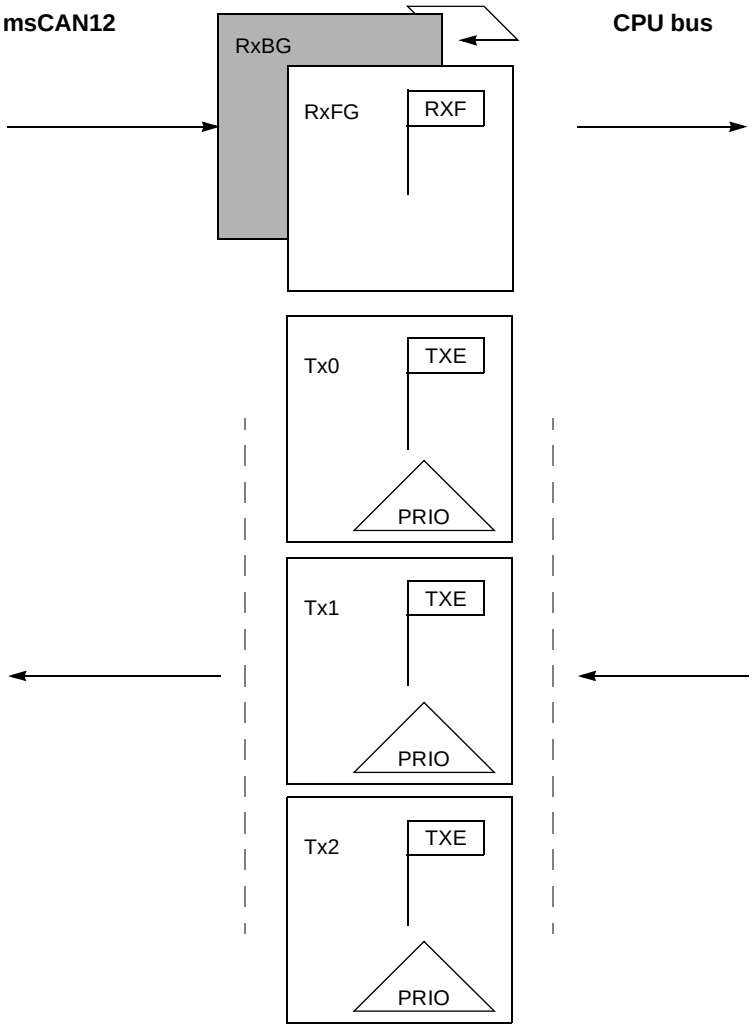


Figure 42 User Model for Message Buffer Organization

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 42.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers (see Programmer's Model of Message Storage). An additional transmit buffer priority register (TBPR) contains an 8-bit so called local priority field (PRIO) (see Transmit Buffer Priority Registers (TBPR)).

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 msCAN12 Transmitter Flag Register (CTFLG)).

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[1] when TXE is set and this 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 prioritizing. 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 grants 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

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

the setting of the ABTAK flag whether the message was actually aborted (ABTAK=1) or sent in the meantime (ABTAK=0).

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 43 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 44 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 45 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.

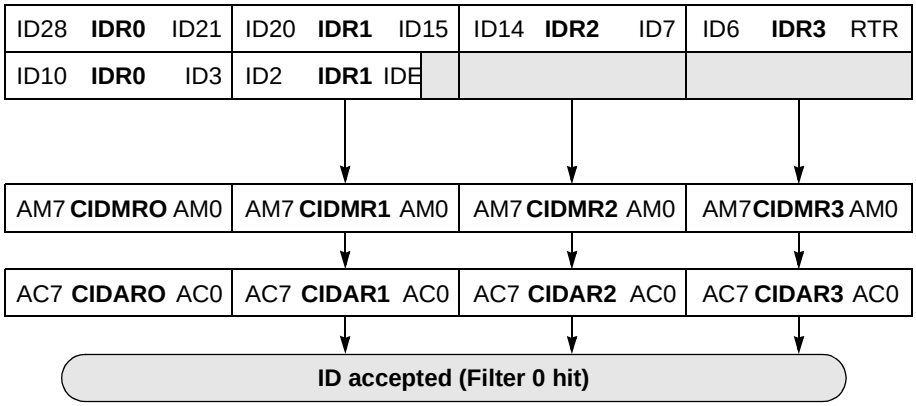


Figure 43 32-bit Maskable Identifier Acceptance Filters

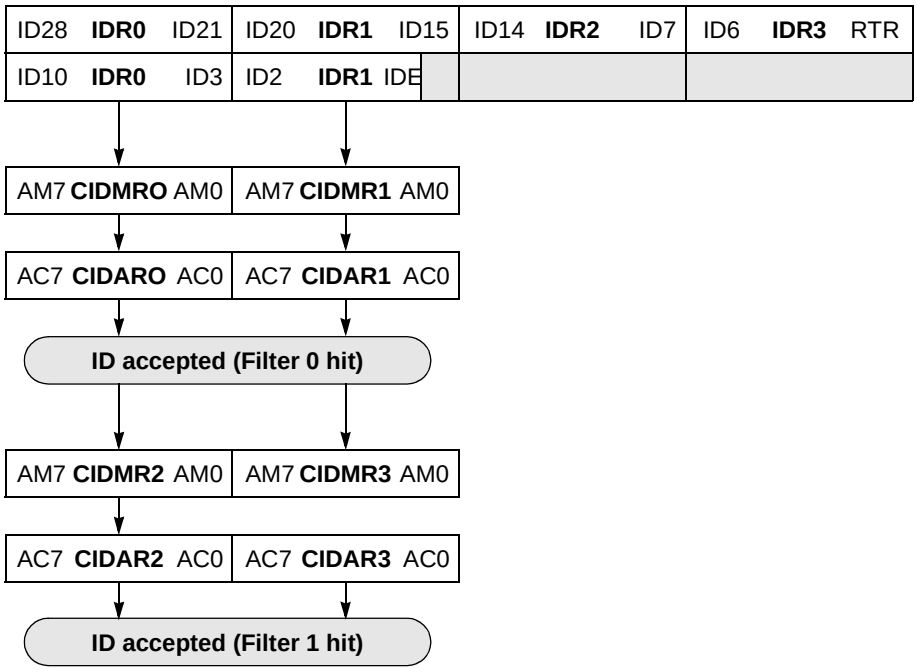


Figure 44 16-bit Maskable Acceptance Filters

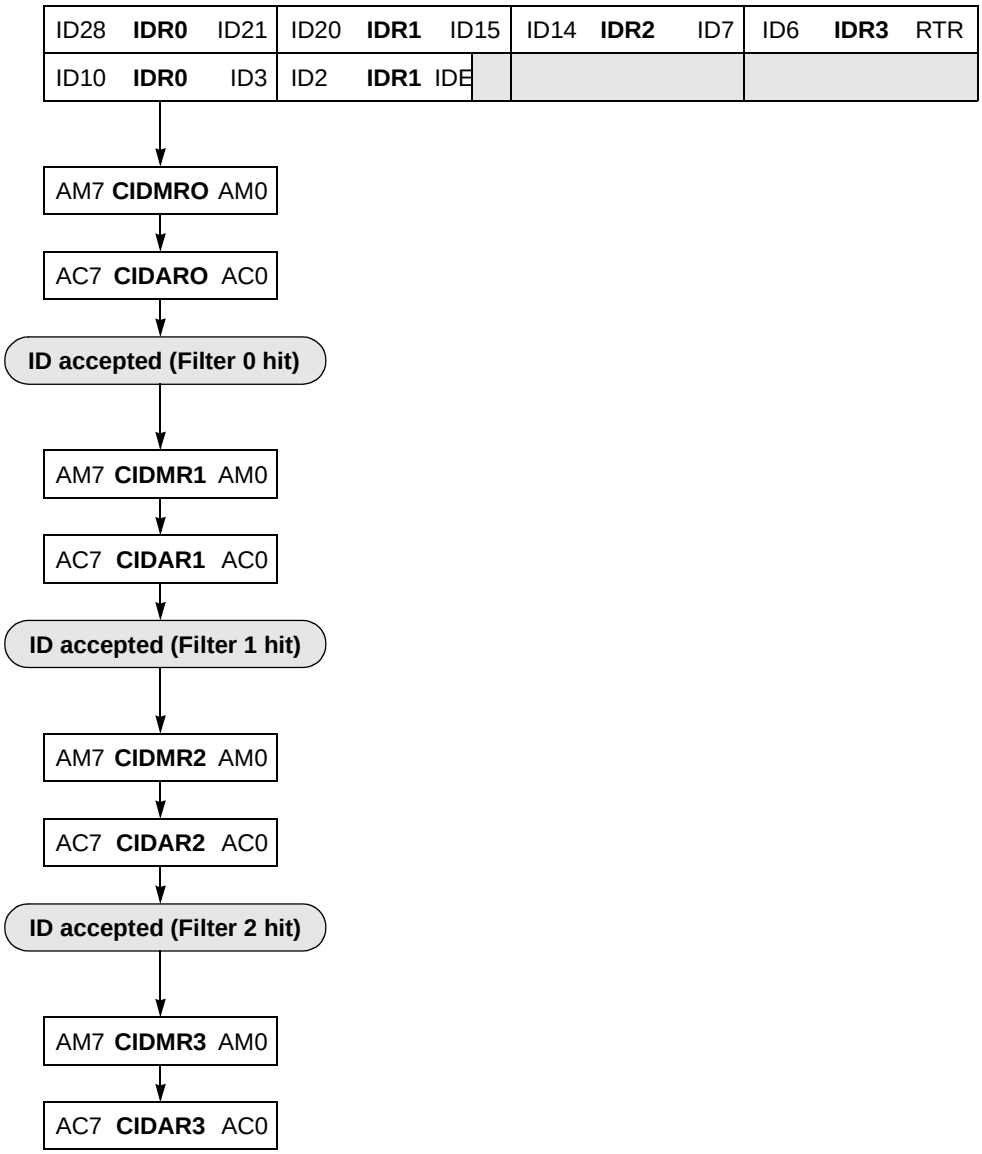


Figure 45 8-bit Maskable Acceptance Filters

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 msCAN12 Receiver Flag Register (CRFLG)) and

three bits in the identifier acceptance control register (see msCAN12 Identifier Acceptance Control Register (CIDAC)). 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.

Interrupts

The msCAN12 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see msCAN12 Receiver Flag Register (CRFLG) to msCAN12 Transmitter Control Register (CTCR)):

- *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 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 BUSOFF state.

Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the msCAN12 receiver flag register (CRFLG) or the msCAN12 transmitter flag register (CTFLG). 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.

NOTE: *Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.*

Interrupt Vectors

The msCAN12 supports four interrupt vectors as shown in Table 38. The vector addresses and the relative interrupt priority are dependent on the chip integration and to be defined.

Table 38 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	

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 cannot 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 msCAN12 Module Control Register (CMCR0)) 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 when the msCAN12 is in any of the low power modes.

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 instructions put the MCU in low power consumption stand-by modes. Table 39 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 39 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.

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 46). 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 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

The application software must avoid setting 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 SLPAK as a handshake indication for the request (SLPRQ) to go into SLEEP mode. When in SLEEP mode, the msCAN12 stops its internal clocks. Clocks to allow register accesses will still run. The TxCAN pin will stay in recessive state. If RXF=1, the message can be read and RXF can be cleared. However, if the msCAN12 is in bus-off state, it stops counting 128*11 consecutive recessive bits due to the stopped clocks. Likewise, copying of RxBG into RxFG will not take place while in sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort will take place while in sleep mode.

The msCAN12 will leave SLEEP mode (wake-up) when

- bus activity occurs or
- the MCU clears the SLPRQ bit or
- the MCU sets SFTRES.

NOTE: *The MCU cannot clear the SLPRQ bit before the msCAN12 is in SLEEP mode (SLPAK = 1).*

After wake-up, the msCAN12 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the msCAN12 is woken-up by a CAN frame, this frame will not be received. The receive message buffers (RxBG and RxFG) will contain messages if they were received before sleep mode was entered. Pending copying of RxBG and RxFG,

as well as pending message aborts and pending message transmissions, will now be executed. If the msCAN12 is still in bus-off state after sleep mode was left, it continues counting the 128*11 consecutive recessive bits.

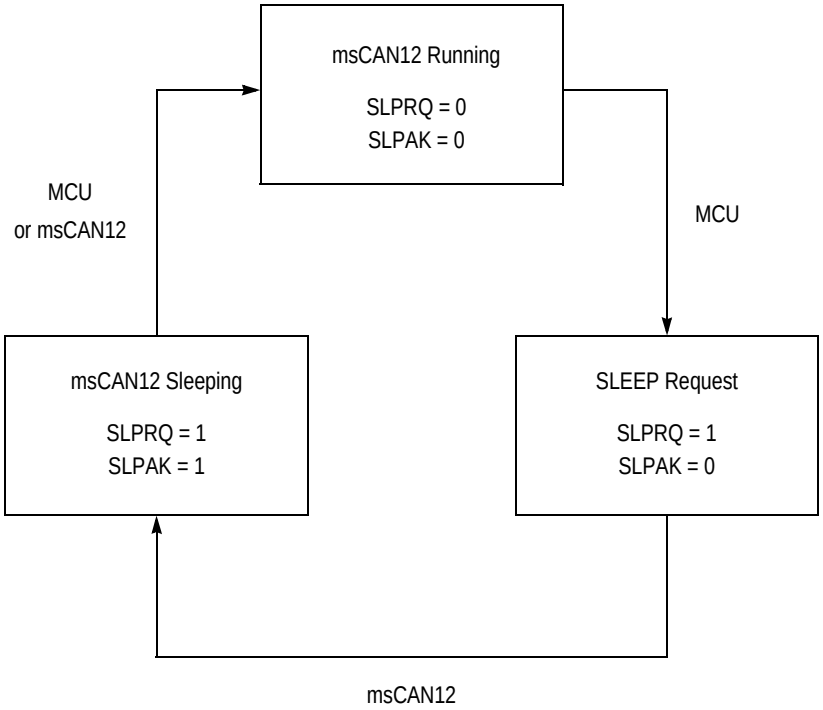


Figure 46 SLEEP Request / Acknowledge Cycle

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 msCAN12 Module Control Register (CMCR0) for a complete description of the SOFT_RESET mode.

When setting the SFTRES bit, the msCAN12 immediately stops all ongoing transmissions and receptions, potentially causing the CAN protocol violations. The user is responsible to take care that the msCAN12 is not active when SOFT_RESET mode is entered. The recommended procedure is to bring the msCAN12 into SLEEP mode before the SFTRES bit is set.

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 msCAN12 Module Control Register (CMCR0)).

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.

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, msCAN12 Module Control Register (CMCR0)). 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.

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 module (ECT). Under the control of the timer link enable (TLNKEN) bit in the CMCR0, this signal will be connected to the ECT timer channel m input[1].

After ECT timer has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

Clock System

Figure 47 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.

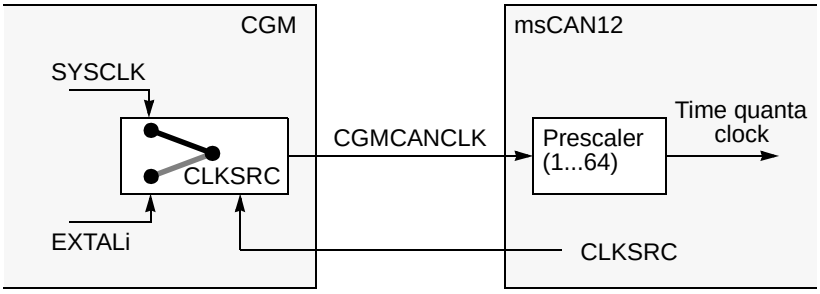


Figure 47 Clocking Scheme

The clock source bit (CLKSRC) in the msCAN12 module control register (CMCR1) (see msCAN12 Bus Timing Register 0 (CBTR0)) defines whether the msCAN12 is connected to the output of the crystal oscillator (EXTALi) or to the system clock (SYSCLK).

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.

1. The timer channel being used for the timer link for CAN0 is channel 4 and for CAN1 is channel 5.

For microcontrollers without the CGM module, CGMCANCLK is driven from the crystal oscillator (EXTALi).

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[1]:

- 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 msCAN12 Bus Timing Register 0 (CBTR0) and msCAN12 Bus Timing Register 1 (CBTR1)).

It is the user's responsibility to make sure that his bit time settings are in compliance with the CAN standard. Figure 49 gives an overview on the CAN conforming segment settings and the related parameter values.

