Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

MC68HC05X4
MC68HC705X4
HCMOS Microcontroller Unit

TECHNICAL DATA

freescale™
semiconductor

List of Sections

MC68HC05X4   Rev 1.0

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

List of Sections

Freescale Semiconductor, Inc.

Table of Contents

MC68HC05X4 Rev 1.0

For More Information On This Product,
Go to: www.freescale.com

Table of Contents

Freescale Semiconductor, Inc.

**For More Information On This Product,
Go to: www.freescale.com**

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Contents

Introduction

The MC68HC05X4 with on-board controller area network (CAN) module is designed around the industry standard M68HC05 CPU core with its familiar and efficient instruction set. The Motorola CAN module (MCAN) is complete with line interface circuitry, comprising output drivers, input comparators and a $V_{DD}/2$ generator. The module can handle all the communication transactions flowing across a serial bus structure with minimal CPU intervention. Other features of the device include a 16-bit programmable timer, a 15-stage multi-purpose core timer and a computer operating properly (COP) watchdog timer. The MC68HC05X4 has two modes of operation, namely single chip and bootloader mode.

This data sheet covers both the MC68HC05X4 ROM based device and the equivalent EPROM based *MC68HC705X4* device. All references in the text to the MC68HC05X4 apply equally to the *MC68HC705X4*, unless otherwise stated. *References specific to the MC68HC705X4 are italicised in the text.*

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Features

- Fully static design featuring the industry standard M68HC05 core

- On-chip oscillator with divide-by-2 or divide-by-10 option

- 4096 bytes of ROM (MC68HC05X4); *4096 bytes EPROM (MC68HC705X4)*

- 176 bytes of RAM

- Single chip and bootloader modes of operation

- Power saving STOP and WAIT modes

- Motorola controller area network module (MCAN) with line interface circuitry (output drivers, input comparators, $V_{DD}/2$ generator)

- Extended temperature range: $-40°C$ to $+125°C$

- 16-bit programmable timer with input capture and output compare

- Multi-purpose core timer with computer operating properly (COP) watchdog and real time interrupt (RTI)

- Four hardware and one software interrupt sources

- 16 bi-directional I/O lines with wired-OR interrupt (WOI) capability

- Mask options for oscillator division ratio and COP disable

- 28-pin SOIC package

Introduction
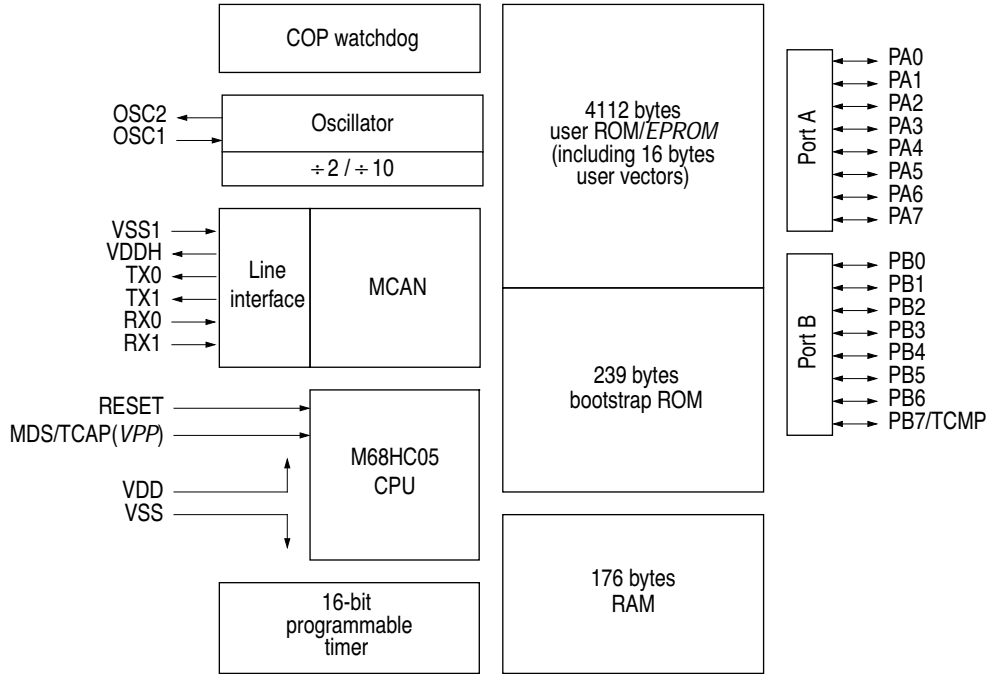
Figure 1. Functional block diagram

Mask options on the MC68HC05X4

There are two mask options on the MC68HC05X4. These bits are programmed during manufacture and must be specified on the order form.

- Oscillator division ratio selection (divide-by-2 or divide-by-10)

- COP watchdog timer (enable/disable)

Mask options on the MC68HC705X4

The same options as for the MC68HC05X4 are available on the *MC68HC705X4* device. These options must be programmed into the mask option register (MOR), at EPROM address $1F00, via the bootloader mode prior to operating the device in single chip mode. The MOR is latched in at reset in single chip mode to allow emulation of the masked ROM part. It is recommended that all unused bits in this register are written as '0'.

Mask option register (MOR)

This register may be written to only in bootloader mode.

Address: $1F00

	Bit 7	6	5	4	3	2	1	Bit 0
					DIV2	0	0	COP
Reset:	–	–	–	–	0	0	0	0

Figure 2. Mask Option Register (MOR)

DIV2 — Oscillator division ratio

1 = Selects divide-by-2 option.
0 = Selects divide-by-10 option.

NOTE: *The divide-by-10 option is forced during t_{PORL} (applies only to the MC68HC705X4).*

COP — Computer operating properly enable/disable

1 = Disables COP.
0 = Enables COP.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Modes of Operation and Pin Descriptions

Contents

Introduction

The MC68HC05X4 has two modes of operation: single chip and bootloader. **Table 1** shows the conditions required to enter each mode on the rising edge of RESET.

Table 1. Operating mode entry conditions

MDS/TCAP (*VPP*)	PB1	PB7	Mode
V_{SS} to V_{DD}	x	x	Single chip
2 V_{DD} V_{PP} (*EPROM*)	V_{DD}	V_{SS}	Bootloader

Modes of Operation and Pin Descriptions

Single chip mode

This is the normal operating mode of the MC68HC05X4. In this mode the device functions as a self-contained microcomputer with all on-board peripherals, including the two 8-bit I/O ports available to the user.

NOTE: *For the MC68HC705X4 all vectors are fetched from EPROM (locations $1FF0–$1FFF) in single chip mode; therefore, the EPROM must be programmed (via the bootloader mode) before the device is powered up in single chip mode. The mask option register is loaded from the EPROM ($1F00) on reset so that emulation of mask programmable features is possible.*

Single chip mode is entered on the rising edge of RESET if the voltage level on the MDS/TCAP*(VPP)* pin is within the normal operating range.

Bootloader modes for the MC68HC05X4 and MC68HC705X4

The entry conditions for these modes are identical. The bootloader mode on the *MC68HC705X4* is therefore a direct replacement for that on the MC68HC05X4.

Bootloader mode is entered on the rising edge of RESET if the MDS/TCAP(*VPP)* pin is at 2 x V_{DD}, the PB7 pin is at logic zero and the PB1 pin at logic one, as shown in **Table 1**.

NOTE: *In order to program the EPROM correctly on the MC68HC705X4 the MDS/TCAP(VPP) pin must be at the appropriate voltage – see **DC electrical characteristics***.

Bootloader Mode for the MC68HC05X4

This mode allows the user to use his own test routines for the device by providing a simple 'RAM load and execute' routine in ROM.

To make use of this feature a circuit board should be constructed as shown in Figure 1. It is then possible, by correctly configuring port pins PA2 and PA3, to load a user program into RAM and then to run it. The following sections explain the functions of this mode.

Freescale Semiconductor, Inc.

NOTE: 'RAMST'   =   the address of the first byte of RAM   =$0050
            'BTROM'   =   the address of the first byte of Bootloader ROM = $1F01
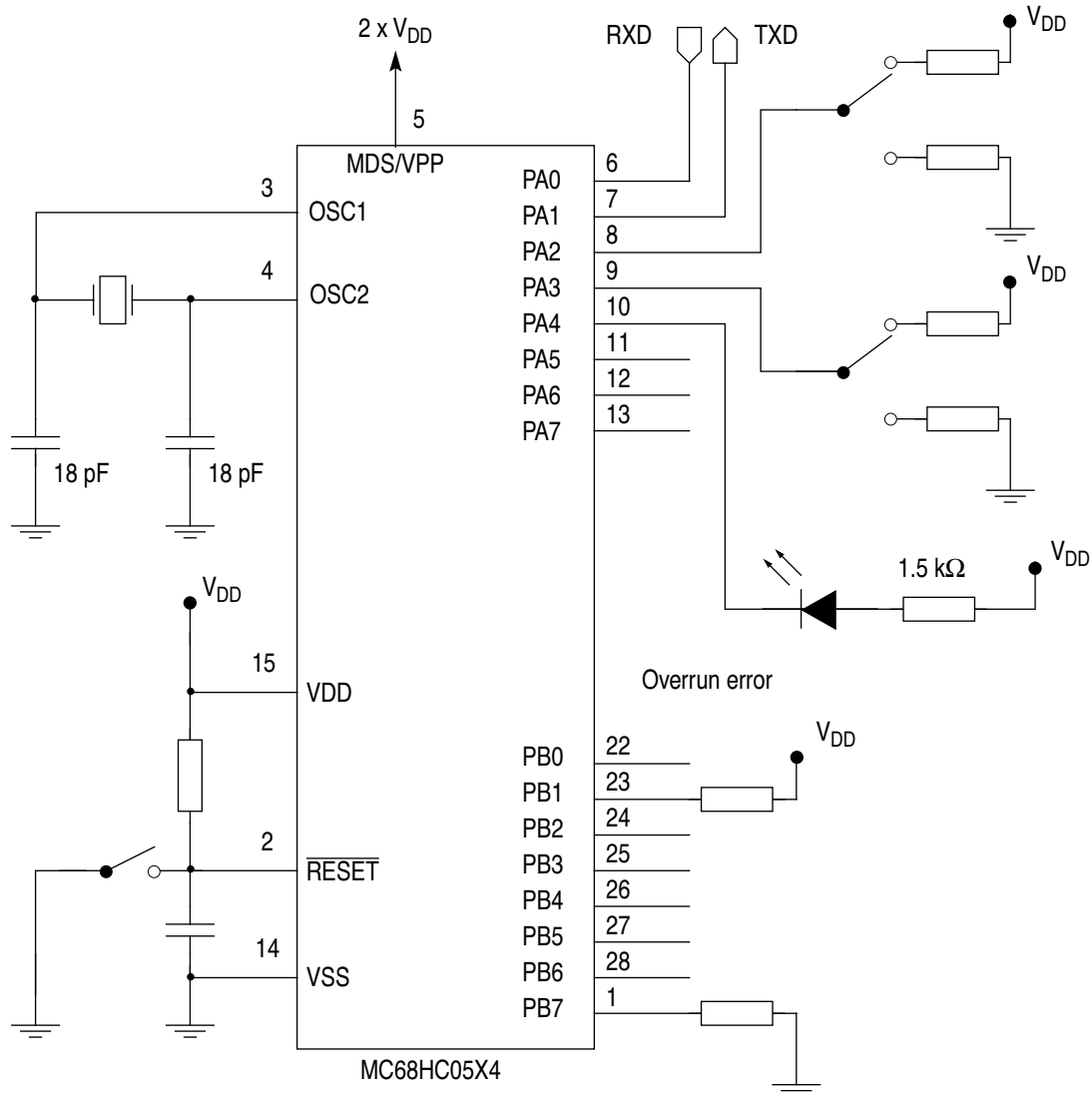
Note:

- All resistors are 10 kΩ unless otherwise specified
- For 9600 baud operation, crystal should be 20 MHz (with divide-by-10 option selected) or 4 MHz (for divide-by-2 option)

PA3	PA2	Mode
X	0	Test port B
0	1	Execute user code
1	1	Load user code

Figure 1. MC68HC05X4 bootloader circuit

Modes of Operation and Pin Descriptions
For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

**Bootloader data
format**

Serial data byte format is 8 data bits, no parity, one stop bit. The baud rate for both transmission and reception will be 9600 for a bus frequency (f_{OP}) of 2MHz.