1. For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.

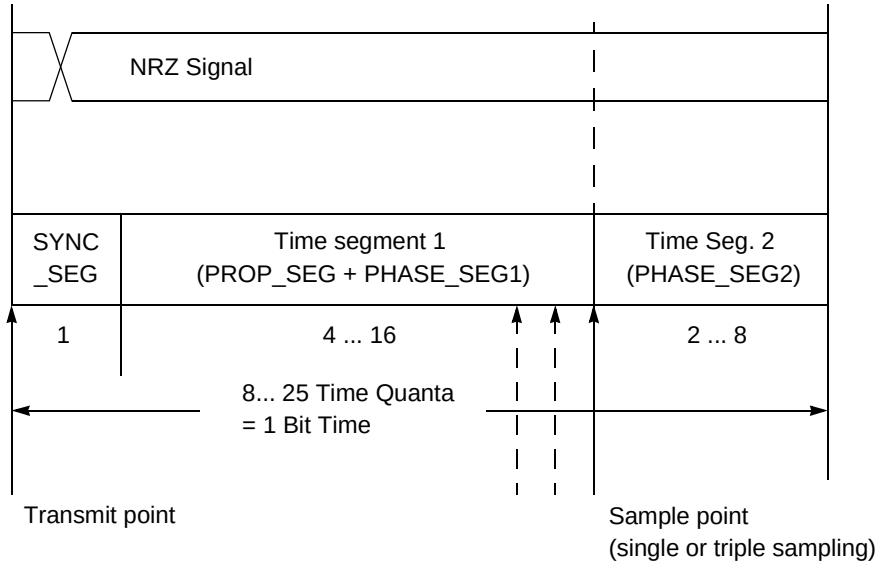


Figure 48 Segments within the Bit Time

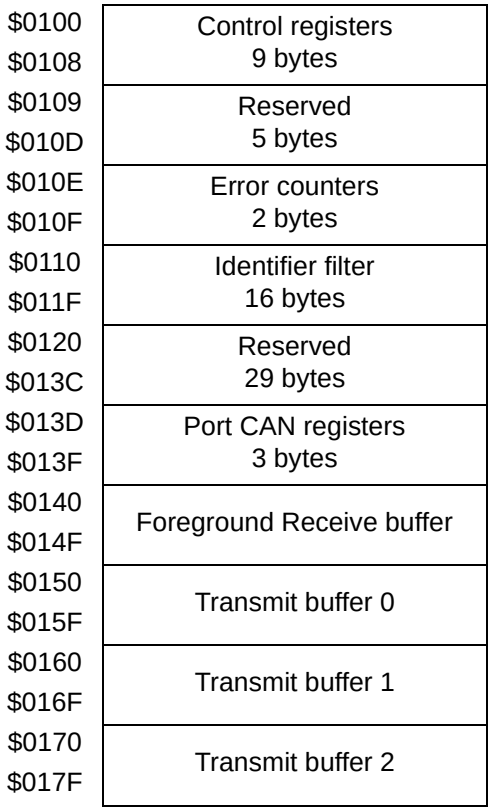
Figure 49 CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchron. 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

Memory Map

The msCAN12 occupies 128 bytes in the CPU12 memory space. The background receive buffer can only be read in test mode.

Figure 50 msCAN12 Memory Map



Programmer's Model of Message Storage

The following section details the organization 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 bytes in the memory map containing a 13 byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

Figure 51 Message Buffer Organization

Address (1)	Register name
01x0	Identifier register 0
01x1	Identifier register 1
01x2	Identifier register 2
01x3	Identifier register 3
01x4	Data segment register 0
01x5	Data segment register 1
01x6	Data segment register 2
01x7	Data segment register 3
01x8	Data segment register 4
01x9	Data segment register 5
01xA	Data segment register 6
01xB	Data segment register 7
01xC	Data length register
01xD	Transmit buffer priority register(2)
01xE	Unused
01xF	Unused

1. x is 4, 5, 6, or 7 depending on which buffer RxFG, Tx0, Tx1, or Tx2 respectively.
2. Not applicable for receive buffers

Message Buffer Outline

Figure 52 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 53. All bits of the 13 byte data structure are undefined out of reset.

NOTE: *The foreground receive buffer can be read anytime but can not be written. The transmit buffers can be read or written anytime.*

Figure 52 Receive/Transmit Message Buffer Extended Identifier

ADDR(1)	REGISTER	R/W	BIT 7	6	5	4	3	2	1	BIT 0
$01x0	IDR0	R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		W								
$01x1	IDR1	R	ID20	ID19	ID18	SRR (1)	IDE (1)	ID17	ID16	ID15
		W								
$01x2	IDR2	R	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		W								
$01x3	IDR3	R	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		W								
$01x4	DSR0	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01x5	DSR1	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01x6	DSR2	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01x7	DSR3	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01x8	DSR4	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01x9	DSR5	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01xA	DSR6	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01xB	DSR7	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
$01xC	DLR	R					DLC3	DLC2	DLC1	DLC0
		W								

1. x is 4, 5, 6, or 7 depending on which buffer RxFG, Tx0, Tx1, or Tx2 respectively.

Figure 53 Standard Identifier Mapping

ADDR(1)	REGISTER	R/W	BIT 7	6	5	4	3	2	1	BIT 0
$01x0	IDR0	R	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
		W								
$01x1	IDR1	R	ID2	ID1	ID0	RTR	IDE(0)			
		W								
$01x2	IDR2	R								
		W								
$01x3	IDR3	R								
		W								

1. x is 4, 5, 6, or 7 depending on which buffer RxFG, Tx0, Tx1, or Tx2 respectively.

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.

0 = Standard format (11-bit)
1 = Extended format (29-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.

0 = Data frame
1 = Remote frame

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 40 shows the effect of setting the DLC bits.

Table 40 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

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.

Transmit Buffer Priority Registers (TBPR)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TBPR(1)	R	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
$01xD	W								
RESET		-	-	-	-	-	-	-	-

1. x is 5, 6, or 7 depending on which buffer Tx0, Tx1, or Tx2 respectively.

PRIO7 – PRIO0 — Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritizing process of the msCAN12 and is defined to be highest for the smallest binary number. The msCAN12 implements the following internal priorization 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.

NOTE: *To ensure data integrity, no registers of the transmit buffers shall be written while the associated TXE flag is cleared.*
To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.

Programmer's Model of Control Registers

Overview

The programmer's model has been laid out for maximum simplicity and efficiency.

msCAN12 Module Control Register (CMCR0)

		Bit 7	6	5	4	3	2	1	Bit 0
CMCR0	R	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
$0100	W								
RESET		0	0	1	0	0	0	0	1

CSWAI — CAN Stops in Wait Mode

0 = The module is not affected during WAIT mode.

1 = The module ceases to be clocked 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.

0 = msCAN12 is not synchronized to the CAN bus

1 = msCAN12 is 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 Timer Link).

0 = The port is connected to the timer input.

1 = The msCAN12 timer signal output 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 msCAN12 SLEEP Mode).

0 = Wake-up – The msCAN12 is not in SLEEP Mode.

1 = SLEEP – The msCAN12 is in SLEEP Mode.

SLPRQ — SLEEP request

This flag allows to request the msCAN12 to go into an internal power-saving mode (see msCAN12 SLEEP Mode).

0 = Wake-up – The msCAN12 will function normally.

1 = SLEEP request – The msCAN12 will go into SLEEP Mode.

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, CIDAR0–3, CIDMR0–3 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 BUSOFF state it will be synchronized after 11 recessive bits on the bus; if the msCAN12 is in BUSOFF 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.

0 = Normal operation

1 = msCAN12 in SOFT_RESET state.

msCAN12 Module Control Register (CMCR1)

		Bit 7	6	5	4	3	2	1	Bit 0
CMCR1	R	0	0	0	0	0	LOOPB	WUPM	CLKSRC
$0101	W								
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 bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

0 = Normal operation

1 = Activate loop back self test mode

WUPM — Wake-Up Mode

This flag defines whether the integrated low-pass filter is applied to protect the msCAN12 from spurious wake-ups (see Programmable Wake-Up Function).

0 = msCAN12 will wake up the CPU after any recessive to dominant edge on the CAN bus.

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}.

CLKSRC — msCAN12 Clock Source

This flag defines which clock source the msCAN12 module is driven from (only for system with CGM module; see Clock System, Figure 47).

0 = The msCAN12 clock source is EXTALi.

1 = The msCAN12 clock source is SYSCLK, twice the frequency of ECLK.

NOTE: *The CMCR1 register can only be written if the SFTRES bit in CMCR0 is set.*

msCAN12 Bus Timing Register 0 (CBTR0)

		Bit 7	6	5	4	3	2	1	Bit 0
CBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
$0102	W								
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 41).

Table 41 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 42.

Table 42 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.*

msCAN12 Bus Timing Register 1 (CBTR1)

		Bit 7	6	5	4	3	2	1	Bit 0
CBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
$0103	W								
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.

0 = One sample per bit.

1 = Three samples per bit[1].

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 43 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 44.

Table 44 Time segment values

TSEG13	TSEG12	TSEG11	TSEG10	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

TSEG22	TSEG21	TSEG20	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.*

msCAN12 Receiver Flag Register (CRFLG)

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

1. In this case PHASE_SEG1 must be at least 2 TimeQuanta.

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
CRFLG	R	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
$0104	W								
RESET		0	0	0	0	0	0	0	0

WUPIF — Wake-up Interrupt Flag

If the msCAN12 detects bus activity while 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.

0 = No wake-up activity has been observed while in SLEEP Mode.
1 = msCAN12 has detected activity on the bus and requested wake-up.

RWRNIF — Receiver Warning Interrupt Flag

This bit will be set when the msCAN12 goes into warning status due to the Receive Error counter (REC) exceeding 96 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set[1]. If not masked, an Error interrupt is pending while this flag is set.

0 = No receiver warning status has been reached.
1 = msCAN12 went into receiver warning status.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the msCAN12 goes into warning status due to the Transmit Error counter (TEC) exceeding 96 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set[2]. If not masked, an Error interrupt is pending while this flag is set.

0 = No transmitter warning status has been reached.
1 = msCAN12 went into transmitter warning status.

1. RWRNIF = (96 ≤ REC) & $\overline{\text{RERRIF}}$& $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$

2. TWRNIF = (96 ≤ TEC) & $\overline{\text{RERRIF}}$& $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$

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 (REC) exceeding 127 and the Bus-Off interrupt flag is not set[1]. If not masked, an Error interrupt is pending while this flag is set.

0 = No receiver error passive status has been reached.
1 = msCAN12 went into receiver error passive status.

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 (TEC) exceeding 127 and the Bus-Off interrupt flag is not set[2]. If not masked, an Error interrupt is pending while this flag is set.

0 = No transmitter error passive status has been reached.
1 = msCAN12 went into transmitter error passive status.

BOFFIF — BUSOFF Interrupt Flag

This bit will be set when the msCAN12 goes into BUSOFF status, due to the Transmit Error counter exceeding 255. It cannot be cleared before the msCAN12 has monitored 128*11 consecutive recessive bits on the bus. If not masked, an Error interrupt is pending while this flag is set.

0 = No BUSOFF status has been reached.
1 = msCAN12 went into BUSOFF status.

OVRIF — Overrun Interrupt Flag

This bit will be set when a data overrun condition occurred. If not masked, an Error interrupt is pending while this flag is set.

0 = No data overrun has occurred.
1 = A data overrun has been detected.

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

1. RERRIF = (128 $\leq$ REC $\leq$ 255) & $\overline{\text{BOFFIF}}$
2. TERRIF = (128 $\leq$ TEC $\leq$ 255) & $\overline{\text{BOFFIF}}$

has read that message from the receive buffer the RXF flag must be handshaked (cleared) in order 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.

0 = The receive buffer is released (not full).

1 = The receive buffer is full. A new message is available.

NOTE: *The CRFLG register is held in the reset state if the SFTRES bit in CMCR0 is set.*

msCAN12 Receiver Interrupt Enable Register (CRIER)

		Bit 7	6	5	4	3	2	1	Bit 0
CRIER	R	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
$0105	W								
RESET		0	0	0	0	0	0	0	0

WUPIE — Wake-up Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A wake-up event will result in a wake-up interrupt.

RWRNIE — Receiver Warning Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A receiver warning status event will result in an error interrupt.

TWRNIE — Transmitter Warning Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A transmitter warning status event will result in an error interrupt.

RERRIE — Receiver Error Passive Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A receiver error passive status event will result in an error interrupt.

TERRIE — Transmitter Error Passive Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A transmitter error passive status event will result in an error interrupt.

BOFFIE — BUSOFF Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A BUSOFF event will result in an error interrupt.

OVRIE — Overrun Interrupt Enable

0 = No interrupt will be generated from this event.

1 = An overrun event will result in an error interrupt.

RXFIE — Receiver Full Interrupt Enable

0 = No interrupt will be generated from this event.

1 = A receive buffer full (successful message reception) event will result in a receive interrupt.

NOTE: *The CRIER register is held in the reset state if the SFTRES bit in CMCR0 is set.*

msCAN12 Transmitter Flag Register (CTFLG)

The Abort Acknowledge flags are read only. The Transmitter Buffer Empty flags are read and clear only. A flag can be cleared by writing a1 to the corresponding bit position. Writing a zero has no effect on the flag setting. The Transmitter Buffer Empty flags each have an associated interrupt enable flag in the CTCR register. A hard or soft reset will reset the register.

		Bit 7	6	5	4	3	2	1	Bit 0
CTFLG	R	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
$0106	W								
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.

0 = The massage has not been aborted, thus has been sent out.

1 = The message has been aborted.

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 (msCAN12 Transmitter Control Register (CTCR)). If not masked, a transmit interrupt is pending while this flag is set.

Clearing this flag will also clear the corresponding Abort Acknowledge flag (ABTAK, see above). Setting this flag will clear the corresponding Abort Request flag (ABTRQ, msCAN12 Transmitter Control Register (CTCR)).

0 = The associated message buffer is full (loaded with a message due for transmission).

1 = The associated message buffer is empty (not scheduled).

NOTE: *The CTFLG register is held in the reset state if the SFTRES bit in CMCR0 is set.*

msCAN12 Transmitter Control Register (CTCR)

		Bit 7	6	5	4	3	2	1	Bit 0
CTCR	R	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
$0107	W								
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 msCAN12 Transmitter Flag Register (CTFLG)) 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.

0 = No abort request.

1 = Abort request pending.

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

0 = No interrupt will be generated from this event.

1 = A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.

NOTE: *The CTCR register is held in the reset state if the SFTRES bit in CMCR0 is set.*

msCAN12 Identifier Acceptance Control Register (CIDAC)

		Bit 7	6	5	4	3	2	1	Bit 0
CIDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
$0108	W								
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 organization (see Identifier Acceptance Filter). Table 44 summarizes the different settings. In Filter Closed mode no messages will be accepted such that the foreground buffer will never be reloaded.

Table 45 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 Identifier Acceptance Filter). Table 44 summarizes the different settings.

Table 46 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.*

msCAN12 Receive Error Counter (CRXERR)

		Bit 7	6	5	4	3	2	1	Bit 0
CRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
$010E	W								
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.

msCAN12 Transmit Error Counter (CTXERR)

		Bit 7	6	5	4	3	2	1	Bit 0
CTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
$010F	W								
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 must only be read when in SLEEP or SOFT_RESET mode.*

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 (CIDMR0/1 and CIDAR0/1) are applied. In the latter case it is required to program the three last bits (AM2 – AM0) in the mask register CIDMR1 to 'don't care'.

Figure 54 Identifier Acceptance Registers (1st bank)

		Bit 7	6	5	4	3	2	1	Bit 0
CIDAR0 $0110	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR1 $0111	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR2 $0112	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR3 $0113	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
RESET		-	-	-	-	-	-	-	-

Figure 55 Identifier Acceptance Registers (2nd bank)

		Bit 7	6	5	4	3	2	1	Bit 0
CIDAR4 $0118	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR5 $0119	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR6 $011A	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR7 $011B	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
RESET		-	-	-	-	-	-	-	-

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.*

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.

Figure 56 Identifier Mask Registers (1st bank)

		Bit 7	6	5	4	3	2	1	Bit 0
CIDMR0 $0114	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR1 $0115	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR2 $0116	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR3 $0117	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
RESET		-	-	-	-	-	-	-	-

Figure 57 Identifier Mask Registers (2nd bank)

		Bit 7	6	5	4	3	2	1	Bit 0
CIDMR4 $011C	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR5 $011D	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR6 $011E	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR7 $011F	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
RESET		-	-	-	-	-	-	-	-

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.

0 = Match corresponding acceptance code register and identifier bits.
1 = Ignore corresponding acceptance code register bit.

NOTE: *The CIDMR0–7 registers can only be written if the SFTRES bit in CMCR0 is set.*

msCAN12 Port CAN Control Register (PCTLCAN)

		Bit 7	6	5	4	3	2	1	Bit 0
PCTLCAN	R	0	0	0	0	0	0	PUPCAN	RDPCAN
$013D	W								
RESET		0	0	0	0	0	0	0	0

The following bits control pins 7 through 2 of Port CAN when they are implemented externally.

PUPCAN — Pull-Up Enable Port CAN

0 = Pull mode disabled for Port CAN.
1 = Pull mode enabled for Port CAN.

RDPCAN — Reduced Drive Port CAN

0 = Reduced drive disabled for Port CAN.
1 = Reduced drive enabled for Port CAN.

msCAN12 Port CAN Data Register (PORTCAN)

		Bit 7	6	5	4	3	2	1	Bit 0
PORTCAN	R	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN
$013E	W								
RESET		-		-		-		-	-

Port bits 7 to 2 will read zero when configured as inputs because they are not implemented externally.

When configured as output, port bits 7 to 2 will read the last value written.

Reading bits 1 and 0 returns the value of the TxCan and RxCan pins, respectively.

msCAN12 Port CAN Data Direction Register (DDRCAN)

		Bit 7	6	5	4	3	2	1	Bit 0
DDRCAN	R	DDCAN7	DDCAN6	DDCAN5	DDCAN4	DDCAN3	DDCAN2	0	0
$013F	W								
RESET		0	0	0	0	0	0	0	0

DDCAN7 – DDCAN2 — This bits served as memory locations since there are no corresponding external port pins.

Analog-To-Digital Converter (ATD)

Contents

Introduction

The MC68HC912DT128A has two identical ATD modules identified as ATD0 and ATD1. Except for the V_{DDA} and V_{SSA} Analog supply voltage, all pins are duplicated and indexed with ‘0’ or ‘1’ in the following description. An ‘x’ indicates either ‘0’ or ‘1’.

The ATD module is an 8-channel, 10-bit or 8-bit, multiplexed-input successive-approximation analog-to-digital converter. It does not require external sample and hold circuits because of the type of charge redistribution technique used. The ATD converter timing can be 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.