Serial data packet format is:

(count) (byte 0) (byte 1) . . . (byte (count – 2))

Program execution will automatically commence after the last byte is received.

Port A pins are used as follows:

PA0   Serial receive data (RXD) – 9600 baud at fOP = 2MHz

PA1   Serial transmit data (TXD) – 9600 baud at fOP = 2MHz

PA2   'High'   Program enters load/execute mode.

       'Low'   Bootloader program runs a simple test routine which outputs the contents of the user ROM to port B one byte at a time.

PA3   (This bit is read only if PA2 is 'High'.)

       'High'   Load – bootloader routine waits for a serial byte. The first byte received is the total byte count (COUNT) for the message. The serial bytes received are loaded, starting at RAMST + 1. When all the bytes have been downloaded the program will automatically begin executing from RAMST + 1. The program will sit in a Wait loop indefinitely if fewer than 'COUNT' bytes are received.

       'Low'   Execute – The program passes control to RAMST +1. This can be used to allow Port A to be tested (using a user generated routine) as follows:

          Download program into RAM with PA3 = 'High', but ensuring that 'COUNT' is greater than the number of bytes to be transferred. Now, while the MCU is waiting for further bytes to arrive, apply a reset signal. Whilst holding RESET low, connect port A as required for testing, then release RESET. The test program in RAM will now test port A.

PA4   'High'   This indicates normal operation.

       'Low'   This bit is cleared if an overrun error occurs. The device should be reset if this happens.

MC68HC05X4 Rev 1.0

4-modes

Modes of Operation and Pin Descriptions
**For More Information On This Product,
Go to: www.freescale.com**

Bootloader routines

The program in bootloader ROM contains the following routines, which are accessed through a jump table at the start of the bootloader program (all routines end with an RTS instruction):

Receive serial byte from user (address: BTROM + 3)

This routine allows a serial byte to be read from PA0. The data read is written directly into memory, using the index register as an offset from the address of the first byte of RAM.

Transmit serial byte to user (address: BTROM + 6)

This routine allows the contents of the accumulator to be transmitted via PA1.

Examine user memory byte (address: BTROM + 9)

This routine reads two bytes from the serial receive pin (RXD). These are taken to be the upper and lower address bytes of the location to be examined. The address is then read and transmitted through the serial transmit pin (TXD).
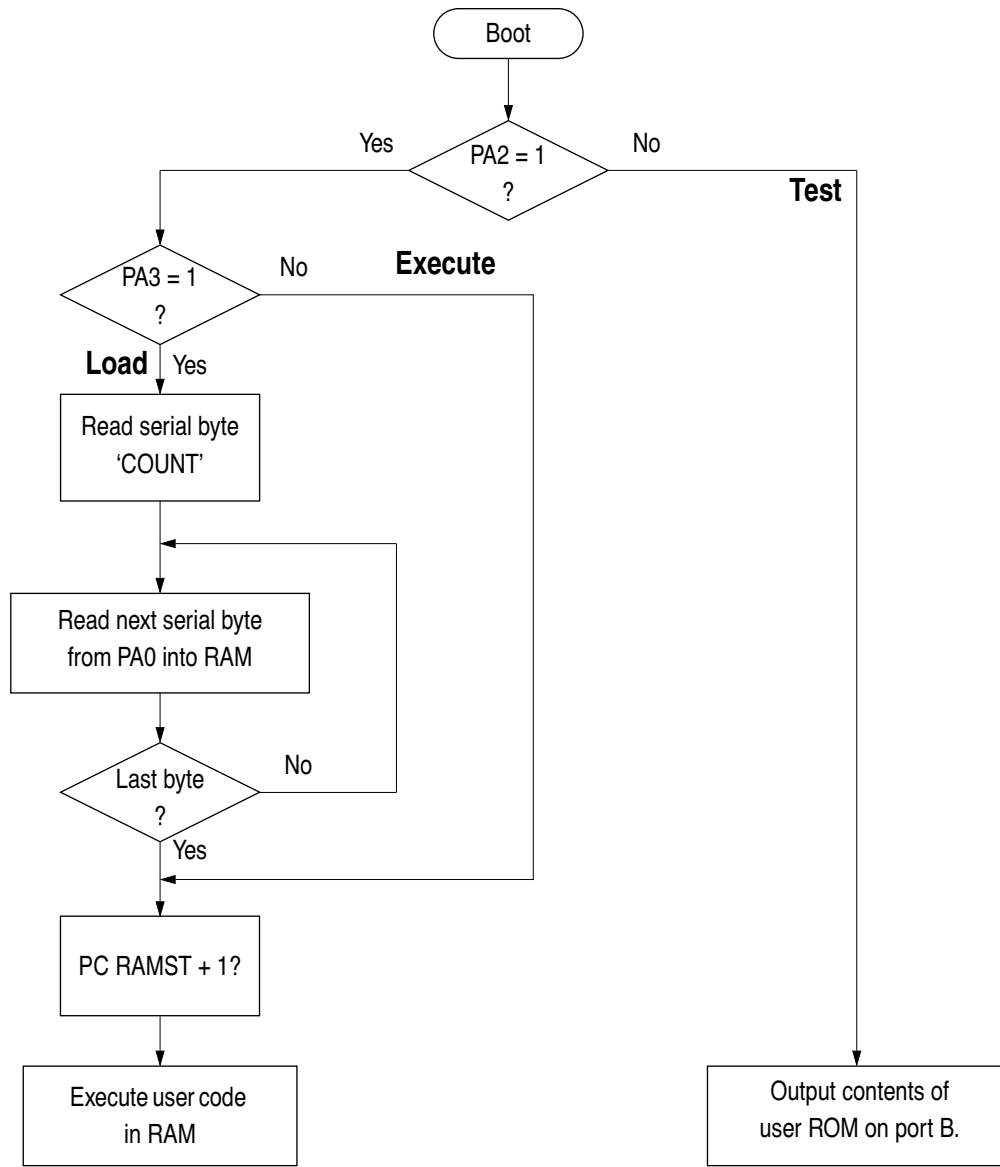
Freescale Semiconductor, Inc.

```
                            ┌──────────┐
                            │   Boot   │
                            └────┬─────┘
                                 │
                                 ▼
              Yes            ╱PA2 = 1╲           No
         ◄───────────────── ╲   ?   ╱ ──────────────────┐
         │                   ╲    ╱                      │
         │                                          Test │
         ▼                                               │
      ╱PA3 = 1╲      No      Execute                      │
      ╲   ?   ╱ ──────────────────┐                      │
       ╲    ╱                     │                      │
   Load   │ Yes                   │                      │
          ▼                       │                      │
   ┌───────────────┐              │                      │
   │ Read serial   │              │                      │
   │ byte 'COUNT'  │              │                      │
   └───────┬───────┘              │                      │
           │                      │                      │
           ▼◄──────────┐          │                      │
   ┌───────────────┐   │          │                      │
   │Read next serial│  │          │                      │
   │byte from PA0   │  │          │                      │
   │into RAM        │  │          │                      │
   └───────┬───────┘   │          │                      │
           ▼           │          │                      │
      ╱Last byte╲  No  │          │                      │
      ╲    ?    ╱──────┘          │                      │
       ╲      ╱                   │                      │
          │ Yes                   │                      │
          ▼◄──────────────────────┘                      │
   ┌───────────────┐                                     │
   │ PC RAMST + 1? │                                     │
   └───────┬───────┘                                     │
           ▼                                             ▼
   ┌───────────────┐                       ┌───────────────────┐
   │Execute user   │                       │Output contents of │
   │code in RAM    │                       │user ROM on port B.│
   └───────────────┘                       └───────────────────┘
```

Figure 2. MC68HC05X4 bootloader flowchart

Bootloader mode for the MC68HC705X4

This mode is used for programming the on-board EPROM and mask option register. In bootloader mode the operation of the device is the same as in single chip mode, except that the vectors are fetched from the bootloader ROM (locations $1F01 to $1FEF) instead of the EPROM. The pin assignments are therefore identical to that of single chip mode shown earlier. Because the addresses in the *MC68HC705X4* and the

MC68HC05X4 Rev 1.0   6-modes

Modes of Operation and Pin Descriptions
For More Information On This Product,
Go to: www.freescale.com

EPROM containing the user code are incremented independently it is essential that the data layout in the 2764 EPROM conforms exactly to the *MC68HC705X4* memory map.

The bootloader uses an external 12-bit counter with a clock and a reset function to address the memory device containing the code to be copied. The 12-bit counter can address up to 4K bytes of memory therefore a port pin must be used to address the additional memory space.

NOTE: *In bootloader mode the device will always default to the divide-by-10 option for the oscillator division ratio.*

EPROM programming register

All necessary manipulation of this register is carried out automatically by the bootloader routine.

Address: $1G00

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	0	ELAT	0	EPGM
Reset:	0	0	0	0	0	0	0	0

Figure 3. EPROM Programming Register (EPROG)

ELAT — EPROM address and data latch
    1 = Configures EPROM address and data buses for programming.
    0 = Configures EPROM address and data buses for normal
        operation.

EPGM — EPROM programming power control
    1 = Switches EPROM programming power on.
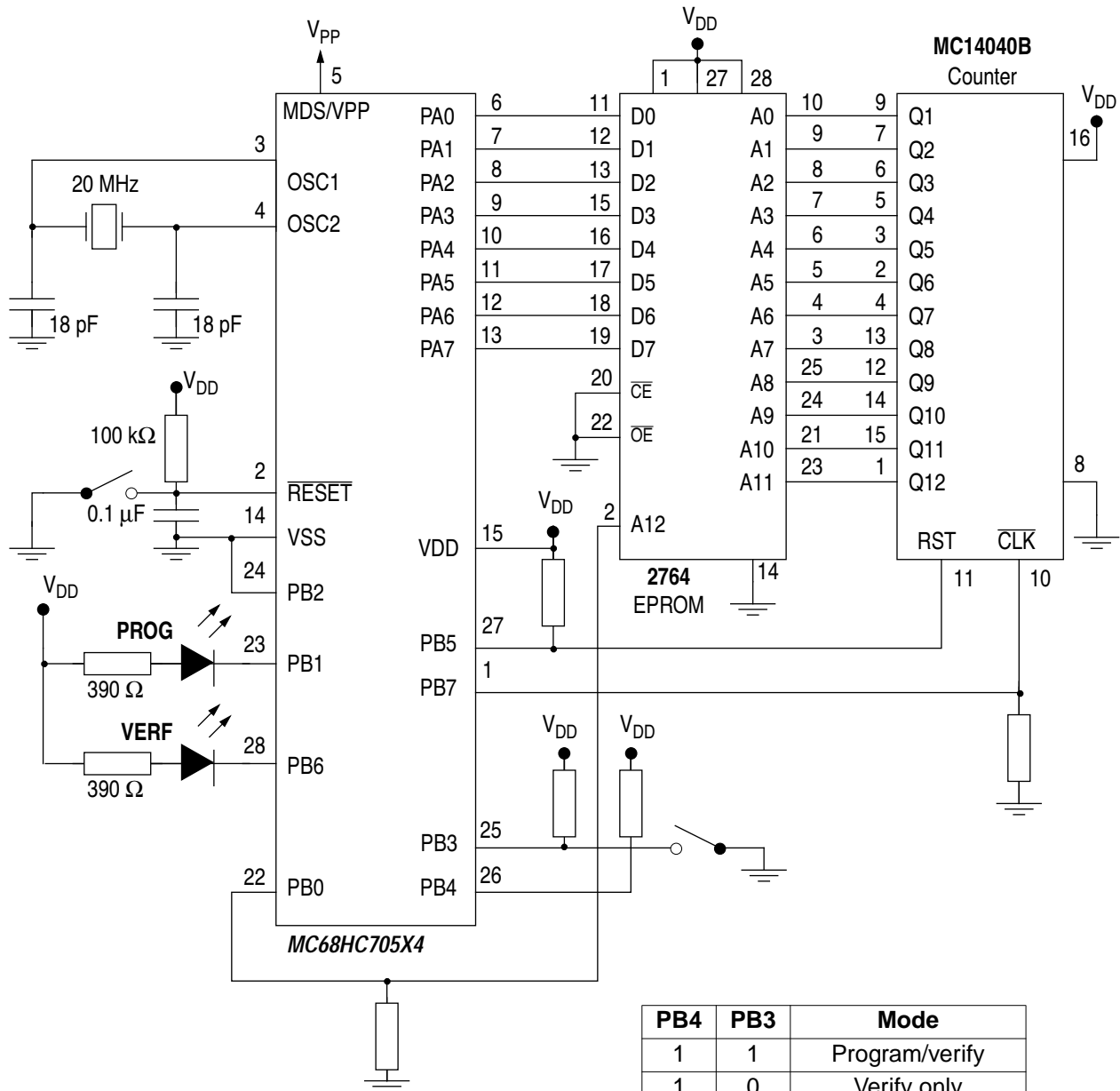   0 = Switches EPROM programming power off.

A byte of EPROM is programmed using the following sequence:

– set ELAT

– write data to the desired EPROM address

– set EPGM for time t_{PROG}

– clear EPGM/ELAT

Bootloader functions

The bootloader code deals with the copying of user code from an external EPROM into the on-chip EPROM. The bootloader function can only be used with an external EPROM. The bootloader performs a programming pass and then does a verify pass.

Pins PB3 and PB4 are used to select various bootloader functions, including the programming mode (see Figure 4). Two other pins, PB1 and PB6, are used to drive the PROG and VERF LED outputs. While the EPROM is being programmed the PROG LED flashes and when programming is complete the VERF LED flashes.

MC68HC05X4 Rev 1.0

8-modes

Modes of Operation and Pin Descriptions
**For More Information On This Product,
Go to: www.freescale.com**

PB4	PB3	Mode
1	1	Program/verify
1	0	Verify only

All resistors are 10 kΩ unless otherwise specified

Figure 4. MC68HC705X4 EPROM programming circuit
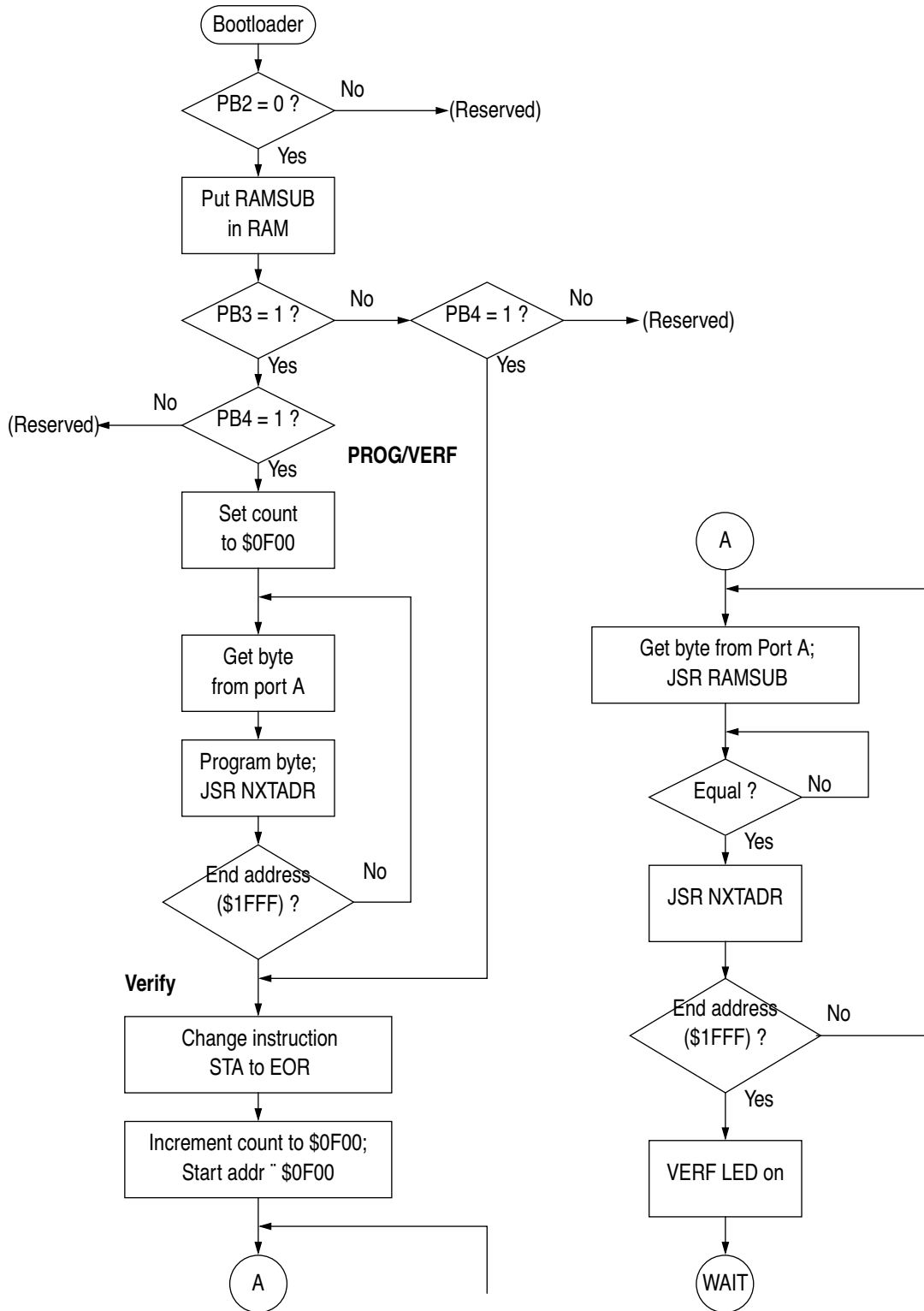
Figure 5. MC68HC705X4 bootloader flowchart

Modes of Operation and Pin Descriptions
For More Information On This Product,
Go to: www.freescale.com

Pin descriptions

PB7/TCMP — 1
RESET — 2
OSC1 — 3
OSC2 — 4
MDS/TCAP (VPP) — 5
PA0 — 6
PA1 — 7
PA2 — 8
PA3 — 9
PA4 — 10
PA5 — 11
PA6 — 12
PA7 — 13
VSS — 14

28 — PB6
27 — PB5
26 — PB4
25 — PB3
24 — PB2
23 — PB1
22 — PB0
21 — VDDH
20 — RX1
19 — RX0
18 — VSS1
17 — TX1
16 — TX0
15 — VDD
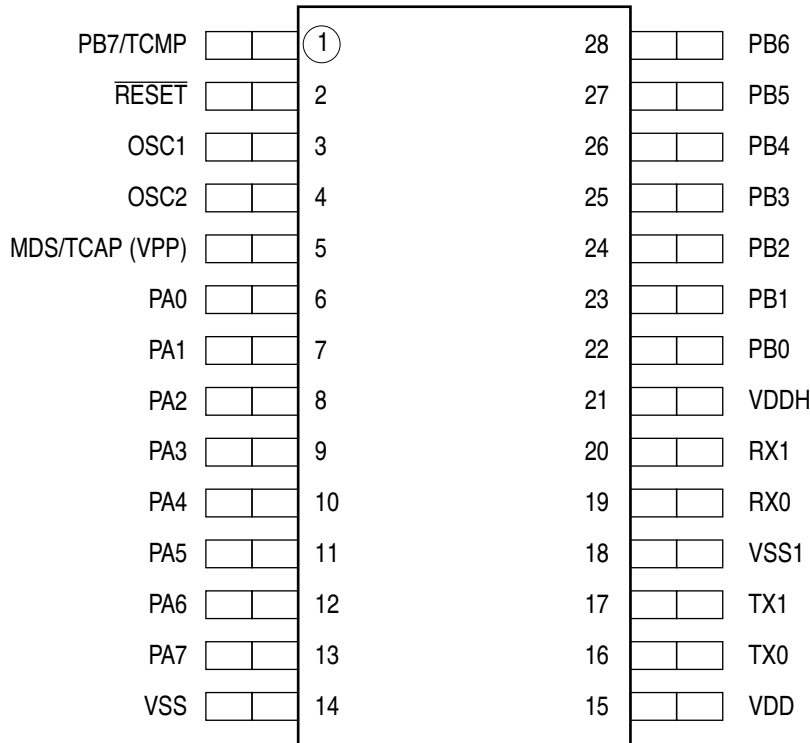
Figure 6. 28-pin SOIC pinout

VDD and VSS

Power is supplied to the microcomputer via these two pins. VDD is the positive supply and VSS is ground.

It is in the nature of CMOS designs that very fast signal transitions occur on the MCU pins. These short rise and fall times place very high short-duration current demands on the power supply. To prevent noise problems, special care must be taken to provide good power supply by-passing at the MCU. By-pass capacitors should have good high-frequency characteristics and be as close to the MCU as possible. By-passing requirements vary, depending on how heavily the MCU pins are loaded.

For More Information On This Product,
Go to: www.freescale.com

MDS/**TCAP** *(VPP)*   During reset this pin is used as a mode select input (MDS) to determine the operating mode. It also serves as the input capture (TCAP) pin for the 16-bit programmable timer. *In addition it is the EPROM programming voltage input pin (VPP) for the MC68HC705X4 device.*

OSC1/**OSC2**   These pins provide control input for an on-chip clock oscillator circuit. A crystal, ceramic resonator or external clock signal connected to these pins provides the oscillator clock. The oscillator frequency (f_{OSC}) is divided by 2 or by 10, chosen via a mask option, to provide the internal bus frequency (f_{OP}).