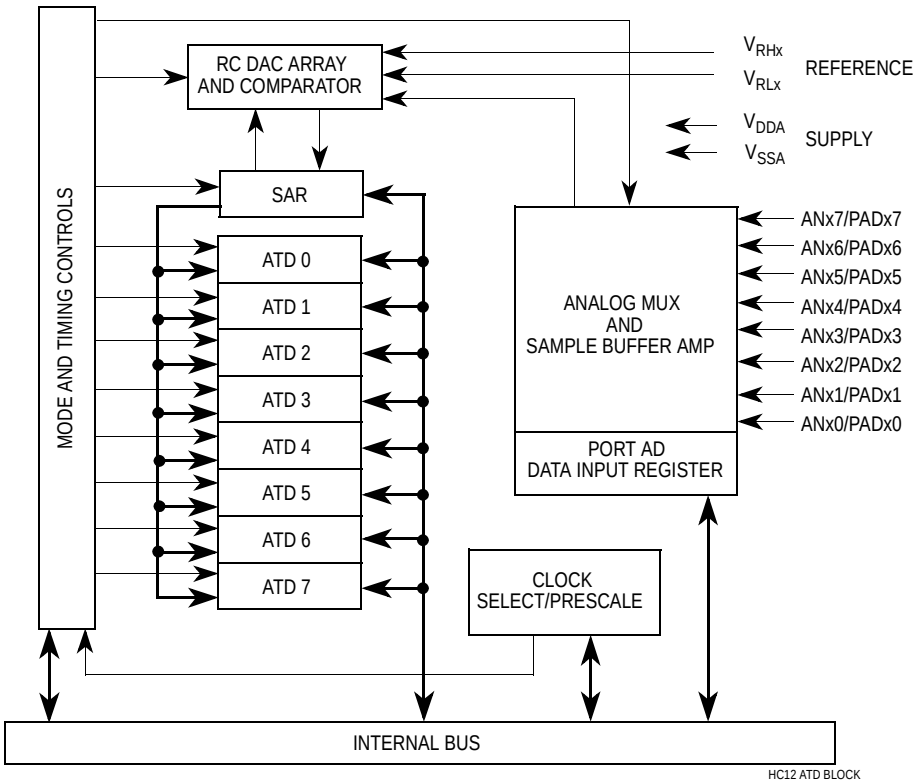


Figure 58 Analog-to-Digital Converter Block Diagram

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.

ATD Registers

Control and data registers for the ATD modules are described below. Both ATDs have identical control registers mapped in two blocks of 16 bytes.

ATD0CTL0/ATD1CTL0 — Reserved **$0060/$01E0**

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

Writes to this register will abort current conversion sequence.
Read or write any time.

ATD0CTL1/ATD1CTL1 — Reserved **$0061/$01E1**

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

WRITE: Write to this register has no meaning.
READ: Special Mode only.

ATD0CTL2/ATD1CTL2 — ATD Control Register 2 **$0062/$01E2**

	Bit 7	6	5	4	3	2	1	Bit 0
	ADPU	AFFC	ASWAI	DJM	DSGN	Reserved	ASCIE	ASCIF
RESET:	0	0	0	0	0	0	0	0

Writes to these registers abort any current conversion sequence.

Read or write anytime except ASCIF bit, which cannot be written.

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 A/D 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 associated 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.

ASWAI — ATD Stops 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

When the ASWAI bit is set and the module enters wait mode, most of the clocks stop and the analog portion powers down. When the module comes out of wait, it is recommended that a stabilization delay (stop and ATD power up recovery time, t_{SR}) is allowed before new conversions are started. Additionally, the ATD does not re-initialize automatically on leaving wait mode.

DJM — Result Register Data Justification Mode

0 = Left justified

1 = Right justified

For 10-bit resolution, left justified mode maps a result register into data bus bits 6 through 15; bit 15 is the MSB. In right justified mode, the result registers maps onto data bus bits 0 through 9; bit 9 is the MSB.

For 8-bit resolution, left justified mode maps a result into the high byte (bits 8 though 15; bit 15 is the MSB). Right justified maps a result into the low byte (bits 0 through 7; bit 7 is the MSB).

DSGN — Signed/Unsigned Result Data Mode

0 = Unsigned result register data

1 = Signed result register data

Note that signed data is not available for right justified data. Therefore, sign extension hardware is not required.

Table 47 summarizes the result data formats available and how they are set up using the control bits. Table 48 illustrates the difference between the signed and unsigned, left justified output codes for an input signal range between 0 and 5.1 Volts.

SRES10	DJM	DSGN	Result Data Formats Description and Bus Bit Mapping
0	0	0	8-bit/left justified/unsign - bits 8-15
0	0	1	8-bit/ left justified/signed - bits 8-15
0	1	X	8-bit/right justified/unsign - bits 0-7
1	0	0	10-bit/left justified/unsign - bits 6-15
1	0	1	10-bit/left justified/signed - bits 6-15
1	1	X	10-bit/right justified/unsign - bits 0-9

Table 47 Result Data Formats Available.

Input Signal Vrl = 0 Volts Vrh = 5.12 Volts	Signed 8-Bit Codes	Unsigned 8-Bit Codes	Signed 10-Bit Codes	Unsigned 10-Bit Codes
5.120 Volts	7F	FF	7FC0	FFC0
5.100	7F	FF	7F00	FF00
5.080	7E	FE	7E00	FE00
2.580	01	81	0100	8100
2.560	00	80	0000	8000
2.540	FF	7F	FF00	7F00
0.020	81	01	8100	0100
0.000	80	00	8000	0000

Table 48 Left Justified, Signed and Unsigned ATD Output Codes.

ASCIE — ATD Sequence Complete Interrupt Enable

0 = Disables ATD interrupt

1 = Enables ATD interrupt on sequence complete

ASCIF — ATD Sequence Complete Interrupt Flag

Cannot be written in any mode.

0 = No ATD interrupt occurred

1 = ATD sequence complete

ATD0CTL3/ATD1CTL3 — ATD Control Register 3 **$0063/$01E3**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	S1C	FIFO	FRZ1	FRZ0
RESET:	0	0	0	0	0	0	0	0

Read or write any time.

S1C — Conversion Sequence Length (Least Significant Bit)

This control bit works with control bit S8C in ATDCTL5 in determining how many conversion are performed per sequence. When the S1C bit is set, a sequence length of 1 is defined. However, if the S8C bit is also set, the S8C bit takes precedence. For sequence length coding information see the description for S8C bit in ATDCTL5.

FIFO — Result Register FIFO Mode

0 = result registers do not map to the conversion sequence

1 = result registers map to the conversion sequence

In normal operation, the A/D conversion results map into the result registers based on the conversion sequence; the result of the first conversion appears in the first result register, the second result in the second result register, and so on. In FIFO mode the result register counter is not reset at the beginning or ending of a conversion sequence; conversion results are placed in consecutive result registers between sequences. The result register counter wraps around when it reaches the end of the result register file. The conversion counter value in ATDSTAT0 can be used to determine where in the result register file, the next conversion result will be placed.

The results register counter is initialized to zero on three events: on reset, the beginning of a normal (non-FIFO) conversion sequence, and the end of a normal (non-FIFO) conversion sequence. Therefore, the reset bit in register ATDTEST1 can be toggled to zero the result register counter; any sequence allowed to complete normally will zero

the result register counter; a new sequence (non-FIFO) initiated with a write to ATDCTL4/5 followed by a write to ATDCTL3 to set the FIFO bit will start a FIFO sequence with the result register initialized.

Finally, which result registers hold valid data can be tracked using the conversion complete flags. Fast flag clear mode may or may not be useful in a particular application to track valid data.

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 49 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

ATD0CTL4/ATD1CTL4 — ATD Control Register 4 **$0064/$01E4**

	Bit 7	6	5	4	3	2	1	Bit 0
	RES10	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.

RES10 — 10 bit Mode

0 = 8 bit operation

1 = 1= 10 bit operation

SMP1, SMP0 — Select Sample Time

Used to select one of four sample times after the buffered sample and transfer has occurred.

Table 50 Final Sample Time Selection

SMP1	SMP0	Final Sample Time
0	0	2 A/D clock periods
0	1	4 A/D clock periods
1	0	8 A/D clock periods
1	1	16 A/D 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 at 2 MHz. Table 51 shows the divide-by operation and the appropriate range of system clock frequencies.

Table 51 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			

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.

ATD0CTL5/ATD1CTL5 — ATD Control Register 5 **$0065/$01E5**

	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 any time. Writes to the ATD control registers initiate 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.

S8C — Conversion Sequence Length (Most Significant Bit)

The S8C/S1C bits define the length of a conversion sequence. Table 52 lists the coding combinations implemented.

S8C	S1C	Number of Conversions per Sequence
0	0	4
0	1	1
1	X	8

Table 52 Conversion Sequence Length Coding.

The result register assignments made to a conversion sequence follow a few simple rules. Normally, the first result is placed in the first register; the second result is placed in the second register, and so on. Table 53 presents the result register assignments for the various conversion lengths that are normally made. If FIFO mode is used, the result register assignments differ. The results are placed in consecutive registers between conversion sequences; the result register mapping wraps around when the end of the register file is reached.

Number of Conversions per Sequence	Result Register Assignment
1	ADR0
4	ADR0 through ADR3
8	ADR0 through ADR7

Table 53 Result Register Assignment for Different Conversion Sequences.

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 S8C 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 54.

CD, CC, CB, and CA — Channel Select for Conversion

Table 54 Multichannel Mode Result Register Assignment

S8CM	CD	CC	CB	CA	Channel Signal	Result in ADRxx if MULT = 1
0	0	0	0	0	AN0	ADRx0
			0	1	AN1	ADRx1
			1	0	AN2	ADRx2
			1	1	AN3	ADRx3
0	0	1	0	0	AN4	ADRx0
			0	1	AN5	ADRx1
			1	0	AN6	ADRx2
			1	1	AN7	ADRx3
0	1	0	0	0	Reserved	ADRx0
			0	1	Reserved	ADRx1
			1	0	Reserved	ADRx2
			1	1	Reserved	ADRx3
0	1	1	0	0	V_{RH}	ADRx0
			0	1	V_{RL}	ADRx1
			1	0	$(V_{RH} + V_{RL})/2$	ADRx2
			1	1	TEST/Reserved	ADRx3
1	0	0	0	0	AN0	ADRx0
			0	1	AN1	ADRx1
			1	0	AN2	ADRx2
			1	1	AN3	ADRx3
		1	0	0	AN4	ADRx4
			0	1	AN5	ADRx5
			1	0	AN6	ADRx6
			1	1	AN7	ADRx7
1	1	0	0	0	Reserved	ADRx0
			0	1	Reserved	ADRx1
			1	0	Reserved	ADRx2
			1	1	Reserved	ADRx3
		1	0	0	V_{RH}	ADRx4
			0	1	V_{RL}	ADRx5
			1	0	$(V_{RH} + V_{RL})/2$	ADRx6
			1	1	TEST/Reserved	ADRx7

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.

ATD0STAT0/ATD1STAT0 — ATD Status Register **$0066/$01E6**

	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

ATD0STAT1/ATD1STAT1 — ATD Status Register **$0067/$01E7**

	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.

ATD0TESTH/ATD1TESTH — ATD Test Register **$0068/$01E8**

	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

ATD0TESTL/ATD1TESTL — ATD Test Register **$0069/$01E9**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAR1	SAR0	RST	0	0	0	0	0
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 a 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).

PORTAD0/PORTAD1 — Port AD Data Input Register **$006F/$01EF**

	Bit 7	6	5	4	3	2	1	Bit 0
	PADx7	PADx6	PADx5	PADx4	PADx3	PADx2	PADx1	PADx0
RESET:	-	-	-	-	-	-	-	-

PADx[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/$01F0**
ADRx0L — A/D Converter Result Register 0 **$0071/$01F1**
ADRx1H — A/D Converter Result Register 1 **$0072/$01F2**
ADRx1L — A/D Converter Result Register 1 **$0073/$01F3**
ADRx2H — A/D Converter Result Register 2 **$0074/$01F4**
ADRx2L — A/D Converter Result Register 2 **$0075/$01F5**
ADRx3H — A/D Converter Result Register 3 **$0076/$01F6**
ADRx3L — A/D Converter Result Register 3 **$0077/$01F7**
ADRx4H — A/D Converter Result Register 4 **$0078/$01F8**
ADRx4L — A/D Converter Result Register 4 **$0079/$01F9**
ADRx5H — A/D Converter Result Register 5 **$007A/$01FA**
ADRx5L — A/D Converter Result Register 5 **$007B/$01FB**
ADRx6H — A/D Converter Result Register 6 **$007C/$01FC**
ADRx6L — A/D Converter Result Register 6 **$007D/$01FD**
ADRx7H — A/D Converter Result Register 7 **$007E/$01FE**
ADRx7L — A/D Converter Result Register 7 **$007F/$01FF**

ADRxxH	Bit 15	6	5	4	3	2	1	Bit 8
ADRxxL	Bit 7	Bit 6	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

ADRxxH[15:8], ADRxxL[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.

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.

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.

Development Support

Contents

Introduction

Development support involves complex interactions between MC68HC912DT128A resources and external development systems. The following section concerns instruction queue and queue tracking signals, background debug mode, and instruction tagging.

Instruction Queue

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. This makes it possible to monitor CPU activity on a cycle-by-cycle basis for debugging. 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 55 shows the meaning of data on the pins.

Table 55 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

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.

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.

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 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.

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 BCLK cycles per bit (nominal speed). The interface times out if 512 BCLK cycles occur between falling edges from the host. The hardware clears the command register when a 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 59 shows an external host transmitting a logic one or zero to the BKGD pin of a target MC68HC912DT128A 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 B 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.

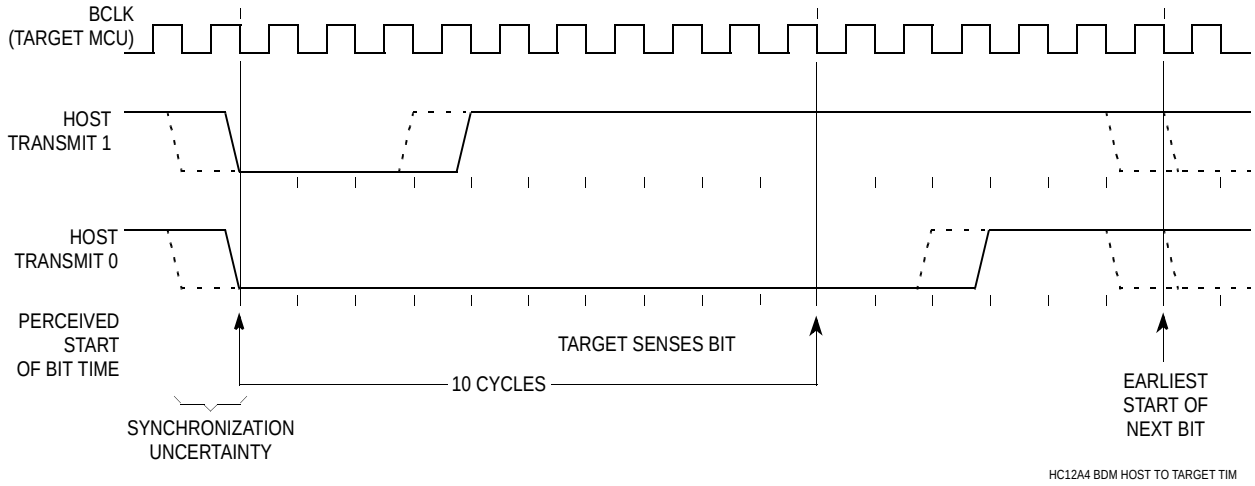


Figure 59 BDM Host to Target Serial Bit Timing

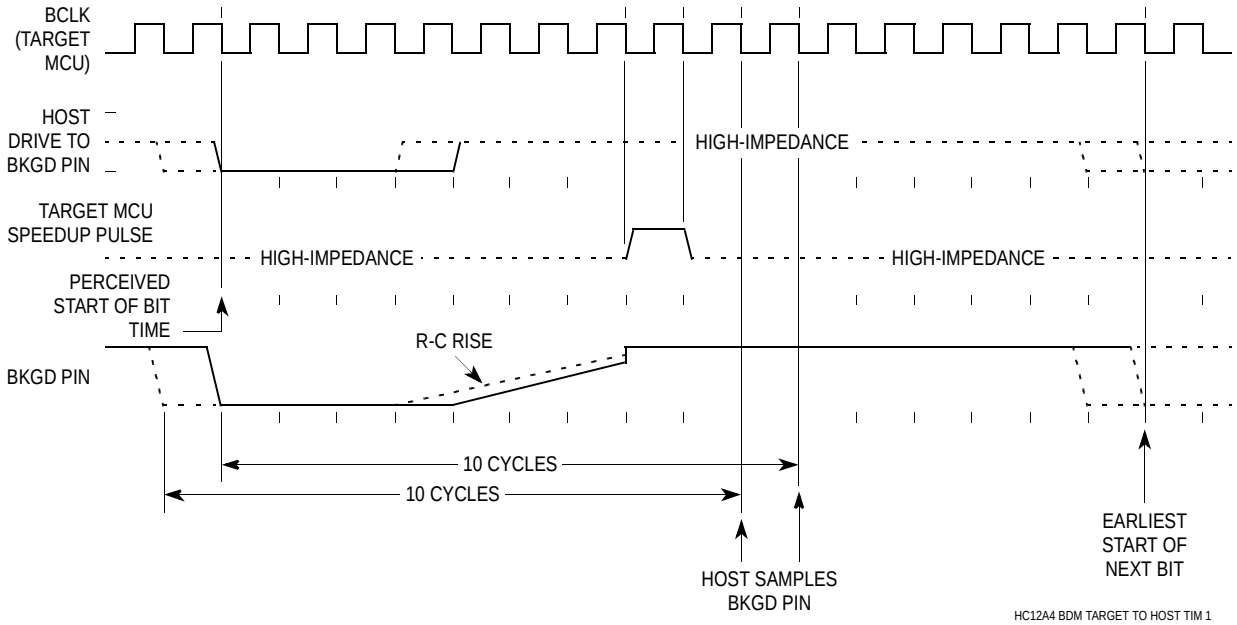


Figure 60 BDM Target to Host Serial Bit Timing (Logic 1)

Figure 60 shows the host receiving a logic one from the target MC68HC912DT128A 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 B 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.