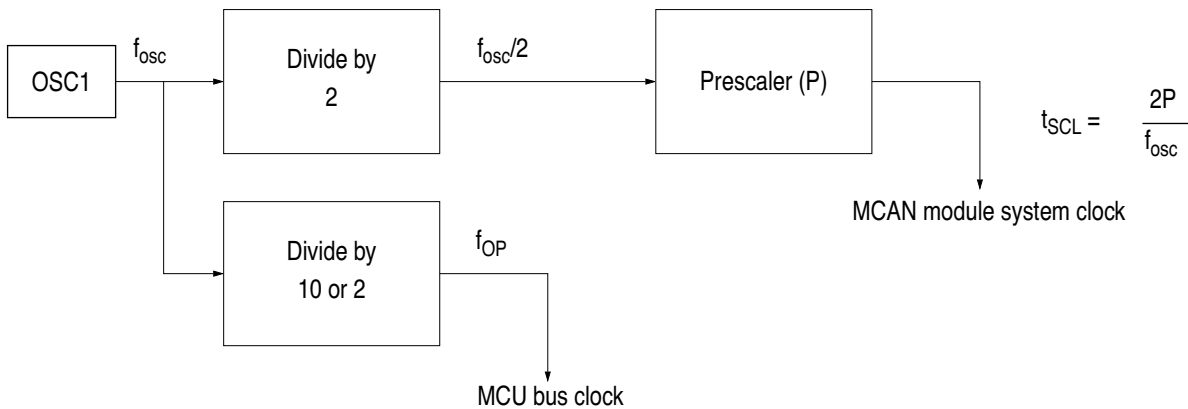*System clocks*   The MCU and MCAN system clocks are obtained as shown in the Figure 7.

Figure 7. Oscillator block diagram

Crystal   The circuit shown in Figure 8(a) is recommended when using a crystal. The internal oscillator is designed to interface with an AT-cut parallel-resonant quartz crystal resonator in the frequency range specified for f_{OSC} (refer to **DC electrical characteristics**). Use of an external CMOS oscillator is recommended when crystals outside the specified ranges are to be used. The crystal and components should be mounted as close as possible to the input pins to minimise output distortion and start-up stabilization time.

Ceramic resonator

A ceramic resonator may be used instead of the crystal in cost-sensitive applications. The circuit in Figure 8(a) is recommended when using a ceramic resonator. The manufacturer of the particular ceramic resonator being considered should be consulted for specific information. This option is recommended only for applications that operate at an external clock frequency of 8MHz or less. Any application requiring an external operating frequency greater that 8MHz should use either a crystal oscillator or an external CMOS compatible clock source.

External clock

An external clock should be applied to the OSC1 input with the OSC2 pin not connected, as shown in Figure 8(c). The t_{OXOV} specification does not apply when using an external clock input. The equivalent specification of the external clock source should be used in lieu of t_{OXOV}.
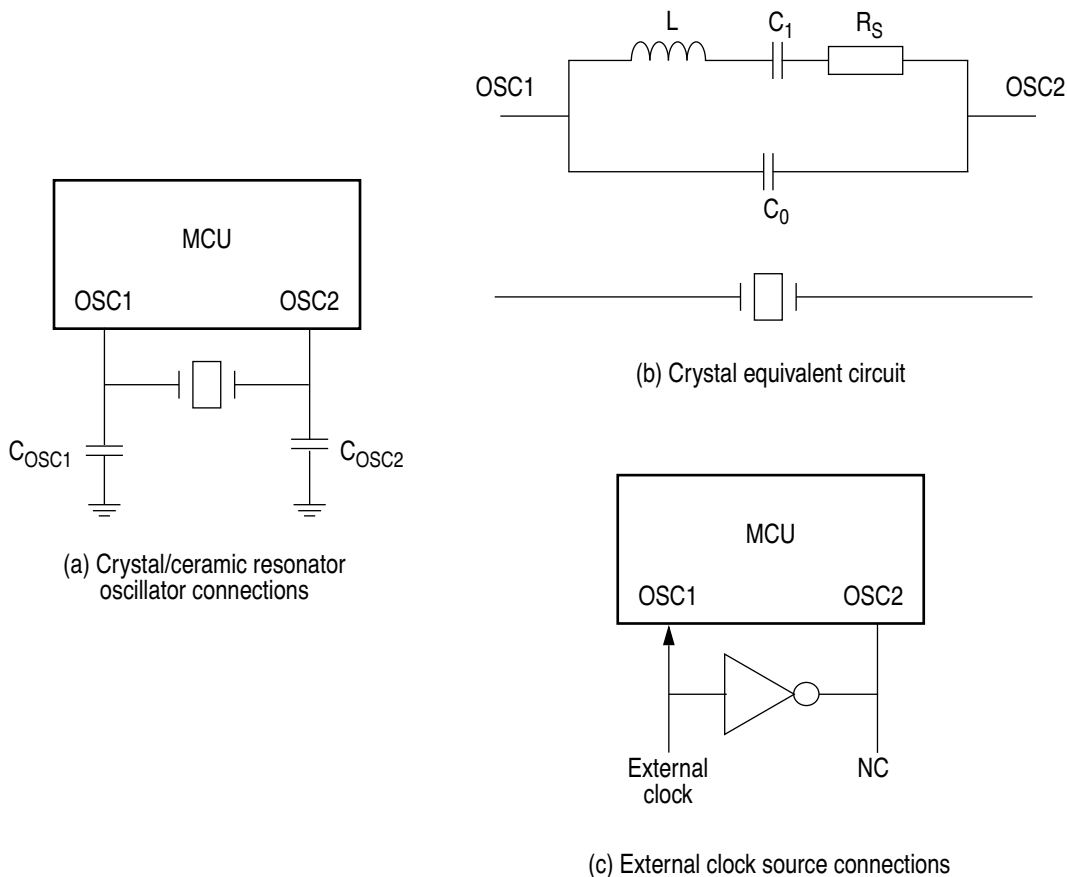
(a) Crystal/ceramic resonator oscillator connections

(b) Crystal equivalent circuit

(c) External clock source connections

Figure 8. Oscillator connections

Freescale Semiconductor, Inc.

Modes of Operation and Pin Descriptions

Freescale Semiconductor, Inc.

RESET This active low input pin is used to reset the MCU. Applying a logic zero to this pin forces the device to a known start-up state. An external RC-circuit can be connected to this pin to generate a power-on reset (POR) if required. In this case, the time constant must be great enough (minimum 100 ms) to allow the oscillator circuit to stabilise. This input has an internal Schmitt trigger to improve noise immunity.

PA0–PA7/PB0–PB7 These sixteen I/O lines comprise ports A and B. The state of any pin is software programmable and all port A and B lines are configured as inputs at reset.

When the 16-bit programmable timer is enabled (via the TIMEN bit in the port configuration register) the PB7/TCMP pin is used for the TCMP output compare function instead of for the port pin. When the current value of the 16-bit timer counter matches the value held in the output compare register then the level specified by the OLVL bit in the timer control register is clocked out to the TCMP pin.

VDDH This pin provides the high voltage reference output for the CAN bus. The output voltage is equal to $V_{DD}/2$.

VSS1 This pin is the ground connection for the input comparator of the CAN bus.

RX0/RX1 These input pins connect the physical bus lines to the input comparator (receive). When the MCAN is in SLEEP mode, a 'dominant' level on these pins will waken it.

TX0/TX1 These output pins connect the output drivers of the MCAN to the physical bus lines (transmit).

NOTE: *CAN bus lines. The bus can have one of two complementary values – 'dominant' or 'recessive'. During simultaneous transmission of 'dominant' and 'recessive' bits the resulting bus value will be 'dominant'. For example, with a positive logic wired-AND implementation of the bus, the 'dominant' level would correspond to a logic '0', and the 'recessive' level to a logic '1'.*

MC68HC05X4 Rev 1.0

14-modes

Modes of Operation and Pin Descriptions
For More Information On This Product,
Go to: www.freescale.com

CPU

Contents

CPU

Introduction

This chapter describes the CPU registers and the HC05 instruction set.

CPU Registers

Figure 1 shows the five CPU registers. CPU registers are not part of the memory map.
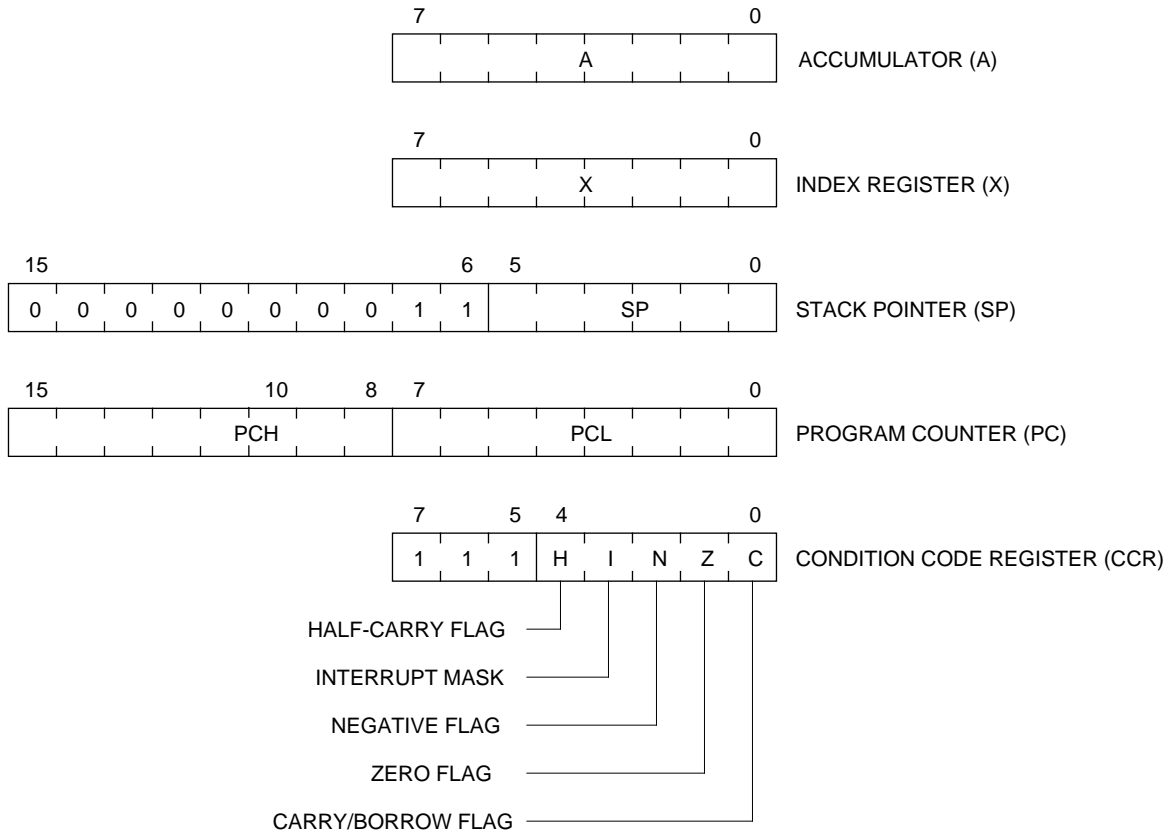

Figure 1. Programming Model

Accumulator   The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and results of arithmetic and non-arithmetic operations.

MC68HC05X4

Bit 7	6	5	4	3	2	1	Bit 0

Reset: Unaffected by reset

Figure 2. Accumulator

Index Register

In the indexed addressing modes, the CPU uses the byte in the index register to determine the conditional address of the operand.

Bit 7	6	5	4	3	2	1	Bit 0

Reset: Unaffected by reset

Figure 3. Index Register

The 8-bit index register can also serve as a temporary data storage location.

Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset or after the reset stack pointer (RSP) instruction, the stack pointer is preset to $00FF. The address in the stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

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

Reset 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

Figure 4. Stack Pointer

The ten most significant bits of the stack pointer are permanently fixed at 000000011, so the stack pointer produces addresses from $00C0 to $00FF. If subroutines and interrupts use more than 64 stack locations, the stack pointer wraps around to address $00FF and begins writing over the previously stored data. A subroutine uses two stack locations. An interrupt uses five locations.

Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched. The two most significant bits of the program counter are ignored internally.

Normally, the address in the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.
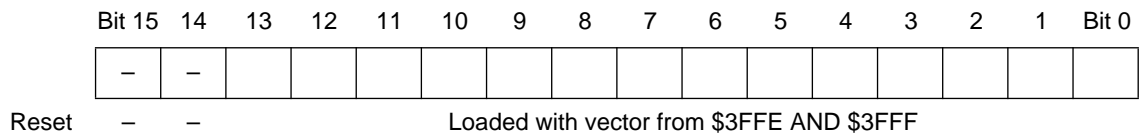
Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
–	–														

Reset   –   –   Loaded with vector from $3FFE AND $3FFF

Figure 5. Program Counter

Condition Code Register

The condition code register is an 8-bit register whose three most significant bits are permanently fixed at 111. The condition code register contains the interrupt mask and four flags that indicate the results of the instruction just executed. The following paragraphs describe the functions of the condition code register.
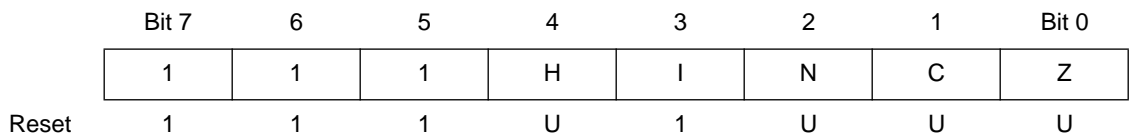
Bit 7	6	5	4	3	2	1	Bit 0
1	1	1	H	I	N	C	Z

Reset   1   1   1   U   1   U   U   U

Figure 6. Condition Code Register

Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between bits 3 and 4 of the accumulator during an ADD or ADC operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations.

Interrupt Mask

Setting the interrupt mask disables interrupts. If an interrupt request occurs while the interrupt mask is logic zero, the CPU saves the CPU registers on the stack, sets the interrupt mask, and then fetches the

interrupt vector. If an interrupt request occurs while the interrupt mask is set, the interrupt request is latched. Normally, the CPU processes the latched interrupt as soon as the interrupt mask is cleared again.

A return from interrupt (RTI) instruction pulls the CPU registers from the stack, restoring the interrupt mask to its cleared state. After any reset, the interrupt mask is set and can be cleared only by a software instruction.

Negative Flag

The CPU sets the negative flag when an arithmetic operation, logical operation, or data manipulation produces a negative result.

Zero Flag

The CPU sets the zero flag when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow flag.

Arithmetic/Logic Unit (ALU)

The ALU performs the arithmetic and logical operations defined by the instruction set.

The binary arithmetic circuits decode instructions and set up the ALU for the selected operation. Most binary arithmetic is based on the addition algorithm, carrying out subtraction as negative addition. Multiplication is not performed as a discrete operation but as a chain of addition and shift operations within the ALU. The multiply instruction (MUL) requires 11 internal clock cycles to complete this chain of operations.

Instruction Set Overview

The MCU instruction set has 62 instructions and uses eight addressing modes. The instructions include all those of the M146805 CMOS Family plus one more: the unsigned multiply (MUL) instruction. The MUL instruction allows unsigned multiplication of the contents of the accumulator (A) and the index register (X). The high-order product is stored in the index register, and the low-order product is stored in the accumulator.

Addressing Modes

The CPU uses eight addressing modes for flexibility in accessing data. The addressing modes provide eight different ways for the CPU to find the data required to execute an instruction. The eight addressing modes are:

- Inherent

- Immediate

- Direct

- Extended

- Indexed, no offset

- Indexed, 8-bit offset

- Indexed, 16-bit offset

- Relative

Inherent

Inherent instructions are those that have no operand, such as return from interrupt (RTI) and stop (STOP). Some of the inherent instructions act on data in the CPU registers, such as set carry flag (SEC) and increment accumulator (INCA). Inherent instructions require no operand address and are one byte long.

Immediate

Immediate instructions are those that contain a value to be used in an operation with the value in the accumulator or index register. Immediate instructions require no operand address and are two bytes long. The opcode is the first byte, and the immediate data value is the second byte.

Direct

Direct instructions can access any of the first 256 memory locations with two bytes. The first byte is the opcode, and the second is the low byte of the operand address. In direct addressing, the CPU automatically uses $00 as the high byte of the operand address.

Extended

Extended instructions use three bytes and can access any address in memory. The first byte is the opcode; the second and third bytes are the high and low bytes of the operand address.

When using the Motorola assembler, the programmer does not need to specify whether an instruction is direct or extended. The assembler automatically selects the shortest form of the instruction.

Indexed, No Offset

Indexed instructions with no offset are 1-byte instructions that can access data with variable addresses within the first 256 memory locations. The index register contains the low byte of the effective address of the operand. The CPU automatically uses $00 as the high byte, so these instructions can address locations $0000–$00FF.

Indexed, no offset instructions are often used to move a pointer through a table or to hold the address of a frequently used RAM or I/O location.

Indexed, 8-Bit Offset

Indexed, 8-bit offset instructions are 2-byte instructions that can access data with variable addresses within the first 511 memory locations. The CPU adds the unsigned byte in the index register to the unsigned byte following the opcode. The sum is the effective address of the operand. These instructions can access locations $0000–$01FE.

Indexed 8-bit offset instructions are useful for selecting the kth element in an n-element table. The table can begin anywhere within the first 256 memory locations and could extend as far as location 510 ($01FE). The k value is typically in the index register, and the address of the beginning of the table is in the byte following the opcode.

Indexed,16-Bit Offset

Indexed, 16-bit offset instructions are 3-byte instructions that can access data with variable addresses at any location in memory. The CPU adds the unsigned byte in the index register to the two unsigned bytes following the opcode. The sum is the effective address of the operand. The first byte after the opcode is the high byte of the 16-bit offset; the second byte is the low byte of the offset.

Indexed, 16-bit offset instructions are useful for selecting the kth element in an n-element table anywhere in memory.

As with direct and extended addressing, the Motorola assembler determines the shortest form of indexed addressing.

Relative

Relative addressing is only for branch instructions. If the branch condition is true, the CPU finds the effective branch destination by adding the signed byte following the opcode to the contents of the program counter. If the branch condition is not true, the CPU goes to the next instruction. The offset is a signed, two's complement byte that gives a branching range of −128 to +127 bytes from the address of the next location after the branch instruction.

When using the Motorola assembler, the programmer does not need to calculate the offset, because the assembler determines the proper offset and verifies that it is within the span of the branch.

Instruction Types

The MCU instructions fall into the following five categories:

- Register/Memory Instructions

- Read-Modify-Write Instructions

- Jump/Branch Instructions

- Bit Manipulation Instructions

- Control Instructions

Register/Memory Instructions

These instructions operate on CPU registers and memory locations. Most of them use two operands. One operand is in either the accumulator or the index register. The CPU finds the other operand in memory.

Table 1. Register/Memory Instructions

Instruction	Mnemonic
Add Memory Byte and Carry Bit to Accumulator	ADC
Add Memory Byte to Accumulator	ADD
AND Memory Byte with Accumulator	AND
Bit Test Accumulator	BIT
Compare Accumulator	CMP
Compare Index Register with Memory Byte	CPX
EXCLUSIVE OR Accumulator with Memory Byte	EOR
Load Accumulator with Memory Byte	LDA
Load Index Register with Memory Byte	LDX
Multiply	MUL
OR Accumulator with Memory Byte	ORA

Table 1. Register/Memory Instructions

Subtract Memory Byte and Carry Bit from Accumulator	SBC
Store Accumulator in Memory	STA
Store Index Register in Memory	STX
Subtract Memory Byte from Accumulator	SUB

Read-Modify-Write Instructions

These instructions read a memory location or a register, modify its contents, and write the modified value back to the memory location or to the register.

NOTE: *Do not use read-modify-write operations on write-only registers.*

Table 2. Read-Modify-Write Instructions

Instruction	Mnemonic
Arithmetic Shift Left (Same as LSL)	ASL
Arithmetic Shift Right	ASR
Bit Clear	BCLR[1]
Bit Set	BSET[1]
Clear Register	CLR
Complement (One's Complement)	COM
Decrement	DEC
Increment	INC
Logical Shift Left (Same as ASL)	LSL
Logical Shift Right	LSR
Negate (Two's Complement)	NEG
Rotate Left through Carry Bit	ROL
Rotate Right through Carry Bit	ROR
Test for Negative or Zero	TST[2]

1. Unlike other read-modify-write instructions, BCLR and BSET use only direct addressing.
2. TST is an exception to the read-modify-write sequence because it does not write a replacement value.

Freescale Semiconductor, Inc.

Jump/Branch Instructions

Jump instructions allow the CPU to interrupt the normal sequence of the program counter. The unconditional jump instruction (JMP) and the jump-to-subroutine instruction (JSR) have no register operand. Branch instructions allow the CPU to interrupt the normal sequence of the program counter when a test condition is met. If the test condition is not met, the branch is not performed.

The BRCLR and BRSET instructions cause a branch based on the state of any readable bit in the first 256 memory locations. These 3-byte instructions use a combination of direct addressing and relative addressing. The direct address of the byte to be tested is in the byte following the opcode. The third byte is the signed offset byte. The CPU finds the effective branch destination by adding the third byte to the program counter if the specified bit tests true. The bit to be tested and its condition (set or clear) is part of the opcode. The span of branching is from –128 to +127 from the address of the next location after the branch instruction. The CPU also transfers the tested bit to the carry/borrow bit of the condition code register.

Table 3. Jump and Branch Instructions

Instruction	Mnemonic
Branch if Carry Bit Clear	BCC
Branch if Carry Bit Set	BCS
Branch if Equal	BEQ
Branch if Half-Carry Bit Clear	BHCC
Branch if Half-Carry Bit Set	BHCS
Branch if Higher	BHI
Branch if Higher or Same	BHS
Branch if $\overline{IRQ}$ Pin High	BIH
Branch if $\overline{IRQ}$ Pin Low	BIL
Branch if Lower	BLO
Branch if Lower or Same	BLS
Branch if Interrupt Mask Clear	BMC
Branch if Minus	BMI
Branch if Interrupt Mask Set	BMS
Branch if Not Equal	BNE
Branch if Plus	BPL
Branch Always	BRA
Branch if Bit Clear	BRCLR
Branch Never	BRN
Branch if Bit Set	BRSET
Branch to Subroutine	BSR
Unconditional Jump	JMP
Jump to Subroutine	JSR

Bit Manipulation Instructions

The CPU can set or clear any writable bit in the first 256 bytes of memory, which includes I/O registers and on-chip RAM locations. The

CPU can also test and branch based on the state of any bit in any of the first 256 memory locations.

Table 4. Bit Manipulation Instructions

Instruction	Mnemonic
Bit Clear	BCLR
Branch if Bit Clear	BRCLR
Branch if Bit Set	BRSET
Bit Set	BSET

Control Instructions

These instructions act on CPU registers and control CPU operation during program execution.