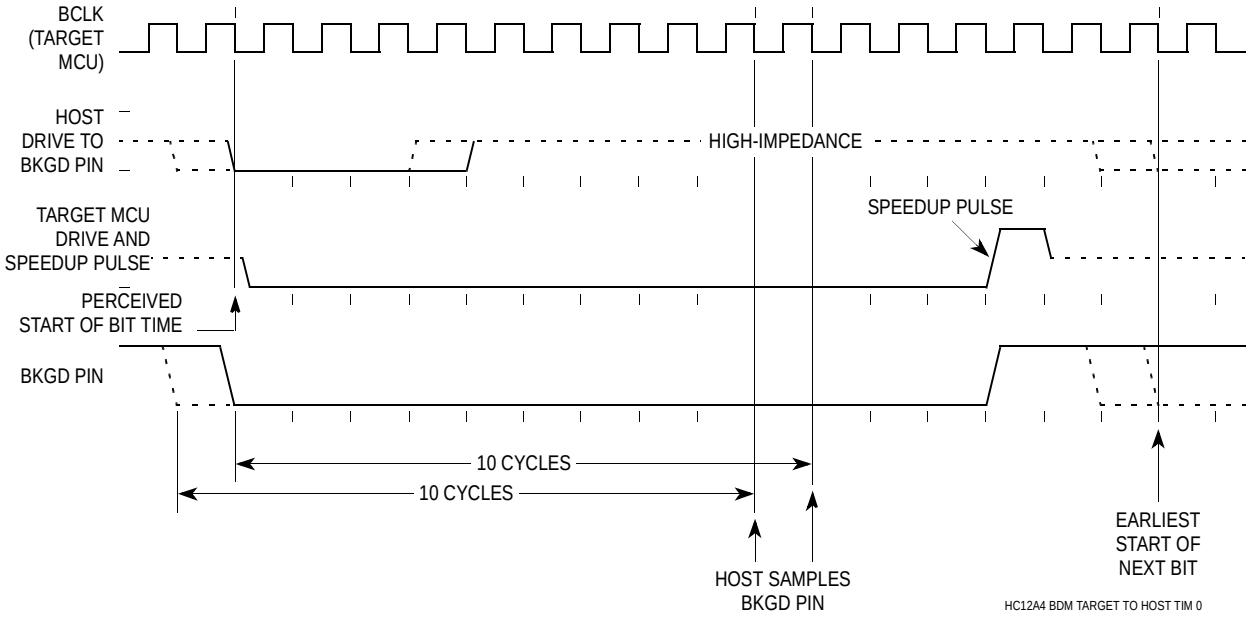


Figure 61 BDM Target to Host Serial Bit Timing (Logic 0)

Figure 61 shows the host receiving a logic zero from the target MC68HC912DT128A 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 MC68HC912DT128A finishes it. Since the target wants the host to receive a logic zero, it drives the BKGD pin low for 13 BCLK 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.

BDM Commands

The BDM command set consists of two types: 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 EEPROM, on-chip I/O and control registers, and external memory that is 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 the background access does not disturb the running application programs. If a free cycle is not found within 128 BCLK cycles, the CPU12 is momentarily frozen so the control logic can steal a cycle.Commands implemented in BDM control logic are listed in Table 56.

Table 56 Hardware Commands(1)

Command	Opcode (Hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware enabled.
READ_BD_BYTE(1)	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.
READ_BD_WORD(1)	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(1)	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.
WRITE_BD_WORD(1)	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.

1. Use these commands only for reading/writing to BDM locations.The BDM firmware ROM and BDM registers are not normally in the HC12 MCU memory map.Since these locations have the same addresses as some of the normal application memory map, there needs to be a way to decide which physical locations are being accessed by the hardware BDM commands.This gives rise to needing separate memory access commands for the BDM locations as opposed to the normal application locations.In logic, this is accomplished by momentarily enabling the BDM memory resources, just for the access cycles of the READ_BD and WRITE_BD commands.This logic allows the debugging system to unobtrusively access the BDM locations even if the application program is running out of the same memory area in the normal application memory map.

The second type of BDM commands are called firmware commands 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 located at $FF00 to $FF06 while BDM is active. The CPU executes code in the BDM firmware 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 57.

Table 57 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 BCLK cycles for a non-intrusive BDM command to execute before another command is sent. This delay includes 128 BCLK 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 BCLK 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 BCLK 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 BCLK 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.

BDM Lockout

The access to the MCU resources by BDM may be prevented by enabling the BDM lockout feature. When enabled, the BDM lockout mechanism prevents the BDM from being active. In this case the BDM ROM is disabled and does not appear in the MCU memory map.

The BDM lockout is enabled by clearing NOBDML bit of EEMCR register. The NOBDML bit is loaded at reset from the SHADOW word of EEPROM module. Modifying the state of the NOBDML and corresponding EEPROM SHADOW bit is only possible in special modes.

Please refer to **EEPROM** for NOBDML information.

Enabling the BDM lockout

Enabling the BDM lockout feature is only possible in special modes (SMODN=0) and is accomplished by the following steps.

1. Remove the SHADOW word protection by clearing SHPROT bit in EEPROT register.
2. Clear NOSHW bit in EEMCR register to make the SHADOW word visible at $0FC0-$0FC1.
3. Program bit 7 of the high byte of the SHADOW word like a regular EEPROM location at address $0FC0 (write $7F into address $0FC0). Do not program other bits of the high byte of the SHADOW word (location $0FC0); otherwise some regular EEPROM array locations will not be visible. At the next reset, the high byte of the SHADOW word is loaded into the EEMCR register. NOBDML bit in EEMCR will be cleared and BDM will not be operational.
4. Protect the SHADOW word by setting SHPROT bit in EEPROT register.

Disabling the BDM lockout

Disabling the BDM lockout is only possible in special modes (SMODN=0) except in special single chip. Follow the same steps as for enabling the BDM lockout, but erase the SHADOW word.

At the next reset, the high byte of SHADOW word is loaded into the EEMCR register. NOBDML bit in EEMCR will be set and BDM becomes operational.

NOTE: *When the BDM lockout is enabled it is not possible to run code from the reset vector in special single chip mode.*

BDM Registers

Seven BDM registers are mapped into the standard 64-Kbyte address space when BDM is active. Mapping is shown in Table 58.

Table 58 BDM registers

Address	Register
$FF00	BDM Instruction Register
$FF01	BDM Status Register

Table 58 BDM registers

Address	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).

STATUS

The STATUS register is read and written by the BDM hardware as a result of serial data shifted in on the BKGD pin.

Read: all modes.

Write: Bits 3 through 5, and bit 7 are writable in all modes. Bit 6, BDMACT, can only be written if bit 7 H/F in the INSTRUCTION register is a zero. Bit 2, CLKSW, can only be written if bit 7 H/F in the INSTRUCTION register is a one. A user would never write ones to bits 3 through 5 because these bits are only used by BDM firmware.

STATUS— BDM Status Register**(1)** **$FF01**

	BIT 7	6	5	4	3	2	1	BIT 0	
	ENBDM	BDMACT	ENTAG	SDV	TRACE	CLKSW	-	-	
RESET:	0 (NOTE 1)	1	0	0	0	0	0	0	Special Single Chip & Periph
RESET:	0	0	0	0	0	0	0	0	All other modes

1. ENBDM is set to 1 by the firmware in Special Single Chip mode.

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 is not active. BDM ROM and registers are not in map.

1 = BDM is active and waiting for serial commands. BDM ROM and registers are in map

The user should be careful that the state of 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 because it would disable the BDM firmware ROM while the CPU12 was 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 — Tagging Enable

Set by the TAGGO command and cleared when BDM mode 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 the BDM firmware.

0 = No valid data. Shift operation is not complete.

1 = Valid Data. Shift operation is complete.

TRACE — Asserted by the TRACE1 command

CLKSW — Clock Switch

0 = BDM system operates with BCLK.

1 = BDM system operates with ECLK.

The WRITE_BD_BYTE@FF01 command that changes CLKSW including 150 cycles after the data portion of the command should be timed at the old speed. Beginning with the start of the next BDM command, the new clock can be used for timing BDM communications.

If ECLK rate is slower than BCLK rate, CLKSW is ignored and BDM system is forced to operate with ECLK.

INSTRUCTION - Hardware Instruction Decode

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.

Read and write: all modes

. The hardware clears the INSTRUCTION register if 512 BCLK cycles occur between falling edges from the host.

INSTRUCTION— BDM Instruction Register (hardware command explanation) **$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 command
1 = Hardware command

DATA — Data Flag - Shows that data accompanies the command.
0 = No data
1 = Data follows the 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 ROM and registers in map

INSTRUCTION — BDM Instruction Register (firmware command bit explanation) **$FF00**

Bit 7	6	5	4	3	2	1	Bit 0
H/F	DATA	R/W	TTAGO		REGN		

The bits in the BDM instruction register have the following meanings when a **firmware** command is executed.

H/F — Hardware/Firmware Flag
0 = Firmware command
1 = Hardware command

DATA — Data Flag – Shows that data accompanies the command.
0 = No data

1 = Data follows the command

R/W — Read/Write Flag

0 = Write

1 = Read

TTAGO — Trace, Tag, Go Field

Table 59 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-incriminated by 2 and the word pointed to by X is then read or written.

Table 60 REGN Decoding

REGN Value	Instruction
000	—
001	—
010	READ/WRITE NEXT
011	PC
100	D
101	X
110	Y
111	SP

SHIFTER

This 16-bit shift register contains data being received or transmitted via the serial interface. It is also used by the BDM firmware for temporary storage.

Read: all modes (but not normally accessed by users)

Write: all modes (but not normally accessed by users)

SHIFTER — BDM Shift Register – High Byte **$FF02**

	BIT 15	14	13	12	11	10	9	BIT 8
	S15	S14	S13	S12	S11	S10	S9	S8
RESET:	X	X	X	X	X	X	X	X

SHIFTER — BDM Shift Register – Low Byte **$FF03**

	BIT 7	6	5	4	3	2	1	BIT 0
	S7	S6	S5	S4	S3	S2	S1	S0
RESET:	X	X	X	X	X	X	X	X

ADDRESS

This 16-bit address register is temporary storage for BDM hardware and firmware commands.

Read: all modes (but not normally accessed by users)

Write: only by BDM hardware (state machine)

ADDRESS — BDM Address Register – High Byte **$FF04**

	BIT 15	14	13	12	11	10	9	BIT 8
	A15	A14	A13	A12	A11	A10	A9	A8
RESET:	X	X	X	X	X	X	X	X

ADDRESS — BDM Address Register – High Byte **$FF05**

	BIT 7	6	5	4	3	2	1	BIT 0
	A7	A6	A5	A4	A3	A2	A1	A0
RESET:	X	X	X	X	X	X	X	X

CCRSAV 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.

Read and write: all modes

CCRSAV— BDM CCR Holding Register **$FF06**

	BIT 7	6	5	4	3	2	1	BIT 0
	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
RESET: NOTE 1 (1)	X	X	X	X	X	X	X	X

1. Initialized to equal the CPU12 CCR register by the firmware.

Breakpoints

Hardware breakpoints are used to debug software on the MC68HC912DT128A 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 MC68HC912DT128A 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

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.

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 BKPM bit and the BKALE bit. The BKxRW and BKxRWE bits are ignored. In dual address mode the BKDBE becomes an enable for the second address breakpoint. The BKSZ8 bit will have no effect when in a dual address mode.

BDM Full Breakpoint Mode

A single full feature breakpoint which causes the part to enter background debug mode. BDM mode may be entered by a breakpoint only if an internal signal from the BDM indicates background debug mode is enabled.

- 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.

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, consistent across modes, by the BKPM bit, the BKALE bit, and the BKxRW and BKxRWE bits. In dual address mode the BKDBE becomes an enable for the second address breakpoint. The BKSZ8 bit 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.

- BKDBE will be used as an enable for the second address only breakpoint.
- 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.

Breakpoint 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 61 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 a break on a match (next instruction boundary) or on a match that will be an executable opcode. Data and non-executed 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 62 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 = The BRKDH/L registers are not used in any comparison

1 = The 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 63 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
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
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
RESET:	0	0	0	0	0	0	0	0

These bits are compared to the most significant byte of the data bus 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
RESET:	0	0	0	0	0	0	0	0

These bits are compared to the least significant byte of the data bus 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.

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. The TAGLO signal shares a pin with the LSTRB signal, and the TAGHI signal shares a pin with the BKGD signal. Tagging information is latched on the falling edge of ECLK.

Table 64 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 64 Tag Pin Function

TAGHI	TAGLO	Tag
1	1	no tag
1	0	low byte
0	1	high byte
0	0	both bytes

The tag follows program information as it advances through the queue. When a tagged instruction reaches the head of the queue, the CPU enters active background debugging mode rather than execute the instruction.

Electrical Characteristics

Contents

Introduction

The MC68HC912DT128A microcontroller unit (MCU) is a16-bit device composed of standard on-chip peripherals including a 16-bit central processing unit (CPU12), 128-Kbyte flash EEPROM, 8K byte RAM, 2K byte EEPROM, two asynchronous serial communications interfaces (SCI), a serial peripheral interface (SPI), an 8-channel, 16-bit timer, two16-bit pulse accumulators and 16-bit down counter (ECT), two 10-bit analog-to-digital converter (ADC), a four-channel pulse-width modulator (PWM), an IIC interface module, and three MSCAN modules. The chip is the first 16-bit microcontroller to include both byte-erasable EEPROM and flash EEPROM on the same device. System resource mapping, clock generation, interrupt control and bus interfacing are managed by the Lite integration module (LIM). The MC68HC912DT128A 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.

Tables of Data

Table 65 Maximum Ratings(1)

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}, V_{DDA}, V_{DDX}	–0.3 to +6.5	V
Input voltage	V_{IN}	–0.3 to +6.5	V
Operating temperature range	T_A	T_L to T_H 0 to +70 –40 to +85	°C
Storage temperature range	T_{stg}	–55 to +150	°C
Current drain per pin(2) Excluding V_{DD} and V_{SS}	I_{IN}	±25	mA
V_{DD} differential voltage	V_{DD}–V_{DDX}	6.5	V

1. Permanent damage can occur if maximum ratings are exceeded. Exposures to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. One pin at a time, observing maximum power dissipation limits. Internal circuitry protects the inputs against damage caused by high static voltages or electric fields; however, normal precautions are necessary to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Extended operation at the maximum ratings can adversely affect device reliability. Tying unused inputs to an appropriate logic voltage level (either GND or V_{DD}) enhances reliability of operation.

NOTE: *Four pins (VRL0, VRH0, VRL1, and VRH1) show ESD susceptibility below the Motorola standards. Care should be exercised in handling these pins.*

Table 66 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Average junction temperature	T_J	$T_A + (P_D \times \Theta_{JA})$	°C
Ambient temperature	T_A	User-determined	°C
Package thermal resistance (junction-to-ambient) 112-pin quad flat pack (QFP)	Θ_{JA}	40	°C/W
Total power dissipation(1)	P_D	$P_{INT} + P_{I/O}$ or $\frac{K}{T_J + 273°C}$	W
Device internal power dissipation	P_{INT}	$I_{DD} \times V_{DD}$	W
I/O pin power dissipation(2)	$P_{I/O}$	User-determined	W
A constant(3)	K	$P_D \times (T_A + 273°C) + \Theta_{JA} \times P_D^2$	W · °C

1. This is an approximate value, neglecting $P_{I/O}$.
2. For most applications $P_{I/O} \ll P_{INT}$ and can be neglected.
3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iteratively for any value of T_A.

Table 67 DC Electrical Characteristics

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
Input high voltage, all inputs	V_{IH}	$0.7 \times V_{DD}$	$V_{DD} + 0.3$	V
Input low voltage, all inputs	V_{IL}	V_{SS}–0.3	$0.2 \times V_{DD}$	V
Output high voltage, all I/O and output pins except XTAL Normal drive strength I_{OH} = –10.0 µA I_{OH} = –0.8 mA Reduced drive strength I_{OH} = –4.0 µA I_{OH} = –0.3 mA	V_{OH}	 $V_{DD} - 0.2$ $V_{DD} - 0.8$ $V_{DD} - 0.2$ $V_{DD} - 0.8$	 — — — —	 V V V V
Output low voltage, all I/O and output pins except XTAL Normal drive strength I_{OL} = 10.0 µA I_{OL} = 1.6 mA Reduced drive strength I_{OL} = 3.6 µA I_{OL} = 0.6 mA	V_{OL}	 — — — —	 V_{SS}+0.2 V_{SS}+0.4 V_{SS}+0.2 V_{SS}+0.4	 V V V V

Table 67 DC Electrical Characteristics

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
Input leakage current(1) $V_{in} = V_{DD}$ or V_{SS}All input only pins except $\overline{IRQ}$, ATD(2) and V_{FP} $V_{in} = V_{DD}$ or $V_{SS}\overline{IRQ}$	I_{in}	— —	±2.5 ±10	µA µA
Three-state leakage, I/O ports, BKGD, and $\overline{RESET}$	I_{OZ}	—	±2.5	µA
Input capacitance All input pins and ATD pins (non-sampling) ATD pins (sampling) All I/O pins	C_{in}	— — —	10 15 20	pF pF pF
Output load capacitance All outputs except PS[7:4] PS[7:4] when configured as SPI	C_L	— —	90 200	pF pF
Programmable active pull-up/pull-down current $\overline{IRQ}$, $\overline{XIRQ}$, $\overline{DBE}$, $\overline{LSTRB}$, R/$\overline{W}$. ports A, B, H, J, K, P, S, T MODA, MODB active pull down during reset BKGD passive pull up	I_{APU}	50 50 50	500 500 500	µA µA µA
RAM standby voltage, power down	V_{SB}	1.5	—	V
RAM standby current	I_{SB}	—	50	µA

1. Specification is for parts in the -40 to +85°C range. Higher temperature ranges will result in increased current leakage.
2. See ATD DC Electrical Characteristics.