Table 5. Control Instructions

Instruction	Mnemonic
Clear Carry Bit	CLC
Clear Interrupt Mask	CLI
No Operation	NOP
Reset Stack Pointer	RSP
Return from Interrupt	RTI
Return from Subroutine	RTS
Set Carry Bit	SEC
Set Interrupt Mask	SEI
Stop Oscillator and Enable $\overline{\text{IRQ}}$ Pin	STOP
Software Interrupt	SWI
Transfer Accumulator to Index Register	TAX
Transfer Index Register to Accumulator	TXA
Stop CPU Clock and Enable Interrupts	WAIT

Freescale Semiconductor, Inc.

Instruction Set Summary

Table 6. Instruction Set Summary
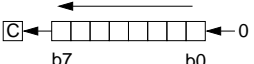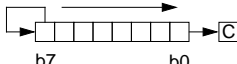
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
ADC #opr ADC opr ADC opr ADC opr,X ADC opr,X ADC ,X	Add with Carry	$A \leftarrow (A) + (M) + (C)$	↕◊	—	↕◊	↕◊	↕◊	IMM DIR EXT IX2 IX1 IX	A9 B9 C9 D9 E9 F9	ii dd hh ll ee ff ff	2 3 4 5 4 3
ADD #opr ADD opr ADD opr ADD opr,X ADD opr,X ADD ,X	Add without Carry	$A \leftarrow (A) + (M)$	↕◊	—	↕◊	↕	↕	IMM DIR EXT IX2 IX1 IX	AB BB CB DB EB FB	ii dd hh ll ee ff ff	2 3 4 5 4 3
AND #opr AND opr AND opr AND opr,X AND opr,X AND ,X	Logical AND	$A \leftarrow (A) \wedge (M)$	—	—	↕◊	↕	—	IMM DIR EXT IX2 IX1 IX	A4 B4 C4 D4 E4 F4	ii dd hh ll ee ff ff	2 3 4 5 4 3
ASL opr ASLA ASLX ASL opr,X ASL ,X	Arithmetic Shift Left (Same as LSL)	C ← [ ] ← 0 b7 b0	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff	5 3 3 6 5
ASR opr ASRA ASRX ASR opr,X ASR ,X	Arithmetic Shift Right	[ ] → C b7 b0	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	37 47 57 67 77	dd ff	5 3 3 6 5
BCC rel	Branch if Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? C = 0$	—	—	—	—	—	REL	24	rr	3
BCLR n opr	Clear Bit n	$Mn \leftarrow 0$	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS rel	Branch if Carry Bit Set (Same as BLO)	$PC \leftarrow (PC) + 2 + rel ? C = 1$	—	—	—	—	—	REL	25	rr	3
BEQ rel	Branch if Equal	$PC \leftarrow (PC) + 2 + rel ? Z = 1$	—	—	—	—	—	REL	27	rr	3
BHCC rel	Branch if Half-Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? H = 0$	—	—	—	—	—	REL	28	rr	3
BHCS rel	Branch if Half-Carry Bit Set	$PC \leftarrow (PC) + 2 + rel ? H = 1$	—	—	—	—	—	REL	29	rr	3
BHI rel	Branch if Higher	$PC \leftarrow (PC) + 2 + rel ? C \vee Z = 0$	—	—	—	—	—	REL	22	rr	3

MC68HC05X4

14-cpu

Table 6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
BHS rel	Branch if Higher or Same	PC ← (PC) + 2 + rel ? C = 0	—	—	—	—	—	REL	24	rr	3
BIH rel	Branch if IRQ Pin High	PC ← (PC) + 2 + rel ? IRQ = 1	—	—	—	—	—	REL	2F	rr	3
BIL rel	Branch if IRQ Pin Low	PC ← (PC) + 2 + rel ? IRQ = 0	—	—	—	—	—	REL	2E	rr	3
BIT #opr BIT opr BIT opr BIT opr,X BIT opr,X BIT ,X	Bit Test Accumulator with Memory Byte	(A) ∧ (M)	—	—	↕◊	↕	—	IMM DIR EXT IX2 IX1 IX	A5 B5 C5 D5 E5 F5	ii dd hh ll ee ff ff	2 3 4 5 4 3
BLO rel	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + rel ? C = 1	—	—	—	—	—	REL	25	rr	3
BLS rel	Branch if Lower or Same	PC ← (PC) + 2 + rel ? C ∨ Z = 1	—	—	—	—	—	REL	23	rr	3
BMC rel	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + rel ? I = 0	—	—	—	—	—	REL	2C	rr	3
BMI rel	Branch if Minus	PC ← (PC) + 2 + rel ? N = 1	—	—	—	—	—	REL	2B	rr	3
BMS rel	Branch if Interrupt Mask Set	PC ← (PC) + 2 + rel ? I = 1	—	—	—	—	—	REL	2D	rr	3
BNE rel	Branch if Not Equal	PC ← (PC) + 2 + rel ? Z = 0	—	—	—	—	—	REL	26	rr	3
BPL rel	Branch if Plus	PC ← (PC) + 2 + rel ? N = 0	—	—	—	—	—	REL	2A	rr	3
BRA rel	Branch Always	PC ← (PC) + 2 + rel ? 1 = 1	—	—	—	—	—	REL	20	rr	3
BRCLR n opr rel	Branch if Bit n Clear	PC ← (PC) + 2 + rel ? Mn = 0	—	—	—	—	↕◊	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN rel	Branch Never	PC ← (PC) + 2 + rel ? 1 = 0	—	—	—	—	—	REL	21	rr	3
BRSET n opr rel	Branch if Bit n Set	PC ← (PC) + 2 + rel ? Mn = 1	—	—	—	—	◊↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET n opr	Set Bit n	Mn ← 1	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR rel	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) − 1; push (PCH) SP ← (SP) − 1 PC ← (PC) + rel	—	—	—	—	—	REL	AD	rr	6
CLC	Clear Carry Bit	C ← 0	—	—	—	—	0	INH	98		2

Table 6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
CLI	Clear Interrupt Mask	I ← 0	—	0	—	—	—	INH	9A		2
CLR *opr* CLRA CLRX CLR *opr*,X CLR ,X	Clear Byte	M ← $00 A ← $00 X ← $00 M ← $00 M ← $00	—	—	0	1	—	DIR INH INH IX1 IX	3F 4F 5F 6F 7F	dd ff	5 3 3 6 5
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X	Compare Accumulator with Memory Byte	(A) − (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A1 B1 C1 D1 E1 F1	ii dd hh ll ee ff ff	2 3 4 5 4 3
COM *opr* COMA COMX COM *opr*,X COM ,X	Complement Byte (One's Complement)	M ← ($\overline{M}$) = $FF − (M) A ← ($\overline{A}$) = $FF − (A) X ← ($\overline{X}$) = $FF − (X) M ← ($\overline{M}$) = $FF − (M) M ← ($\overline{M}$) = $FF − (M)	—	—	↕	↕	1	DIR INH INH IX1 IX	33 43 53 63 73	dd ff	5 3 3 6 5
CPX #*opr* CPX *opr* CPX *opr* CPX *opr*,X CPX *opr*,X CPX ,X	Compare Index Register with Memory Byte	(X) − (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A3 B3 C3 D3 E3 F3	ii dd hh ll ee ff ff	2 3 4 5 4 3
DEC *opr* DECA DECX DEC *opr*,X DEC ,X	Decrement Byte	M ← (M) − 1 A ← (A) − 1 X ← (X) − 1 M ← (M) − 1 M ← (M) − 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3A 4A 5A 6A 7A	dd ff	5 3 3 6 5
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X	EXCLUSIVE OR Accumulator with Memory Byte	A ← (A) ⊕ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A8 B8 C8 D8 E8 F8	ii dd hh ll ee ff ff	2 3 4 5 4 3
INC *opr* INCA INCX INC *opr*,X INC ,X	Increment Byte	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3C 4C 5C 6C 7C	dd ff	5 3 3 6 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Unconditional Jump	PC ← Jump Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2

MC68HC05X4

Table 6. Instruction Set Summary (Continued)

Source Form	Operation	Description	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) − 1 Push (PCH); SP ← (SP) − 1 PC ← Effective Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 7 6 5
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X	Load Accumulator with Memory Byte	A ← (M)	—	—	↕◊	↕	—	IMM DIR EXT IX2 IX1 IX	A6 B6 C6 D6 E6 F6	ii dd hh ll ee ff ff	2 3 4 5 4 3
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X	Load Index Register with Memory Byte	X ← (M)	—	—	↕◊	↕◊	—	IMM DIR EXT IX2 IX1 IX	AE BE CE DE EE FE	ii dd hh ll ee ff ff	2 3 4 5 4 3
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X	Logical Shift Left (Same as ASL)	C ← [b7 ... b0] ← 0	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff	5 3 3 6 5
LSR *opr* LSRA LSRX LSR *opr*,X LSR ,X	Logical Shift Right	0 → [b7 ... b0] → C	—	—	0	↕	↕	DIR INH INH IX1 IX	34 44 54 64 74	dd ff	5 3 3 6 5
MUL	Unsigned Multiply	X : A ← (X) × (A)	0	—	—	—	0	INH	42		11
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X	Negate Byte (Two's Complement)	M ← −(M) = $00 − (M) A ← −(A) = $00 − (A) X ← −(X) = $00 − (X) M ← −(M) = $00 − (M) M ← −(M) = $00 − (M)	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	30 40 50 60 70	dd ff	5 3 3 6 5
NOP	No Operation		—	—	—	—	—	INH	9D		2
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X	Logical OR Accumulator with Memory	A ← (A) ∨ (M)	—	—	↕◊	↕	—	IMM DIR EXT IX2 IX1 IX	AA BA CA DA EA FA	ii dd hh ll ee ff ff	2 3 4 5 4 3
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X	Rotate Byte Left through Carry Bit	C ← [b7 ... b0] ←	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	39 49 59 69 79	dd ff	5 3 3 6 5

Table 6. Instruction Set Summary (Continued)
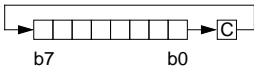
Source Form	Operation	Description	Effect on CCR H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
ROR opr RORA RORX ROR opr,X ROR ,X	Rotate Byte Right through Carry Bit	(diagram) b7 → → C → b0	—	—	↕◊	↕	↕	DIR INH INH IX1 IX	36 46 56 66 76	dd ff	5 3 3 6 5
RSP	Reset Stack Pointer	SP ← $00FF	—	—	—	—	—	INH	9C		2
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕◊	↕	↕	↕	↕	INH	80		9
RTS	Return from Subroutine	SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	—	—	—	—	—	INH	81		6
SBC #opr SBC opr SBC opr SBC opr,X SBC opr,X SBC ,X	Subtract Memory Byte and Carry Bit from Accumulator	A ← (A) − (M) − (C)	—	—	◊↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A2 B2 C2 D2 E2 F2	ii dd hh ll ee ff ff	2 3 4 5 4 3
SEC	Set Carry Bit	C ← 1	—	—	—	—	1	INH	99		2
SEI	Set Interrupt Mask	I ← 1	—	1	—	—	—	INH	9B		2
STA opr STA opr STA opr,X STA opr,X STA ,X	Store Accumulator in Memory	M ← (A)	—	—	↕◊	↕	—	DIR EXT IX2 IX1 IX	B7 C7 D7 E7 F7	dd hh ll ee ff ff	4 5 6 5 4
STOP	Stop Oscillator and Enable IRQ Pin		—	0	—	—	—	INH	8E		2
STX opr STX opr STX opr,X STX opr,X STX ,X	Store Index Register In Memory	M ← (X)	—	—	↕◊	↕	—	DIR EXT IX2 IX1 IX	BF CF DF EF FF	dd hh ll ee ff ff	4 5 6 5 4
SUB #opr SUB opr SUB opr SUB opr,X SUB opr,X SUB ,X	Subtract Memory Byte from Accumulator	A ← (A) − (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A0 B0 C0 D0 E0 F0	ii dd hh ll ee ff ff	2 3 4 5 4 3
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) − 1; Push (PCH) SP ← (SP) − 1; Push (X) SP ← (SP) − 1; Push (A) SP ← (SP) − 1; Push (CCR) SP ← (SP) − 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	—	1	—	—	—	INH	83		10
TAX	Transfer Accumulator to Index Register	X ← (A)	—	—	—	—	—	INH	97		2

Table 6. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
TST *opr* TSTA TSTX TST *opr*,X TST ,X	Test Memory Byte for Negative or Zero	(M) – $00	—	—	↕	↕	—	DIR INH INH IX1 IX	3D 4D 5D 6D 7D	dd ff	4 3 3 5 4
TXA	Transfer Index Register to Accumulator	A ← (X)	—	—	—	—	—	INH	9F		2
WAIT	Stop CPU Clock and Enable Interrupts		—	0 ◊	—	—	—	INH	8F		2

A	Accumulator	*opr*	Operand (one or two bytes)
C	Carry/borrow flag	PC	Program counter
CCR	Condition code register	PCH	Program counter high byte
dd	Direct address of operand	PCL	Program counter low byte
dd rr	Direct address of operand and relative offset of branch instruction	REL	Relative addressing mode
DIR	Direct addressing mode	*rel*	Relative program counter offset byte
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing	rr	Relative program counter offset byte
EXT	Extended addressing mode	SP	Stack pointer
ff	Offset byte in indexed, 8-bit offset addressing	X	Index register
H	Half-carry flag	Z	Zero flag
hh ll	High and low bytes of operand address in extended addressing	#	Immediate value
I	Interrupt mask	∧	Logical AND
ii	Immediate operand byte	∨	Logical OR
IMM	Immediate addressing mode	⊕	Logical EXCLUSIVE OR
INH	Inherent addressing mode	( )	Contents of
IX	Indexed, no offset addressing mode	–( )	Negation (two's complement)
IX1	Indexed, 8-bit offset addressing mode	←	Loaded with
IX2	Indexed, 16-bit offset addressing mode	?	If
M	Memory location	:	Concatenated with
N	Negative flag	↕	Set or cleared
n	Any bit	—	Not affected

Freescale Semiconductor, Inc.

Table 7. Opcode Map

Legend box:

MSB / LSB		
0	BRSET0 3 DIR	5

- MSB of Opcode in Hexadecimal
- Number of Cycles
- Opcode Mnemonic
- Number of Bytes/Addressing Mode
- LSB of Opcode in Hexadecimal

Each cell below: **Mnemonic – Addressing Mode (cycles / bytes)**

LSB \ MSB	0 Bit Manip DIR	1 Bit Manip DIR	2 Branch REL	3 RMW DIR	4 RMW INH	5 RMW INH	6 RMW IX1	7 RMW IX	8 Control INH	9 Control INH	A Reg/Mem IMM	B Reg/Mem DIR	C Reg/Mem EXT	D Reg/Mem IX2	E Reg/Mem IX1	F Reg/Mem IX
0	BRSET0 DIR (5/3)	BSET0 DIR (5/2)	BRA REL (3/2)	NEG DIR (5/2)	NEGA INH (3/1)	NEGX INH (3/1)	NEG IX1 (6/2)	NEG IX (5/1)	RTI INH (9/1)		SUB IMM (2/2)	SUB DIR (3/2)	SUB EXT (4/3)	SUB IX2 (5/3)	SUB IX1 (4/2)	SUB IX (3/1)
1	BRCLR0 DIR (5/3)	BCLR0 DIR (5/2)	BRN REL (3/2)						RTS INH (6/1)		CMP IMM (2/2)	CMP DIR (3/2)	CMP EXT (4/3)	CMP IX2 (5/3)	CMP IX1 (4/2)	CMP IX (3/1)
2	BRSET1 DIR (5/3)	BSET1 DIR (5/2)	BHI REL (3/2)		MUL INH (11/1)						SBC IMM (2/2)	SBC DIR (3/2)	SBC EXT (4/3)	SBC IX2 (5/3)	SBC IX1 (4/2)	SBC IX (3/1)
3	BRCLR1 DIR (5/3)	BCLR1 DIR (5/2)	BLS REL (3/2)	COM DIR (5/2)	COMA INH (3/1)	COMX INH (3/1)	COM IX1 (6/2)	COM IX (5/1)	SWI INH (10/1)		CPX IMM (2/2)	CPX DIR (3/2)	CPX EXT (4/3)	CPX IX2 (5/3)	CPX IX1 (4/2)	CPX IX (3/1)
4	BRSET2 DIR (5/3)	BSET2 DIR (5/2)	BCC REL (3/2)	LSR DIR (5/2)	LSRA INH (3/1)	LSRX INH (3/1)	LSR IX1 (6/2)	LSR IX (5/1)			AND IMM (2/2)	AND DIR (3/2)	AND EXT (4/3)	AND IX2 (5/3)	AND IX1 (4/2)	AND IX (3/1)
5	BRCLR2 DIR (5/3)	BCLR2 DIR (5/2)	BCS/BLO REL (3/2)								BIT IMM (2/2)	BIT DIR (3/2)	BIT EXT (4/3)	BIT IX2 (5/3)	BIT IX1 (4/2)	BIT IX (3/1)
6	BRSET3 DIR (5/3)	BSET3 DIR (5/2)	BNE REL (3/2)	ROR DIR (5/2)	RORA INH (3/1)	RORX INH (3/1)	ROR IX1 (6/2)	ROR IX (5/1)			LDA IMM (2/2)	LDA DIR (3/2)	LDA EXT (4/3)	LDA IX2 (5/3)	LDA IX1 (4/2)	LDA IX (3/1)
7	BRCLR3 DIR (5/3)	BCLR3 DIR (5/2)	BEQ REL (3/2)	ASR DIR (5/2)	ASRA INH (3/1)	ASRX INH (3/1)	ASR IX1 (6/2)	ASR IX (5/1)		TAX INH (2/1)		STA DIR (4/2)	STA EXT (5/3)	STA IX2 (6/3)	STA IX1 (5/2)	STA IX (4/1)
8	BRSET4 DIR (5/3)	BSET4 DIR (5/2)	BHCC REL (3/2)	ASL/LSL DIR (5/2)	ASLA/LSLA INH (3/1)	ASLX/LSLX INH (3/1)	ASL/LSL IX1 (6/2)	ASL/LSL IX (5/1)		CLC INH (2/1)	EOR IMM (2/2)	EOR DIR (3/2)	EOR EXT (4/3)	EOR IX2 (5/3)	EOR IX1 (4/2)	EOR IX (3/1)
9	BRCLR4 DIR (5/3)	BCLR4 DIR (5/2)	BHCS REL (3/2)	ROL DIR (5/2)	ROLA INH (3/1)	ROLX INH (3/1)	ROL IX1 (6/2)	ROL IX (5/1)		SEC INH (2/1)	ADC IMM (2/2)	ADC DIR (3/2)	ADC EXT (4/3)	ADC IX2 (5/3)	ADC IX1 (4/2)	ADC IX (3/1)
A	BRSET5 DIR (5/3)	BSET5 DIR (5/2)	BPL REL (3/2)	DEC DIR (5/2)	DECA INH (3/1)	DECX INH (3/1)	DEC IX1 (6/2)	DEC IX (5/1)		CLI INH (2/1)	ORA IMM (2/2)	ORA DIR (3/2)	ORA EXT (4/3)	ORA IX2 (5/3)	ORA IX1 (4/2)	ORA IX (3/1)
B	BRCLR5 DIR (5/3)	BCLR5 DIR (5/2)	BMI REL (3/2)							SEI INH (2/1)	ADD IMM (2/2)	ADD DIR (3/2)	ADD EXT (4/3)	ADD IX2 (5/3)	ADD IX1 (4/2)	ADD IX (3/1)
C	BRSET6 DIR (5/3)	BSET6 DIR (5/2)	BMC REL (3/2)	INC DIR (5/2)	INCA INH (3/1)	INCX INH (3/1)	INC IX1 (6/2)	INC IX (5/1)		RSP INH (2/1)		JMP DIR (2/2)	JMP EXT (3/3)	JMP IX2 (4/3)	JMP IX1 (3/2)	JMP IX (2/1)
D	BRCLR6 DIR (5/3)	BCLR6 DIR (5/2)	BMS REL (3/2)	TST DIR (4/2)	TSTA INH (3/1)	TSTX INH (3/1)	TST IX1 (5/2)	TST IX (4/1)		NOP INH (2/1)	BSR REL (6/2)	JSR DIR (5/2)	JSR EXT (6/3)	JSR IX2 (7/3)	JSR IX1 (6/2)	JSR IX (5/1)
E	BRSET7 DIR (5/3)	BSET7 DIR (5/2)	BIL REL (3/2)						STOP INH (2/1)		LDX IMM (2/2)	LDX DIR (3/2)	LDX EXT (4/3)	LDX IX2 (5/3)	LDX IX1 (4/2)	LDX IX (3/1)
F	BRCLR7 DIR (5/3)	BCLR7 DIR (5/2)	BIH REL (3/2)	CLR DIR (5/2)	CLRA INH (3/1)	CLRX INH (3/1)	CLR IX1 (6/2)	CLR IX (5/1)	WAIT INH (2/1)	TXA INH (2/1)		STX DIR (4/2)	STX EXT (5/3)	STX IX2 (6/3)	STX IX1 (5/2)	STX IX (4/1)

INH = Inherent   REL = Relative
IMM = Immediate   IX = Indexed, No Offset
DIR = Direct   IX1 = Indexed, 8-Bit Offset
EXT = Extended   IX2 = Indexed, 16-Bit Offset

Freescale Semiconductor, Inc.