Table 68 Supply Current

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	2 MHz	4 MHz	8 MHz	Unit
Maximum total supply current **RUN:** Single-chip mode Expanded mode	I_{DD}	 20 30	 30 50	 55 90	 mA mA
WAIT: (All peripheral functions shut down) Single-chip mode Expanded mode	W_{IDD}	 3 4	 5 6.5	 10 15	 mA mA
STOP: Single-chip mode, no clocks	S_{IDD}	 150	 150	 150	 µA

Table 69 ATD DC Electrical Characteristics

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD Clock = 2 MHz, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
Analog supply voltage	V_{DDA}	4.5	5.5	V
Analog supply currentNormal operation STOP	I_{DDA}		1.0 10	mA µA
Reference voltage, low	V_{RL}	V_{SSA}	$V_{DDA}/2$	V
Reference voltage, high	V_{RH}	$V_{DDA}/2$	V_{DDA}	V
V_{REF} differential reference voltage(1)	$V_{RH}–V_{RL}$	4.5	5.5	V
Input voltage(2)	V_{INDC}	V_{SSA}	V_{DDA}	V
Input current, off channel(3)	I_{OFF}		100	µA
Reference supply current	I_{REF}		250	µA
Input capacitanceNot Sampling Sampling	C_{INN} C_{INS}		10 15	pF pF

1. Accuracy is guaranteed at $V_{RH} - V_{RL}$ = 5.0V ±10%.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10°C decrease from maximum temperature.

Table 70 Analog Converter Characteristics (Operating)

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD Clock = 2 MHz, unless otherwise noted

Characteristic	Symbol	Min	Typical	Max	Unit
8-bit resolution(1)	1 count		20		mV
8-bit differential non-linearity(2)	DNL	–0.5		+0.5	count
8-bit integral non-linearity(2)	INL	–1		+1	count
8-bit absolute error(3) 2, 4, 8, and 16 ATD sample clocks	AE	–1		+1	count
10-bit resolution(1)	1 count		5		mV
10-bit differential non-linearity(2)	DNL	–2		2	count
10-bit integral non-linearity(2)	INL	–2		2	count
10-bit absolute error(3) 2, 4, 8, and 16 ATD sample clocks	AE	–2.5		2.5	count
Maximum source impedance	R_S		20	See note(4)	KΩ

1. $V_{RH} - V_{RL} \geq 5.12V$; $V_{DDA} - V_{SSA} = 5.12V$
2. At V_{REF} = 5.12V, one 8-bit count = 20 mV, and one 10-bit count = 5mV.
   INL and DNL are characterized using the process window parameters affecting the ATD accuracy, but they are not tested.
3. These values include quantization error which is inherently 1/2 count for any A/D converter.
4. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.
   Error from junction leakage is a function of external source impedance and input leakage current. Expected error in result value due to junction leakage is expressed in voltage (V_{ERRJ}):

$$V_{ERRJ} = R_S \times I_{OFF}$$

   where I_{OFF} is a function of operating temperature. Charge-sharing effects with internal capacitors are a function of ATD clock speed, the number of channels being scanned, and source impedance. For 8-bit conversions, charge pump leakage is computed as follows:

$$V_{ERRJ} = .25pF \times V_{DDA} \times R_S \times ATDCLK/(8 \times \text{number of channels})$$

Table 71 ATD AC Characteristics (Operating)

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD Clock = 2 MHz, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
ATD operating clock frequency	f_{ATDCLK}	0.5	2.0	MHz
Conversion time per channel 0.5 MHz ≤ f_{ATDCLK} ≤ 2 MHz 16 ATD clocks 30 ATD clocks	t_{CONV}	 8.0 15.0	 32.0 60.0	 µs µs
Stop and ATD power up recovery time(1) V_{DDA} = 5.0V	t_{SR}		10	µs

1. From the time ADPU is asserted until the time an ATD conversion can begin.

Table 72 ATD Maximum Ratings

Characteristic	Symbol	Value	Units
ATD reference voltage $V_{RH} \le V_{DDA}$ $V_{RL} \ge V_{SSA}$	 V_{RH} V_{RL}	 –0.3 to +6.5 –0.3 to +6.5	 V V
V_{SS} differential voltage	$\lvert V_{SS}–V_{SSA}\rvert$	0.1	V
V_{DD} differential voltage	$V_{DD}–V_{DDA}$ $V_{DDA}–V_{DD}$	6.5 0.3	V V
V_{REF} differential voltage	$\lvert V_{RH}–V_{RL}\rvert$	6.5	V
Reference to supply differential voltage	$\lvert V_{RH}–V_{DDA}\rvert$	6.5	V

Table 73 EEPROM Characteristics

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
Minimum programming clock frequency	f_{PROG}	250K		hz
Programming time	t_{PROG}		10	ms
Clock recovery time, following STOP, to continue programming	t_{CRSTOP}		t_{PROG}+ 1	ms
Erase time	t_{ERASE}		10	ms
Write/erase endurance		10,000		cycles
Data retention		10(1)		years
EEPROM Programming Maximum Time to ‘AUTO’ Bit Set	—	—	500	µs
EEPROM Erasing Maximum Time to ‘AUTO’ Bit Set	—	—	10	ms

1. Based on the average life time operating temperature of 70°C.

Table 74 Flash EEPROM Characteristics

V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	Min	Max	Units
Bytes per row	—	64	64	Bytes
Read bus clock frequency	f_{READ}	32K	8M	hz
Erase time	t_{ERAS}	8	—	ms
PGM/ERAS to HVEN set up time	t_{NVS}	10	—	µs
High-voltage hold time	t_{NVH}	5	—	µs
High-voltage hold time (erase)	t_{NVHL}	100	—	µs
Program hold time	t_{PGS}	5	—	µs
Program time	t_{FPGM}	30	40	µs
Return to read time	t_{RCV}	1	—	µs
Cumulative program hv period	t_{HV}	—	8	ms
Row program/erase endurance	—	100		cycles
Data retention	—	10(1)		years

1. Based on the average life time operating temperature of 70°C.

Table 75 Pulse Width Modulator Characteristics

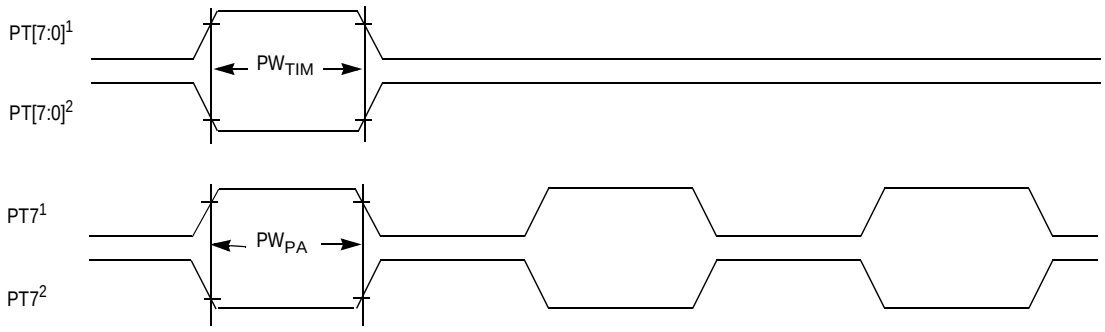
V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristic	Symbol	Min	Max	Unit
E-clock frequency	f_{eclk}	0.004	8.0	MHz
A-clock frequency Selectable	f_{aclk}	$f_{eclk}/128$	f_{eclk}	Hz
BCLK frequency Selectable	f_{bclk}	$f_{eclk}/128$	f_{eclk}	Hz
Left-aligned PWM frequency 8-bit 16-bit	f_{lpwm}	 $f_{eclk}/1M$ $f_{eclk}/256M$	 $f_{eclk}/2$ $f_{eclk}/2$	 Hz Hz
Left-aligned PWM resolution	r_{lpwm}	$f_{eclk}/4K$	f_{eclk}	Hz
Center-aligned PWM frequency 8-bit 16-bit	f_{cpwm}	 $f_{eclk}/2M$ $f_{eclk}/512M$	 f_{eclk} f_{eclk}	 Hz Hz
Center-aligned PWM resolution	r_{cpwm}	$f_{eclk}/4K$	f_{eclk}	Hz

Table 76 Control Timing

Characteristic	Symbol	8.0 MHz		Unit
		Min	Max	
Frequency of operation	f_o	0.004	8.0	MHz
E-clock period	t_{cyc}	0.125	250	µs
Crystal frequency	f_{XTAL}	0.5	16.0	MHz
External oscillator frequency	f_{eo}	0.5	16.0	MHz
Processor control setup time $t_{PCSU} = t_{cyc}/2+ 30$	t_{PCSU}	92	—	ns
Reset input pulse width To guarantee external reset vector Minimum input time (can be preempted by internal reset)	PW_{RSTL}	32 2	— —	t_{cyc} t_{cyc}
Mode programming setup time	t_{MPS}	4	—	t_{cyc}
Mode programming hold time	t_{MPH}	20	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode $PW_{IRQ} = 2t_{cyc} + 20$	PW_{IRQ}	270	—	ns
Wait recovery startup time	t_{WRS}	—	4	cycles
Timer input capture pulse width	PW_{TIM}	270	—	ns

1. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for 16 clock cycles, releases the pin, and samples the pin level 8 cycles later to determine the source of the interrupt.



NOTES:
1. Rising edge sensitive input
2. Falling edge sensitive input

Figure 62 Timer Inputs

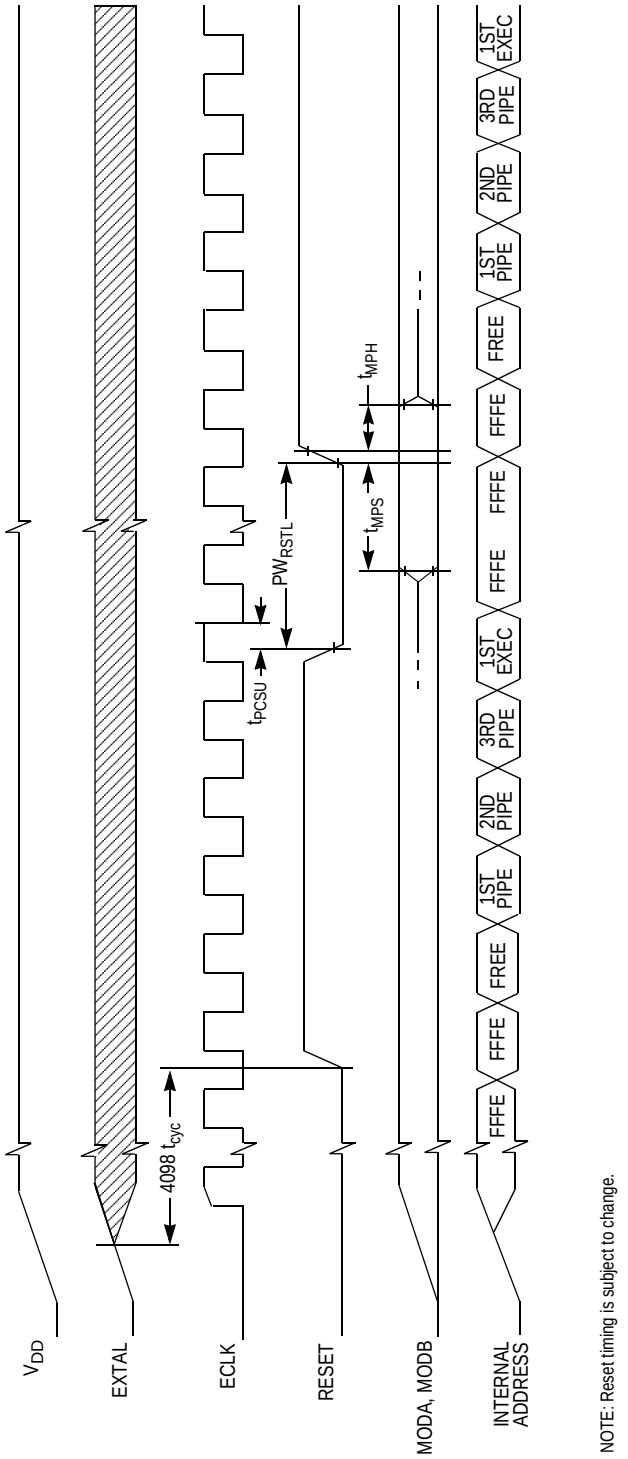


Figure 63 POR and External Reset Timing Diagram

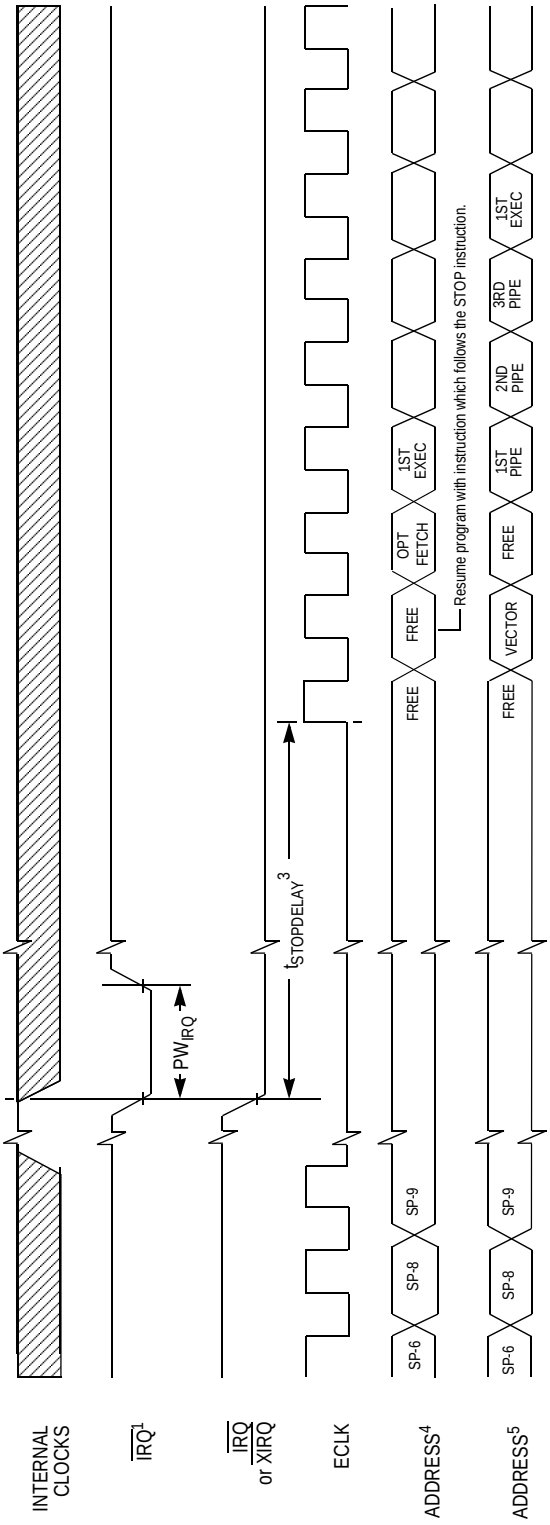


Figure 64 STOP Recovery Timing Diagram

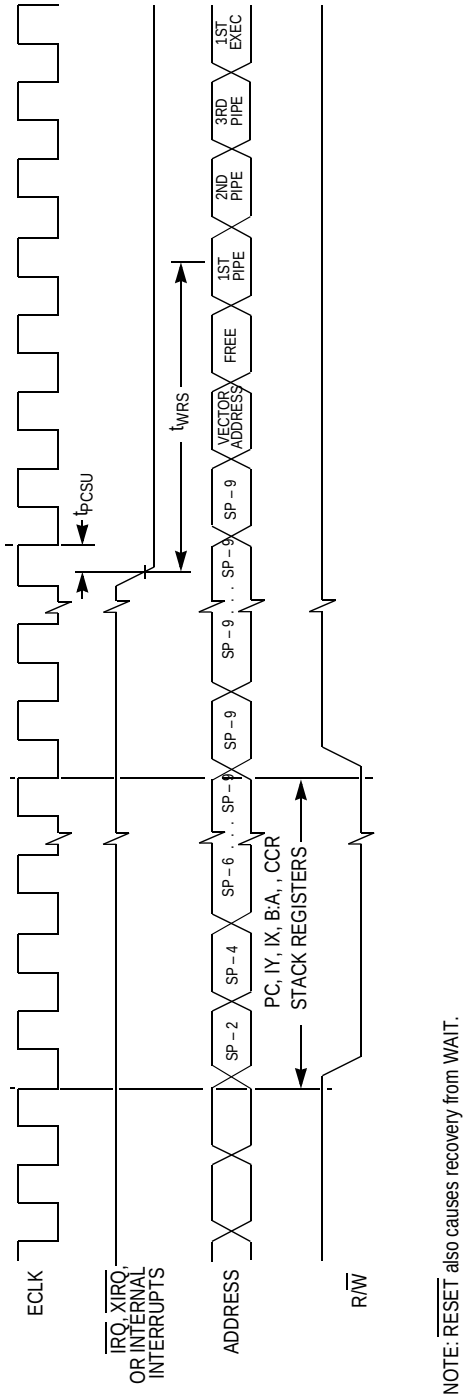


Figure 65 WAIT Recovery Timing Diagram

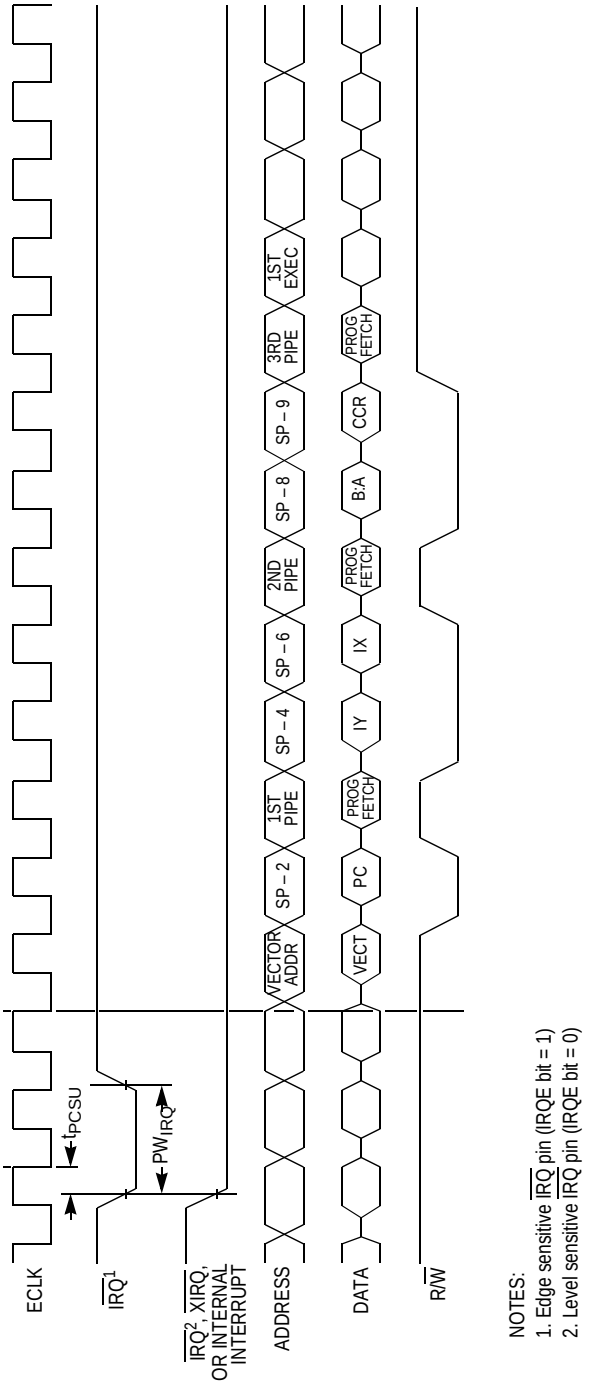


Figure 66 Interrupt Timing Diagram

Table 77 Peripheral Port Timing

Characteristic	Symbol	8.0 MHz		Unit
		Min	Max	
Frequency of operation (E-clock frequency)	f_o	0.004	8.0	MHz
E-clock period	t_{cyc}	0.125	250	µs
Peripheral data setup time MCU read of ports	t_{PDSU}	81	—	ns
Peripheral data hold time MCU read of ports	t_{PDH}	0	—	ns
Delay time, peripheral data write MCU write to ports except Port CAN	t_{PWD}	—	40	ns
Delay time, peripheral data write MCU write to Port CAN	t_{PWD}	—	71	ns

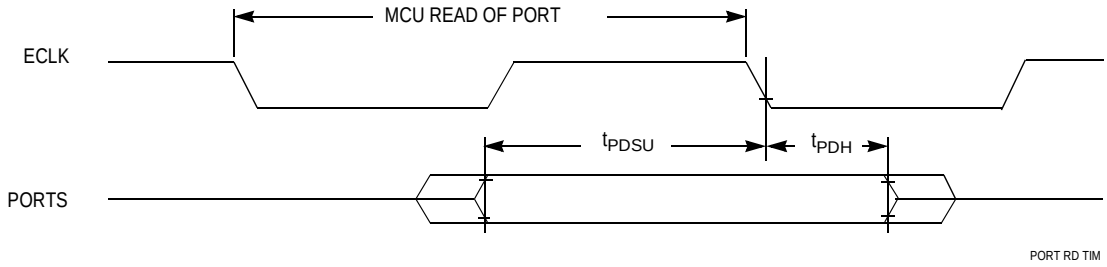


Figure 67 Port Read Timing Diagram

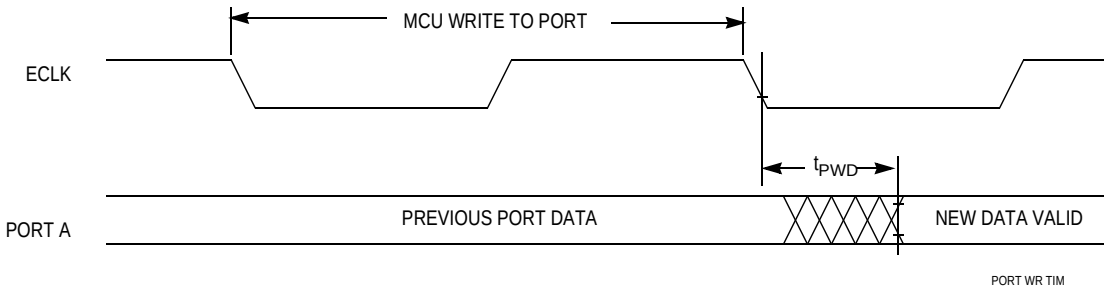


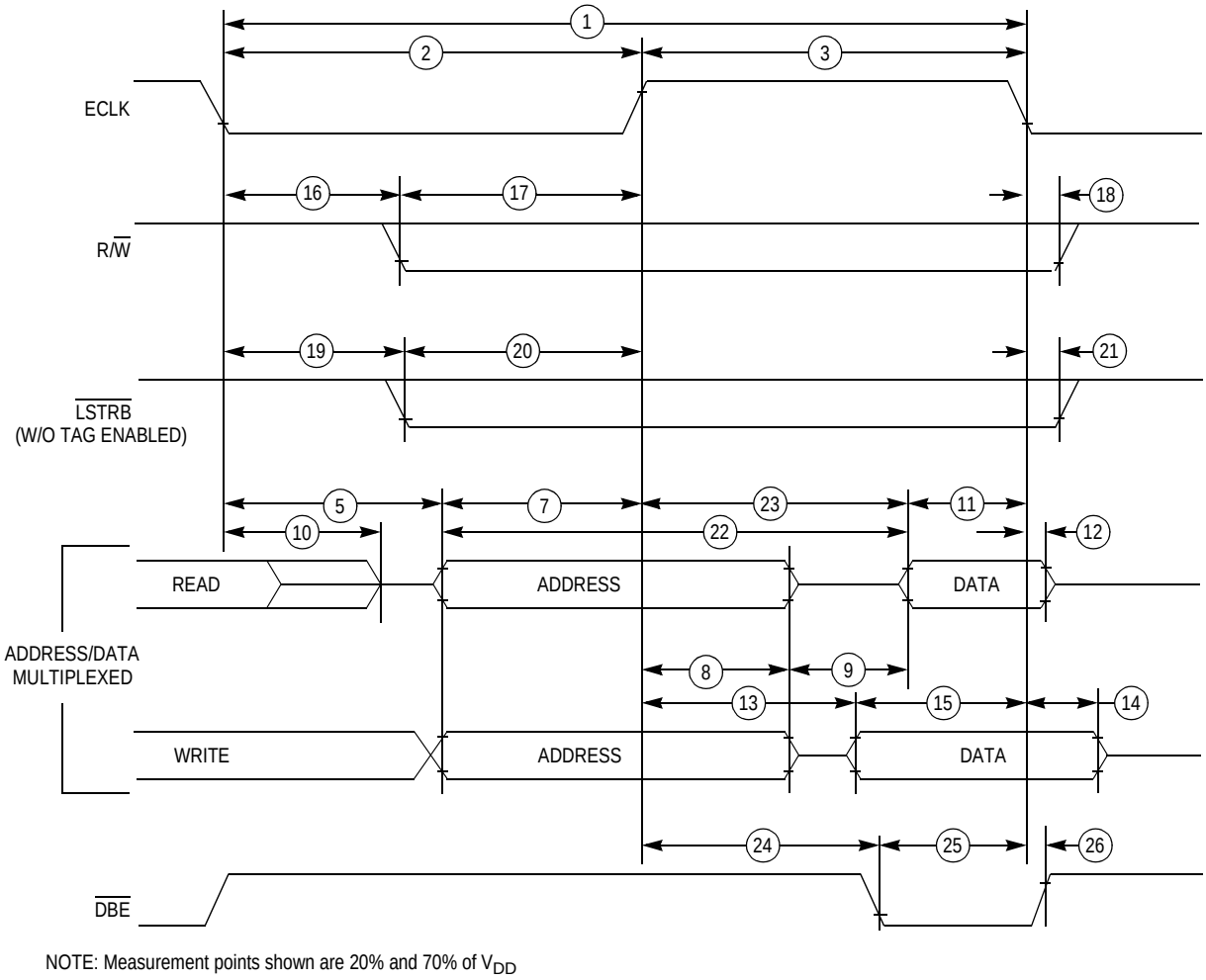
Figure 68 Port Write Timing Diagram

Table 78 Multiplexed Expansion Bus Timing

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Num	Characteristic[1], [2], [3], [4]	Delay	Symbol	8 MHz		Unit
				Min	Max	
	Frequency of operation (E-clock frequency)		f_o	0.004	8.0	MHz
1	Cycle time $t_{cyc} = 1/f_o$	—	t_{cyc}	0.125	250	µs
2	Pulse width, E low $PW_{EL} = t_{cyc}/2 + delay$	–2	PW_{EL}	60		ns
3	Pulse width, E high[5] $PW_{EH} = t_{cyc}/2 + delay$	–2	PW_{EH}	60		ns
5	Address delay time $t_{AD} = t_{cyc}/4 + delay$	14	t_{AD}		45	ns
7	Address valid time to ECLK rise $t_{AV} = PW_{EL} - t_{AD}$	—	t_{AV}	15		ns
8	Multiplexed address hold time $t_{MAH} = t_{cyc}/4 + delay$	–16	t_{MAH}	15		ns
9	Address Hold to Data Valid	—	t_{AHDS}	5		ns
10	Data Hold to High Z $t_{DHZ} = t_{AD} - 20$	—	t_{DHZ}		20	ns
11	Read data setup time	—	t_{DSR}	38		ns
12	Read data hold time	—	t_{DHR}	0		ns
13	Write data delay time $t_{DDW} = t_{cyc}/4 + delay$	14	t_{DDW}		45	ns
14	Write data hold time $t_{DHW} = t_{cyc}/4 + delay$	–11	t_{DHW}	20		ns
15	Write data setup time[5] $t_{DSW} = PW_{EH} - t_{DDW}$	—	t_{DSW}	15		ns
16	Read/write delay time $t_{RWD} = t_{cyc}/4 + delay$	19	t_{RWD}		50	ns
17	Read/write valid time to E rise $t_{RWV} = PW_{EL} - t_{RWD}$	—	t_{RWV}	10		ns
18	Read/write hold time $t_{RWH} = t_{cyc}/4 + delay$	–26	t_{RWH}	5		ns
19	Low strobe[6] delay time $t_{LSD} = t_{cyc}/4 + delay$	26	t_{LSD}		57	ns
20	Low strobe[6] valid time to E rise $t_{LSV} = PW_{EL} - t_{LSD}$	—	t_{LSV}	3		ns
21	Low strobe[6] hold time $t_{LSH} = t_{cyc}/4 + delay$	–26	t_{LSH}	5		ns
22	Address access time[5] $t_{ACCA} = t_{cyc} - t_{AD} - t_{DSR}$	—	t_{ACCA}		26	ns
23	Access time from E rise[5] $t_{ACCE} = PW_{EH} - t_{DSR}$	—	t_{ACCE}		22	ns
24	$\overline{DBE}$ delay from ECLK rise[5] $t_{DBED} = t_{cyc}/4 + delay$	20	t_{DBED}		51	ns
25	$\overline{DBE}$ valid time $t_{DBE} = PW_{EH} - t_{DBED}$	—	t_{DBE}	9		ns
26	$\overline{DBE}$ hold time from ECLK fall		t_{DBEH}	0	10	ns

1. All timings are calculated for normal port drives.
2. Crystal input is required to be within 45% to 55% duty.
3. Reduced drive must be off to meet these timings.
4. Unequalled loading of pins will affect relative timing numbers.
5. This characteristic is affected by clock stretch.
   Add $N \times t_{cyc}$ where N = 0, 1, 2, or 3, depending on the number of clock stretches.
6. Without TAG enabled.



NOTE: Measurement points shown are 20% and 70% of V_{DD}

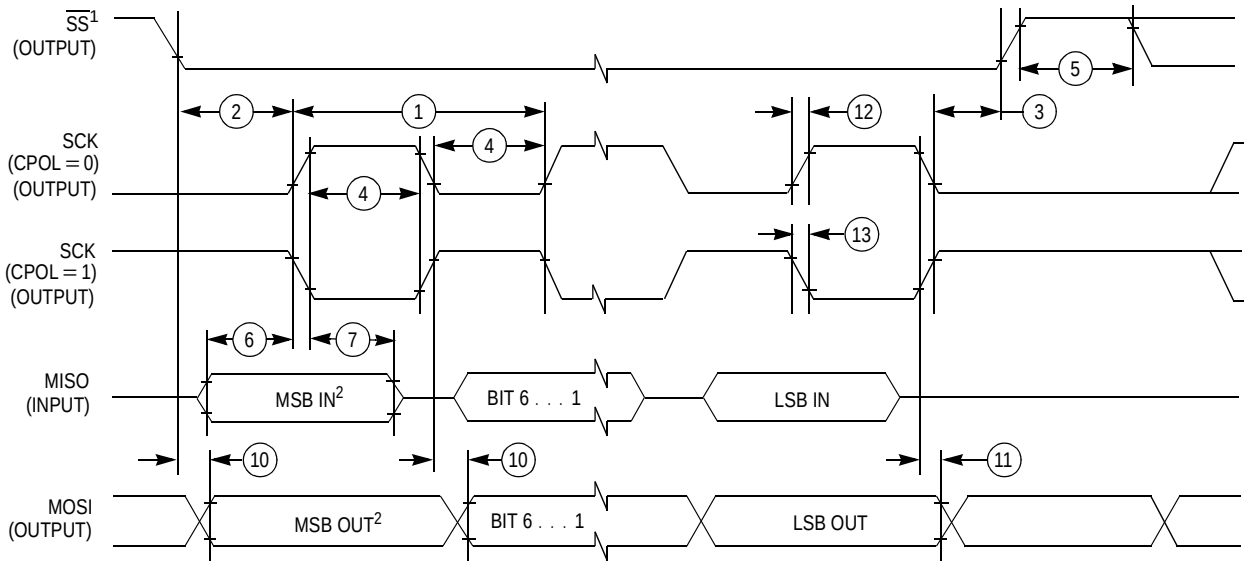
Figure 69 Multiplexed Expansion Bus Timing Diagram

Table 79 SPI Timing

(V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H , 200 pF load on all SPI pins)[1]

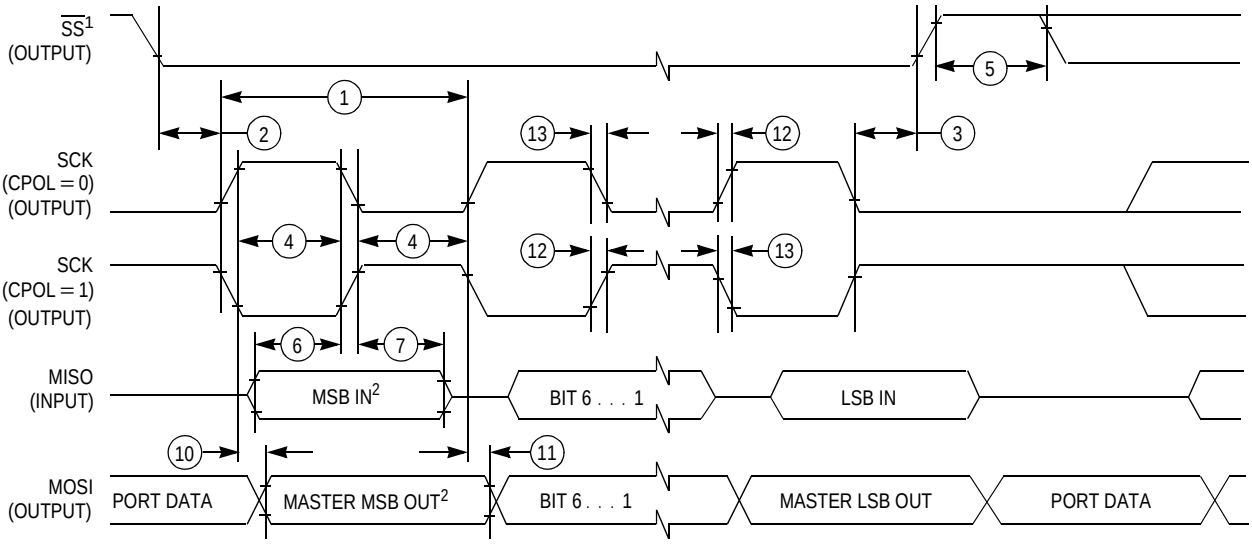
Num	Function	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	f_{op}	 1/256 1/256	 1/2 1/2	f_{eclk}
	SCK Period Master Slave	t_{sck}	 2 2	 256 —	 t_{cyc} t_{cyc}
	Enable Lead Time Master Slave	t_{lead}	 1/2 1	 — —	 t_{sck} t_{cyc}
	Enable Lag Time Master Slave	t_{lag}	 1/2 1	 — —	 t_{sck} t_{cyc}
	Clock (SCK) High or Low Time Master Slave	t_{wsck}	 t_{cyc} – 60 t_{cyc} – 30	 128 t_{cyc} —	 ns ns
	Sequential Transfer Delay Master Slave	t_{td}	 1/2 1	 — —	 t_{sck} t_{cyc}
	Data Setup Time (Inputs) Master Slave	t_{su}	 30 30	 — —	 ns ns
	Data Hold Time (Inputs) Master Slave	t_{hi}	 0 30	 — —	 ns ns
	Slave Access Time	t_a	—	1	t_{cyc}
	Slave MISO Disable Time	t_{dis}	—	1	t_{cyc}
	Data Valid (after SCK Edge) Master Slave	t_v	 — —	 50 50	 ns ns
	Data Hold Time (Outputs) Master Slave	t_{ho}	 0 0	 — —	 ns ns
	Rise Time Input Output	 t_{ri} t_{ro}	 — —	 t_{cyc} – 30 30	 ns ns
	Fall Time Input Output	 t_{fi} t_{fo}	 — —	 t_{cyc} – 30 30	 ns ns

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.



1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

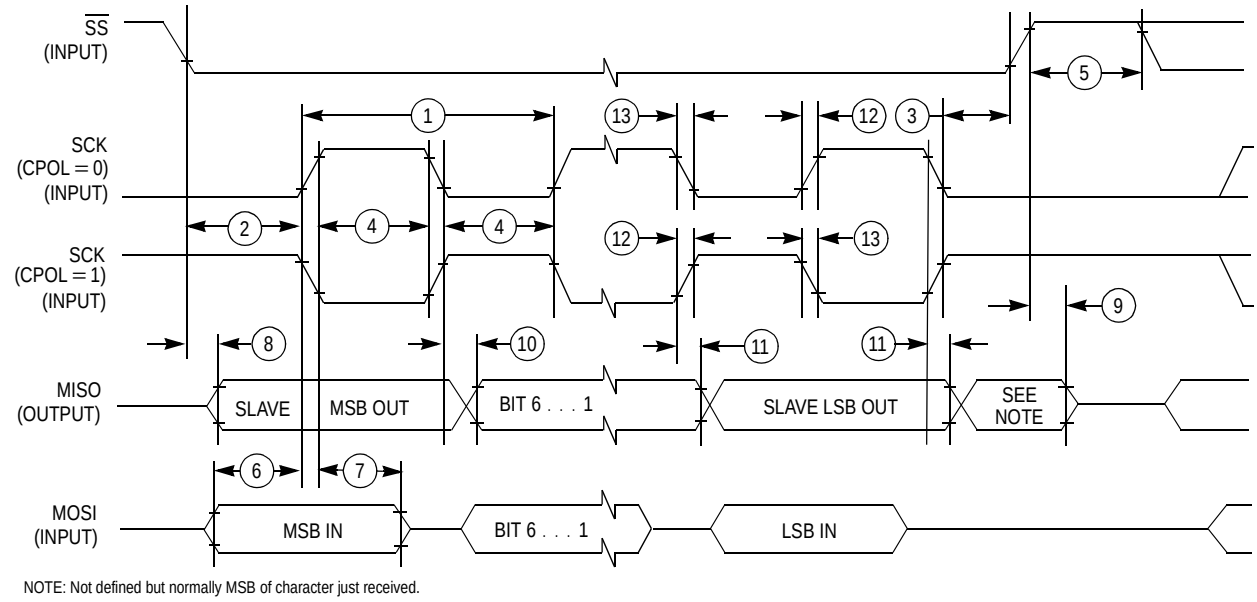
A) SPI Master Timing (CPHA = 0)



1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

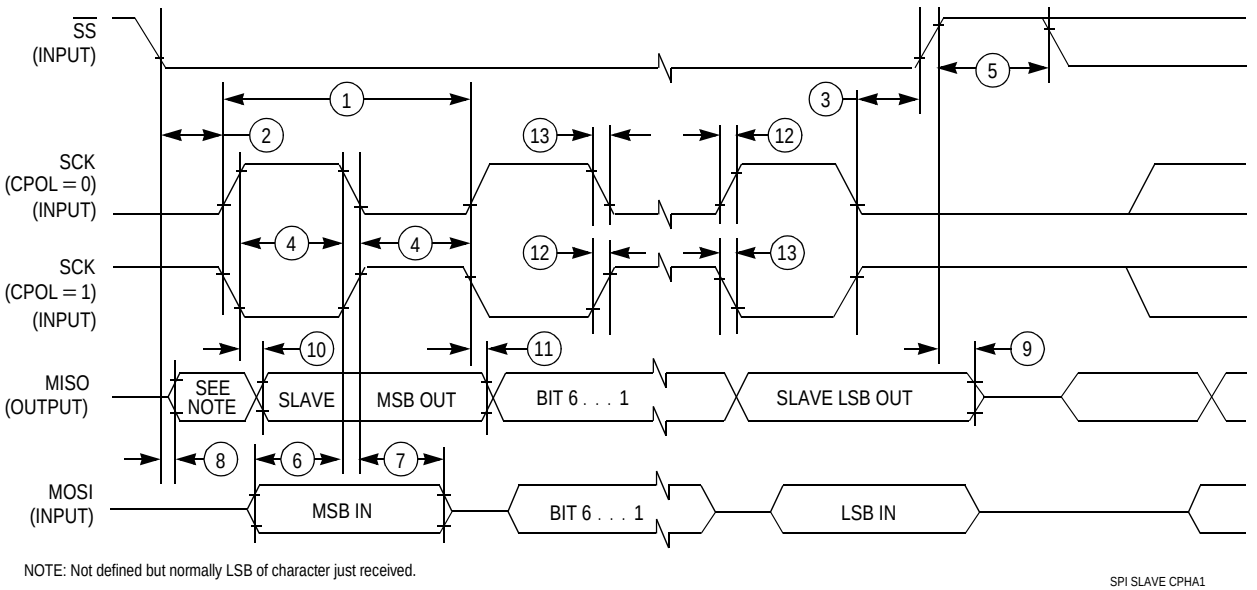
B) SPI Master Timing (CPHA = 1)

Figure 70 SPI Timing Diagram (1 of 2)



NOTE: Not defined but normally MSB of character just received.

A) SPI Slave Timing (CPHA = 0)



NOTE: Not defined but normally LSB of character just received.

SPI SLAVE CPHA1

B) SPI Slave Timing (CPHA = 1)

Figure 71 SPI Timing Diagram (2 of 2)

Table 80 CGM Characteristics

5.0 Volts +/- 10%

Characteristic	Symbol	Min.	Max.	Unit
PLL reference frequency, crystal oscillator range(1)	f_{REF}	0.5	8	MHz
Bus frequency	f_{BUS}	0.004	8	MHz
VCO range	f_{VCO}	2.5	8	MHz
VCO Limp-Home frequency	f_{VCOMIN}	0.5	2.5	MHz
Lock Detector transition from Acquisition to Tracking mode	Δ_{trk}	3%	4%	—
Lock Detection	Δ_{Lock}	0%	1.5%	—
Un-Lock Detection	Δ_{unl}	0.5%	2.5%	—
Lock Detector transition from Tracking to Acquisition mode	Δ_{unt}	6%	8%	—
PLLON Stabilization delay(2)				
PLLON Total Stabilization delay(3)	t_{stab}		3	ms
PLLON Acquisition mode stabilization delay(3)	t_{acq}		1	ms
PLLON Tracking mode stabilization delay(3)	t_{al}		2	ms

1. VDDPLL at VDD level.

2. PLL stabilization delay is highly dependent on operational requirement and external component values (e.e. crystal, XFC filter component values). Note **(3)** shows typical delay values for a typical configuration. Appropriate XFC filter values should be chosen based on operational requirement of system.

3. f_{REF} = 4MHz, f_{SYS} = 8MHz (REFDV = #$00, SYNR = #$01), XFC: Cs = 33nF, Cp - 3.3nF, Rs = 2.7KΩ.

Table 81 STOP Key Wake-up Filter

Characteristic	Symbol	Min.	Max.	Unit
STOP Key Wake-Up Filter time	t_{KWSTP}	2	10	µs
STOP Key Wake-Up Filter pulse interval	t_{KWSP}	20	-	µs

Appendix A: MC68HC912DT128A

Contents

Significant changes from the MC68HC912DG128 (non-A suffix device)

1. Flash

1a. Flash Architecture

The flash arrays are made from a new non-volatile memory (NVM) technology. An external VFP is no longer used. Programming is now carried out on a whole row (64 bytes) at a time. Erasing is still a bulk erase of the entire array.

1b. Flash Control Register

The Flash Control Register (FEECTL) is in the same location. However, the individual bit functions have changed significantly to support the new technology.

1c. Flash Programming Procedure

Programming of the flash is greatly simplified over previous HC12s. The read / verify / re-pulse programming algorithm is replaced by a much simpler method.

1d. Flash Programming Time

The most significant change resulting from the new flash technology is that the bulk erase and program times are now fixed. The erase time is at least twice as fast while the word programming time is at least 20% faster.

1e. Flash External Programming Voltage

The new flash does not require an external high voltage supply. All voltages required for programming and erase are now generated internally. Pin 97 that was the VFP pin on the 68HC912DG128 is now a factory TEST pin. On ealy production devices it is recommended that this pin is not connected within the application, but it may be connected to VSS or 5.5V max without issue. On later production parts this pin is not bonded out.

2. EEPROM

2a. EEPROM Architecture

Like the flash, the EEPROM is also made from this new NVM technology. The architecture and basic programming and erase operations are unchanged. However, there is a new optional programming method that allows faster programming of the EEPROM.

3b. EEPROM Clock Source and Pre-scaler

The first major difference on the new EEPROM is that it requires a constant time base source to ensure secure programming and erase operations. The clock source that is going to drive the clock divider input is the external clock input, EXTALi. The divide ratio from this source has to be set by programming an 10-bit time base pre-scalar into bits spread over two new registers, EEDIVH and EEDIVL.

The EEDIVH and EEDIVL registers are volatile. However, they are loaded upon reset by the contents of the non-volatile SHADOW word much in the same way as the EEPROM module control register (EEMCR) bits interact with the SHADOW word for configuration control on the existing revision.

2c. EEPROM AUTO programming & erasing

The second major change to the EEPROM is the inclusion in the EEPROM control register (EEPROG) of an AUTO function using the previously unused bit 5 of this register.

The AUTO function enables the logic of the MCU to automatically use the optimum programming or erasing time for the EEPROM. If using AUTO, the user does not need to wait for the normal minimum specified programming or erasing time. After setting the EEPGM bit as normal the user just has to poll that bit again, waiting for the MCU to clear it indicating that programming or erasing is complete.

2d. EEPROM Selective Write More Zeros

For some applications it may be advantageous to track more than 10k events with a single byte of EEPROM by programming one bit at a time. For that purpose, a special selective bit programming technique is available.

When this technique is utilized, a program / erase cycle is defined as multiple writes (up to eight) to a unique location followed by a single erase sequence.

3. STOP mode

This new version will correctly exit STOP mode without having to synchronize the start of STOP with the RTI clock.

4. WAIT mode

This new version will correctly exit WAIT mode using short XIRQ or IRQ inputs.

5. KWU Filter

The KWU filter will now ignore pulses shorter than 2 microseconds.

6. Port ADx

Power must be applied to VDDA at all times even if the ADC is not being used. This is necessary for port AD0 and port AD1 to function correctly as digital inputs.

7. ATD

Bit CC of ATDxCTL5 is not masked when bit S8CM = 1.

Appendix A: MC68HC912DT128A

Appendix B: CGM Practical Aspects

Contents

Introduction

This sections provides useful and practical pieces of information concerning the implementation of the CGM module.

A Few Hints For The CGM Crystal Oscillator Application

What Loading Capacitors To Choose?

First, from small-signal analysis, it is known that relatively large values for C1 and C2 have a positive impact on the phase margin. However, the higher loading they represent decreases the loop gain. Alternatively, small values for these capacitors will lead to higher open loop gain, but as the frequency of oscillation approaches the parallel resonance, the phase margin, and consequently the ability to start-up correctly, will decrease. From this it is clear that relatively large capacitor values (>33pF), are reserved for low frequency crystals in the MHz range.

NOTE: *Using the recommended loading capacitor CL value from the crystal manufacturer is a good starting point. Taking into account unavoidable strays, this equates to about (CL-2pF).*

Theoretically speaking, nothing precludes the use of non-identical values for C1 and C2. As this complicate a bit the management of the final board device list, this is not recommended. However, if asymmetrical capacitors are chosen, the value of C1 should be higher than C2 (because the reflected loading is proportional to the square of the impedance of C2).

DC Bias

Due to the nature of the translated ground Colpitts oscillator a DC voltage bias is applied to the crystal.

Please contact the crystal manufacturer for specific DC bias conditions and recommended capacitance value (if applicable).

What Is the Final Oscillation Frequency?

The exact calculation is not straightforward as it takes into account the resonator characteristics and the loading capacitors values as well as internal design parameters which can vary with Process Voltage Temperature (PVT) conditions. Nevertheless, if L is the series inductance, R is the series resistance, C is the series capacitance and Cc the parallel capacitance of the crystal, we can then use the following simplified equation:

$$Fosc = \frac{1}{2\pi} \cdot \sqrt{\frac{1}{LC} + \frac{1}{L \cdot (Cc + C1 \parallel C2)}}$$

C1=C2=Cl yields to

$$Fosc = \frac{1}{2\pi} \cdot \sqrt{\frac{1}{LC} + \frac{1}{L \cdot (Cc + Cl/2)}}$$

How Do I Control The Peak to Peak Oscillation Amplitude?

The CGM oscillator is equipped with an Amplitude Limitation Control loop which integrates the peak to peak ‘extal’ amplitude and in return reduces the steady current of the transconductor device until a stable quiescent point is reached. Controlling this final peak to peak amplitude can be performed by three means:

1. Reducing the values of C1 and C2. This decreases the loading so that the necessary gm value required to sustain oscillation can be

smaller. The consequently smaller current will be reached with a larger ‘extal’ swing.

2. Using VDDPLL=VSS (i.e. shutting off the PLL). Doing so increases the starting current by approximately 50%. All other parameters staying the same, a larger ‘extal’ swing will be required to reduce this starting current to its quiescent value.
3. Also, placing a high value resistor (>1MΩ) across the EXTAL and XTAL pins significantly increased the oscillation amplitude. Because this complicates the design analysis as it transforms a pure susceptance jωC1 into a complex admittance G+jωC1, Motorola cannot promote this application trick.

What Do I Do In Case The Oscillator Does Not Start-up?

1. First, verify that the application schematic respects the principle of operation, i.e. crystal mounted between EXTAL and VSS, Capacitor C1 between XTAL and EXTAL, Capacitor C2 between XTAL and VSS, nothing else. This is not the conventional MCU application schematic of the Pierce oscillator as it can be seen on other HC12 derivatives!
2. Re-consider the choice of the tuning capacitors.
3. If the quartz or resonator is of a high frequency type (e.g. above 10MHz), consider using VDDPLL=0V. Obviously, this choice precludes the use of the PLL. However, in this case the oscillator circuitry has more quiescent current available and the starting transconductance is thus significantly higher.
4. The oscillator circuitry is powered internally from a core VDD pad and the return path is the VSSPLL pad. Verify on the application PCB the correct connection of these pads (especially VSSPLL), but also verify the waveform of the VDD voltage as it is imposed on the pad. Sometimes external components (for instance choke inductors), can cause oscillations on the power line. This is of course detrimental to the oscillator circuitry.
5. If possible, consider using a resonator with built-in tuning capacitors. They may offer better performances with respect to their discrete elements implementation.

Practical Aspects For The PLL Usage

Synthesized Bus Frequency

Starting from a ceramic resonator or quartz crystal frequency F_{XTAL}, if 'refdv' and 'synr' are the decimal content of the REFDV and SYNR registers respectively, then the MCU bus frequency will simply be:

$$F_{BUS} = F_{VCO} = \frac{F_{XTAL} \cdot (synr + 1)}{(refdv + 1)}$$

$$synr \in \{0,1,2,3...63\}$$

$$refdv \in \{0,1,2,3...7\}$$

NOTE: *It is not allowed to synthesize a bus frequency that is lower than the crystal frequency, as the correct functioning of some internal synchronizers would be jeopardized (e.g. the MCLK and XCLK clock generators).*

Operation Under Adverse Environmental Conditions

The normal operation for the PLL is the so-called 'automatic bandwidth selection mode' which is obtained by having the AUTO bit set in the PLLCR register. When this mode is selected and as the VCO frequency approaches its target, the charge pump current level will automatically switch from a relatively high value of around 40 µA to a lower value of about 3 µA. It can happen that this low level of charge pump current is not enough to overcome leakages present at the XFC pin due to adverse environmental conditions. These conditions are frequently encountered for uncoated PCBs in automotive applications. The main symptom for this failure is an unstable characteristic of the PLL which in fact 'hunts' between acquisition and tracking modes. It is then advised for the running software to place the PLL in manual, forced acquisition mode by clearing both the AUTO and the $\overline{ACQ}$ bits in the PLLCR register. Doing so will maintain the high current level in the charge pump constantly and will permit to sustain higher levels of leakages at the XFC pin. This latest revision of the Clock Generator Module maintains the lock detection feature even in manual bandwidth control, offering then to the application software the same flexibility for the clocking control as the automatic mode.

Filter Components Selection Guide

Equations Set

These equations can be used to select a set of filter components. Two cases are considered:

1. The ‘tracking’ mode. This situation is reached normally when the PLL operates in automatic bandwidth selection mode (AUTO=1 in the PLLCR register).
2. The ‘acquisition’ mode. This situation is reached when the PLL operates in manual bandwidth selection mode and forced acquisition (AUTO=0, $\overline{ACQ}$=0 in the PLLCR register).

In both equations, the power supply should be 5V. Start with the target loop bandwidth as a function of the other parameters, but obviously, nothing prevents the user from starting with the capacitor value for example. Also, remember that the smoothing capacitor is always assumed to be one tenth of the series capacitance value.

So with:

m: the multiplying factor for the reference frequency (i.e. (synr+1))
R: the series resistance of the low pass filter in Ω
C: the series capacitance of the low pass filter in nF
F_{bus}: the target bus frequency expressed in MHz
ζ: the desired damping factor
F_c: the desired loop bandwidth expressed in Hz

for the ‘tracking’ mode:

$$F_c = \frac{2 \cdot 10^9 \cdot \zeta^2}{\pi \cdot R \cdot C} = \frac{37.78 \cdot e^{\left(\frac{1.675 - F_{bus}}{10.795}\right)} \cdot R}{2 \cdot \pi \cdot m}$$

and for the ‘acquisition’ mode:

$$F_c = \frac{2 \cdot 10^9 \cdot \zeta^2}{\pi \cdot R \cdot C} = \frac{415.61 \cdot e^{\left(\frac{1.675 - F_{bus}}{10.795}\right)} \cdot R}{2 \cdot \pi \cdot m}$$

Particular Case of an 8MHz Synthesis

Assume that a desired value for the damping factor of the second order system is close to 0.9 as this leads to a satisfactory transient response. Then, derived from the equations above, **Table 82** and **Table 83** suggest sets of values corresponding to several loop bandwidth possibilities in the case of an 8MHz synthesis for the two cases mentioned above.

The filter components values are chosen from standard series (e.g. E12 for resistors). The operating voltage is assumed to be 5V (although there is only a minor difference between 3V and 5V operation). The smoothing capacitor Cp in parallel with R and C is set to be 1/10 of the value of C. The reference frequencies mentioned in this table correspond to the output of the fine granularity divider controlled by the REFDV register. This means that the calculations are irrespective of the way the reference frequency is generated (directly for the crystal oscillator or after division). The target frequency value also has an influence on the calculations of the filter components because the VCO gain is NOT constant over its operating range.

The bandwidth limit corresponds to the so-called Gardner's criteria. It corresponds to the maximum value that can be chosen before the continuous time approximation ceases to be justified. It is of course advisable to stay far away from this limit.

Table 82 Suggested 8MHz Synthesis PLL Filter Elements (Tracking Mode)

Reference [MHz]	SYNR	Fbus [MHz]	C [nF]	R [kΩ]	Loop Bandwidth [kHz]	Bandwidth Limit [kHz]
0.614	$0C	7.98	100	4.3	1.1	157
0.614	$0C	7.98	4.7	20	5.3	157
0.614	$0C	7.98	1	43	11.5	157
0.614	$0C	7.98	0.33	75	20	157
0.8	$09	8.00	220	2.7	0.9	201
0.8	$09	8.00	10	12	4.2	201
0.8	$09	8.00	2.2	27	8.6	201
0.8	$09	8.00	0.47	56	19.2	201
1	$07	8.00	220	2.4	1	251
1	$07	8.00	10	11	4.7	251
1	$07	8.00	2.2	24	9.9	251
1	$07	8.00	0.47	51	21.4	251
1.6	$05	8.00	330	1.5	1	402
1.6	$05	8.00	10	9.1	5.9	402
1.6	$05	8.00	3.3	15	10.2	402
1.6	$05	8.00	1	27	18.6	402
2	$03	8.00	470	1.1	0.96	502
2	$03	8.00	22	5.1	4.4	502
2	$03	8.00	4.7	11	9.6	502
2	$03	8.00	1	24	20.8	502
2.66	$02	8.00	220	1.5	1.6	668
2.66	$02	8.00	22	4.7	5.1	668
2.66	$02	8.00	4.7	10	11	668
2.66	$02	8.00	1	22	24	668
4	$01	8.00	220	1.2	1.98	1005
4	$01	8.00	33	3	5.1	1005
4	$01	8.00	10	5.6	9.3	1005
4	$01	8.00	2.2	12	19.8	1005

Table 83 Suggested 8MHz Synthesis PLL Filter Elements (Acquisition Mode)

Reference [MHz]	SYNR	Fbus [MHz]	C [nF]	R [kΩ]	Loop Bandwidth [kHz]	Bandwidth Limit [kHz]
0.614	$0C	7.98	1000	0.43	1.2	157
0.614	$0C	7.98	47	2	5.5	157
0.614	$0C	7.98	10	4.3	12	157
0.614	$0C	7.98	3.3	7.5	21	157
0.8	$09	8.00	2200	0.27	0.9	201
0.8	$09	8.00	100	1.2	4.4	201
0.8	$09	8.00	22	2.4	9.3	201
0.8	$09	8.00	4.7	5.6	20.1	201
1	$07	8.00	2200	0.22	1	251
1	$07	8.00	100	1	4.8	251
1	$07	8.00	2.	2.2	10.4	251
1	$07	8.00	4.7	4.7	22.5	251
1.6	$05	8.00	3300	0.15	1.1	402
1.6	$05	8.00	100	0.82	6.2	402
1.6	$05	8.00	33	1.5	10.7	402
1.6	$05	8.00	10	2.7	19.5	402
2	$03	8.00	4700	0.1	1	502
2	$03	8.00	220	0.51	4.6	502
2	$03	8.00	47	1	10	502
2	$03	8.00	10	2.4	21.8	502
2.66	$02	8.00	2200	0.12	1.7	668
2.66	$02	8.00	220	0.43	5.3	668
2.66	$02	8.00	47	1	11.6	668
2.66	$02	8.00	10	2	25.1	668
4	$01	8.00	2200	0.1	2.1	1005
4	$01	8.00	330	0.27	5.4	1005
4	$01	8.00	100	0.51	9.7	1005
4	$01	8.00	22	1	20.8	1005

Printed Circuit Board Guidelines

Printed Circuit Boards (PCBs) are the board of choice for volume applications. If designed correctly, a very low noise system can be built on a PCB with consequently good EMI/EMC performances. If designed incorrectly, PCBs can be extremely noisy and sensitive modules, and the CGM could be disrupted. Some common sense rules can be used to prevent such problems.

- Use a ‘star’ style power routing plan as opposed to a ‘daisy chain’. Route power and ground from a central location to each chip individually, and use the widest trace practical (the more the chip draws current, the wider the trace). NEVER place the MCU at the end of a long string of serially connected chips.
- When using PCB layout software, first direct the routing of the power supply lines as well as the CGM wires (crystal oscillator and PLL). Layout constraints must be then reported on the other signals and not on these ‘hot’ nodes. Optimizing the ‘hot’ nodes at the end of the routing process usually gives bad results.
- Avoid notches in power traces. These notches not only add resistance (and are not usually accounted for in simulations), but they can also add unnecessary transmission line effects.
- Avoid ground and power loops. This has been one of the most violated guidelines of PCB layout. Loops are excellent noise transmitters and can be easily avoided. When using multiple layer PCBs, the power and ground plane concept works well but only when strictly adhered to (do not compromise the ground plane by cutting a hole in it and running signals on the ground plane layer). Keep the spacing around via holes to a minimum (but not so small as to add capacitive effects).
- Be aware of the three dimensional capacitive effects of multi-layered PCBs.
- Bypass (decouple) the power supplies of all chips as close to the chip as possible. Use one decoupling capacitor per power supply pair (VDD/VSS, VDDX/VSSX...). Two capacitors with a ratio of about 100 sometimes offer better performances over a broader

spectrum. This is especially the case for the power supply pins close to the E port, when the E clock and/or the calibration clock are used.

- On the general VDD power supply input, a ‘T’ low pass filter LCL can be used (e.g. 10µH-47µF-10µH). The ‘T’ is preferable to the ‘Π’ version as the exhibited impedance is more constant with respect to the VDD current. Like many modular micro controllers, HC12 devices have a power consumption which not only varies with clock edges but also with the functioning modes.
- Keep high speed clock and bus trace length to a minimum. The higher the clock speed, the shorter the trace length. If noisy signals are sent over long tracks, impedance adjustments should be considered at both ends of the line (generally, simple resistors suffice).
- Bus drivers like the CAN physical interface should be installed close to their connector, with dedicated filtering on their power supply.
- Mount components as close to the board as possible. Snip excess lead length as close to the board as possible. Preferably use Surface Mount Devices (SMDs).
- Mount discrete components as close to the chip that uses them as possible.
- Do not cross sensitive signals ON ANY LAYER. If a sensitive signal must be crossed by another signal, do it as many layers away as possible and at right angles.
- Always keep PCBs clean. Solder flux, oils from fingerprints, humidity and general dirt can conduct electricity. Sensitive circuits can easily be disrupted by small amounts of leakage.
- Choose PCB coatings with care. Certain epoxies, paints, gelatins, plastics and waxes can conduct electricity. If the manufacturer cannot provide the electrical characteristics of the substance, do not use it.

In addition to the above general pieces of advice, the following guidelines should be followed for the CGM pins (but also more generally for any sensitive analog circuitry):

- Mount the PLL filter and oscillator components as close to the MCU as possible.
- Do not allow the EXTAL and XTAL signals to interfere with the XFC node. Keep these tracks as short as possible.
- Do not cross the CGM signals with any other signal on any level.
- Remember that the reference voltage for the XFC filter is VDDPLL.
- As the return path for the oscillator circuitry is VSSPLL, it is extremely important to CONNECT VSSPLL to VSS even if the PLL is not to be powered.Surface mount components reduce the susceptibility of signal contamination.
- Ceramic resonators with built-in capacitors are available. This is an interesting solution because the parasitic components involved are minimized due to the close proximity of the resonating elements.

Appendix B: CGM Practical Aspects

Glossary

A — See “accumulators (A and B or D).”

accumulators (A and B or D) — Two 8-bit (A and B) or one 16-bit (D) general-purpose registers in the CPU. The CPU uses the accumulators to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation with large loop bandwidth. Also see ’tracking mode’.

address bus — The set of wires that the CPU or DMA uses to read and write memory locations.

addressing mode — The way that the CPU determines the operand address for an instruction. The M68HC12 CPU has 15 addressing modes.

ALU — See “arithmetic logic unit (ALU).”

analogue-to-digital converter (ATD) — The ATD module is an 8-channel, multiplexed-input successive-approximation analog-to-digital converter.

arithmetic logic unit (ALU) — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

ATD — See “analogue-to-digital converter”.

Glossary

B — See “accumulators (A and B or D).”

baud rate — The total number of bits transmitted per unit of time.

BCD — See “binary-coded decimal (BCD).”

binary — Relating to the base 2 number system.

binary number system — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

binary-coded decimal (BCD) — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

bit — A binary digit. A bit has a value of either logic 0 or logic 1.

branch instruction — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

break module — The break module allows software to halt program execution at a programmable point in order to enter a background routine.

breakpoint — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

break interrupt — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

bus — A set of wires that transfers logic signals.

bus clock — See "CPU clock".

byte — A set of eight bits.

CAN — See "Motorola scalable CAN."

CCR — See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — See “clock generator module (CGM).”

clear — To change a bit from logic 1 to logic 0; the opposite of set.

clock — A square wave signal used to synchronize events in a computer.

clock generator module (CGM) — The CGM module generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and/or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

computer operating properly module (COP) — A counter module that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

Glossary

COP — See "computer operating properly module (COP)."

CPU — See “central processor unit (CPU).”

CPU12 — The CPU of the MC68HC12 Family.

CPU clock — Bus clock select bits BCSP and BCSS in the clock select register (CLKSEL) determine which clock drives SYSCLK for the main system, including the CPU and buses. When EXTALi drives the SYSCLK, the CPU or bus clock frequency (f_0) is equal to the EXTALi frequency divided by 2.

CPU cycles — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC12 are:

- A (8-bit accumulator)
- B (8-bit accumulator)
  - D (16-bit accumulator formed by concatenation of accumulators A and B)
- IX (16-bit index register)
- IY (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (8-bit condition code register)

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

D — See “accumulators (A and B or D).”

decimal number system — Base 10 numbering system that uses the digits zero through nine.

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

ECT — See “enhanced capture timer.”

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically erased and reprogrammed.

EPROM — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

enhanced capture timer (ECT) — The HC12 Enhanced Capture Timer module has the features of the HC12 Standard Timer module enhanced by additional features in order to enlarge the field of applications.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

flash EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically erased and reprogrammed. Does not support byte or word erase.

free-running counter — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

full-duplex transmission — Communication on a channel in which data can be sent and received simultaneously.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

high byte — The most significant eight bits of a word.

illegal address — An address not within the memory map

illegal opcode — A nonexistent opcode.

index registers (IX and IY) — Two 16-bit registers in the CPU. In the indexed addressing modes, the CPU uses the contents of IX or IY to determine the effective address of the operand. IX and IY can also serve as a temporary data storage locations.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

interrupt — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

interrupt request — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

I/O — See “input/output (I/0).”

jitter — Short-term signal instability.

latch — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

latency — The time lag between instruction completion and data movement.

least significant bit (LSB) — The rightmost digit of a binary number.

logic 1 — A voltage level approximately equal to the input power voltage (V_{DD}).

logic 0 — A voltage level approximately equal to the ground voltage (V_{SS}).

low byte — The least significant eight bits of a word.

M68HC12 — A Motorola family of 16-bit MCUs.

mark/space — The logic 1/logic 0 convention used in formatting data in serial communication.

mask — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

MCU — Microcontroller unit. See “microcontroller.”

memory location — Each M68HC12 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

MI-Bus — See "Motorola interconnect bus".

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

modulo counter — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

most significant bit (MSB) — The leftmost digit of a binary number.

Motorola interconnect bus (MI-Bus) — The Motorola Interconnect Bus (MI Bus) is a serial communications protocol which supports distributed real-time control efficiently and with a high degree of noise immunity.

Motorola scalable CAN (msCAN) — The Motorola scalable controller area network is a serial communications protocol that efficiently supports distributed real-time control with a very high level of data integrity.

msCAN — See "Motorola scalable CAN".

MSI — See "multiple serial interface".

multiple serial interface — A module consisting of multiple independent serial I/O sub-systems, e.g. two SCI and one SPI.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

nibble — A set of four bits (half of a byte).

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

opcode — A binary code that instructs the CPU to perform an operation.

open-drain — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

operand — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

OTPROM — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

overflow — A quantity that is too large to be contained in one byte or one word.

page zero — The first 256 bytes of memory (addresses $0000–$00FF).

parity — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

PC — See “program counter (PC).”

peripheral — A circuit not under direct CPU control.

phase-locked loop (PLL) — A clock generator circuit in which a voltage controlled oscillator produces an oscillation which is synchronized to a reference signal.

PLL — See "phase-locked loop (PLL)."

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

polarity — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, V_{DD} and V_{SS}.

polling — Periodically reading a status bit to monitor the condition of a peripheral device.

port — A set of wires for communicating with off-chip devices.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

program counter (PC) — A 16-bit register in the CPU. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

pullup — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

push — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

PWM period — The time required for one complete cycle of a PWM waveform.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To copy the contents of a memory location to the accumulator.

register — A circuit that stores a group of bits.

reserved memory location — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

reset — To force a device to a known condition.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

SCI — See "serial communication interface module (SCI)."

serial — Pertaining to sequential transmission over a single line.

serial communications interface module (SCI) — A module that supports asynchronous communication.

serial peripheral interface module (SPI) — A module that supports synchronous communication.

set — To change a bit from logic 0 to logic 1; opposite of clear.

shift register — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

signed — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

software — Instructions and data that control the operation of a microcontroller.

software interrupt (SWI) — An instruction that causes an interrupt and its associated vector fetch.

SPI — See "serial peripheral interface module (SPI)."

stack — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

stack pointer (SP) — A 16-bit register in the CPU containing the address of the next available storage location on the stack.

start bit — A bit that signals the beginning of an asynchronous serial transmission.

status bit — A register bit that indicates the condition of a device.

stop bit — A bit that signals the end of an asynchronous serial transmission.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

tracking mode — A mode of PLL operation with narrow loop bandwidth. Also see ‘acquisition mode.’

two’s complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two’s complement number indicates the sign of the number (1 indicates negative). The two’s complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unimplemented memory location — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value.

variable — A value that changes during the course of program execution.

VCO — See "voltage-controlled oscillator."

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

voltage-controlled oscillator (VCO) — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

wired-OR — Connection of circuit outputs so that if any output is high, the connection point is high.

word — A set of two bytes (16 bits).

write — The transfer of a byte of data from the CPU to a memory location.

Glossary

Literature Updates

This document contains the latest data available at publication time. For updates, contact one of the centers listed below:

Literature Distribution Centers

Order literature by mail or phone.

USA/Europe

Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado, 80217
Phone 1-303-675-2140

Japan

Motorola Japan Ltd.
Tatsumi-SPD-JLDC
Toshikatsu Otsuki
6F Seibu-Butsuryu Center
3-14-2 Tatsumi Koto-Ku
Tokyo 135, Japan
Phone 03-3521-8315

Hong Kong

Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
Phone 852-26629298

Customer Focus Center

1-800-521-6274

Microcontroller Division's Web Site

Directly access the Microcontroller Division's web site with the following URL:

http://mcu.motsps.com/

How to reach us:

USA/EUROPE: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140

JAPAN: Motorola Japan 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