Resets, Interrupts and Low Power Modes

Contents

Resets

The MCU can be reset in four ways: by the initial power-on reset function, by an active low input to the RESET pin, by an opcode fetch from an illegal address, and by a COP watchdog timer reset. See Figure 1, below.
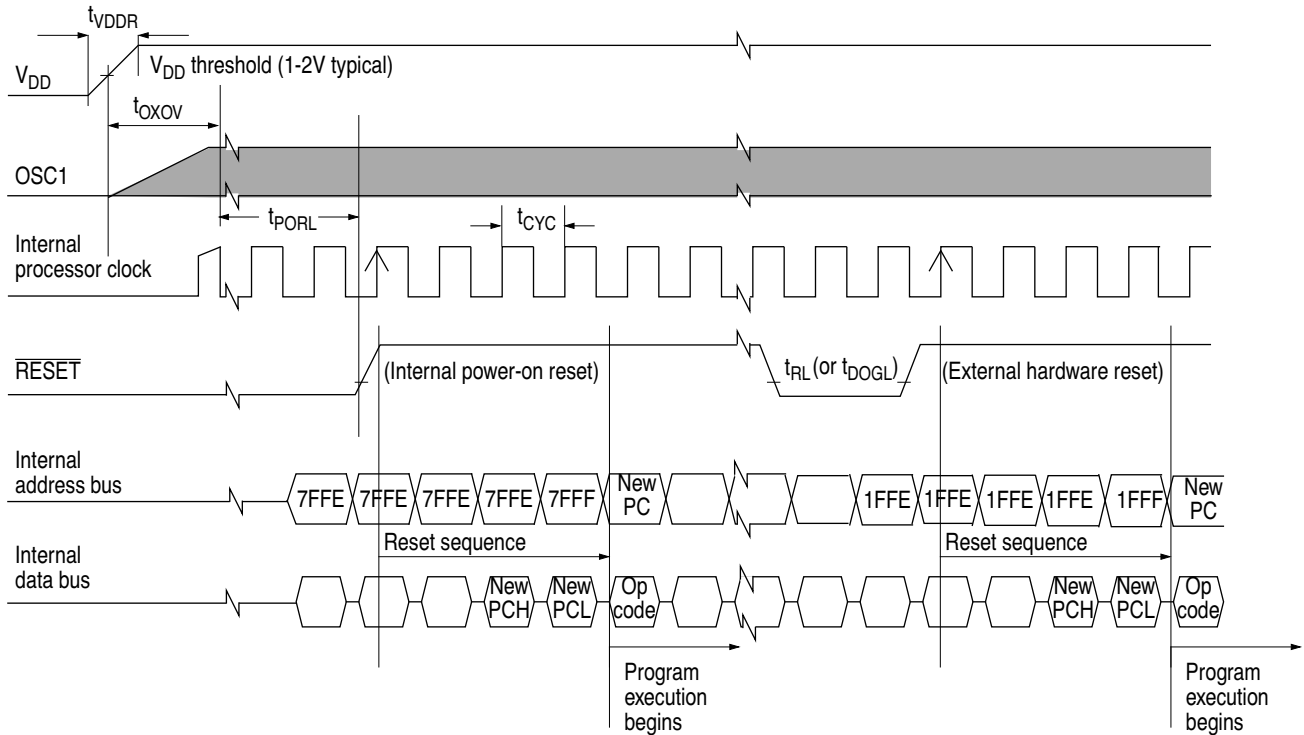
Figure 1. Power-on reset and RESET

Power-on reset

A power-on reset occurs when a positive transition is detected on VDD. The power-on reset function is strictly for power turn-on conditions and should not be used to detect drops in the power supply voltage. The power-on circuitry provides a stabilisation delay (t_{PORL}) from when the oscillator becomes active. If the external RESET pin is low at the end of this delay then the processor remains in the reset state until RESET goes high. The user must ensure that the voltage on VDD has risen to a point where the MCU can operate properly by the time t_{PORL} has elapsed. If there is doubt, the external RESET pin should remain low until the voltage on VDD has reached the specified minimum operating voltage. This may be accomplished by connecting an external RC-circuit to this pin to generate a power-on reset (POR). In this case, the time constant must be great enough (at least 100 ms) to allow the oscillator circuit to stabilise.

RESET pin

When the oscillator is running in a stable state, the MCU is reset when a logic zero is applied to the RESET input for a minimum period of 1.5 machine cycles (t_{CYC}). This pin contains an internal Schmitt Trigger as part of its input to improve noise immunity.

Illegal address reset

When an opcode fetch occurs from an address which is not part of the RAM ($0050 – $00FF) or of the NVM ($0F00 – $1F00 and $1FF0 – $1FFF) then the device is automatically reset.

Computer operating properly (COP) reset

The MCU contains a watchdog timer that automatically times out if not reset (cleared) within a specific time by a program reset sequence. If the COP watchdog timer is allowed to time-out, an internal reset is generated to reset the MCU. Because the internal reset signal is used, the MCU comes out of a COP reset in the same operating mode it was in when the COP time-out was generated.

The COP reset function is enabled or disabled by a mask option.

Refer to **Computer operating properly (COP) watchdog timer** for more information on the COP watchdog timer.

Interrupts

The MCU can be interrupted by five different sources, four maskable hardware interrupts and one non-maskable software interrupt:

- MCAN

- Wired-OR function on ports A and B

- Core timer

- 16-bit programmable timer

- Software Interrupt instruction (SWI)

Interrupts cause the processor to save the register contents on the stack and to set the interrupt mask (I-bit) to prevent additional interrupts. The

RTI instruction (return from interrupt) causes the register contents to be recovered from the stack and normal processing to resume.

Unlike reset, hardware interrupts do not cause the current instruction execution to be halted, but are considered pending until the current instruction is complete. The current instruction is the one already fetched and being operated on. When the current instruction is complete, the processor checks all pending hardware interrupts. If interrupts are not masked (CCR I-bit clear) and the corresponding interrupt enable bit is set, the processor proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If both an external interrupt and a timer interrupt are pending after an instruction execution, the external interrupt is serviced first.

Table 1 shows the relative priority of all the possible interrupt sources. Figure 2 shows the interrupt processing flow.

Table 1. Interrupt priorities

Source	Register	Flags	Vector address	Priority
Reset	—	—	$1FFE, $1FFF	highest
Software interrupt (SWI)	—	—	$1FFC, $1FFD	↑
CAN interrupt (CIRQ)	CINT	WIF, OIF, EIF, TIF, RIF	$1FFA, $1FFB	
Core timer (CTIMER)	CTCSR	CTOF, RTIF	$1FF8, $1FF9	
Wired-OR interrupt	PCR	WOIF	$1FF6, $1FF7	
Programmable timer	TSR	ICF, OFC, TOF	$1FF4, $1FF5	lowest

Non-maskable software interrupt (SWI)

The software interrupt (SWI) is an executable instruction and a non-maskable interrupt: it is executed regardless of the state of the I-bit in the CCR. If the I-bit is zero (interrupts enabled), SWI is executed after interrupts that were pending when the SWI was fetched, but before interrupts generated after the SWI was fetched. The SWI interrupt service routine address is specified by the contents of memory locations $1FFC and $1FFD.

Maskable hardware interrupts

If the interrupt mask bit (I-bit) of the CCR is set, all maskable interrupts (internal and external) are masked. Clearing the I-bit allows interrupt processing to occur.

NOTE: *The internal interrupt latch is cleared in the first part of the interrupt service routine; therefore, one external interrupt pulse could be latched and serviced as soon as the I-bit is cleared.*

MCAN interrupt (CIRQ)

Several sources can trigger a CIRQ. The MCAN interrupt register at $0023 is used to identify the source. Each CIRQ source can be individually enabled (except the wake-up interrupt, which is always enabled) by different bits of the MCAN control register at $0020.

The CIRQ sources are (see also **MCAN interrupt register (CINT)**):

Receive IRQ: this signals successful reception of a complete message,

Transmit IRQ: this signals successful transmission of a complete message,

Error IRQ: this is set when either the error status or bus status bits in the MCAN status register change state (see **MCAN status register (CSTAT)**),

Data Overrun: an incoming message on the bus cannot be received because both receive buffers are tied up,

Wake-up IRQ: this signals activity on the bus while the MCAN is in SLEEP mode.

CIRQ interrupts are serviced by the routine located at the address specified by the contents of $1FFA and $1FFB.

Wired-OR interrupt (WOI)

An external WOI capability is provided on all I/O pins. When WOI is enabled on a given pin (refer to **Input/output programming** and **Port A**), an external interrupt is requested when this pin is pulled high. The interrupt request is latched immediately following the rising edge of the external WOI interrupt signal. It is then synchronised internally and serviced by the interrupt routine whose start address is contained in memory locations $1FF6 and $1FF7. The address of the latch bit for the

Freescale Semiconductor, Inc.

WOI interrupt is bit 5 of the port configuration register ($03). This latch is set by the WOI, and is cleared by writing a zero to the bit. A WOI will cause the MPU to exit from STOP mode.

Real time and core timer (CTIMER) interrupts

There are two different core timer interrupt flags that cause a CTIMER interrupt whenever an interrupt is enabled and its flag becomes set, namely RTIF and CTOF. The interrupt flags and enable bits are located in the CTIMER control and status register (CTCSR). These interrupts will vector to the same interrupt service routine, whose start address is contained in memory locations $1FF8 and $1FF9 (see **Core timer control and status register (CTCSR)** and Figure 1).

To make use of the real time interrupt the RTIE bit must first be set. The RTIF bit will then be set after the specified number of counts.

To make use of the core timer overflow interrupt the CTOFE bit must first be set. The CTOF bit will then be set when the core timer counter register overflows from $FF to $00.

Programmable 16-bit timer interrupt

There are three different timer interrupt flags (ICF, OCF, TOF) that cause a timer interrupt whenever they are set and enabled. The timer interrupt enable bits (ICIE, OCIE, TOIE) are located in the timer control register (TCR) and the timer interrupt flag is located in the timer status register (TSR). All three interrupts will vector to the same service routine, whose start address is contained in memory locations $1FF4 and $1FF5.

Hardware controlled interrupt sequence

The following three functions (RESET, STOP, and WAIT) are not in the strictest sense interrupts. However, they are acted upon in a similar manner. Flowcharts for STOP and WAIT are shown in Figure 3.

RESET: A reset condition causes the program to vector to its starting address, which is contained in memory locations $1FFE (MSB) and $1FFF (LSB). The I-bit in the condition code register is also set, to disable interrupts.

STOP: The STOP instruction causes the oscillator to be turned off and the processor to 'sleep' until an external interrupt

MC68HC05X4

(CIRQ or WOI) occurs or the device is reset. However the processor will only stop the oscillator if the MCAN is in 'sleep' mode. Otherwise only the MPU clocks will be turned off and the MCAN will remain active.

WAIT: The WAIT instruction causes all processor clocks to stop, but leaves the timer clocks running. This 'rest' state of the processor can be cleared by reset, an external interrupt (CIRQ or WOI), or a timer interrupt (core or 16-bit). There are no special WAIT vectors for these interrupts.

Low power modes

STOP

The STOP instruction places the MCU in its lowest power consumption mode. In STOP mode, the internal oscillator is turned off (providing the MCAN is asleep, see **Sleep mode**), halting all internal processing, including timer (and COP watchdog timer) operation.

During the STOP mode, the core timer interrupt flags (CTOF and RTIF) and interrupt enable bits (CTOFE and RTIE) in the CTCSR, as well as the timer flags in register TSR, and interrupt enable bits in register TCR, are cleared by internal hardware. This removes any pending timer interrupt requests and disables any further timer interrupts. The timer prescaler is cleared. The I-bit in the CCR is cleared to enable external interrupts. All other registers, the remaining bits in the CTCSR and memory contents remain unaltered. All input/output lines remain unchanged. The processor can be brought out of the STOP mode only by an external interrupt (CIRQ or WOI) or a reset.

WAIT

The WAIT instruction places the MCU in a low-power consumption mode, though it consumes more power than in STOP mode. All CPU action is suspended, but the timers (core and 16-bit) remain active. An interrupt from either of the timers, if enabled, will cause the MCU to exit the WAIT mode.

During the WAIT mode, the I-bit in the CCR is cleared to enable interrupts. All other registers, memory, and input/output lines remain in their previous state. The Core Timer may be enabled to allow a periodic exit from the WAIT mode. See **Core timer during WAIT**.

Data retention mode

The contents of the RAM are retained at supply voltages as low as 2.0 Vdc. This is called the data retention mode, in which data is maintained but the device is not guaranteed to operate.

For lowest power consumption in data retention mode the device should be put into STOP mode before reducing the supply voltage, to ensure that all the clocks are stopped. If the device is not in STOP mode then it is recommended that RESET be held low whilst the power supply is outwith the normal operating range, to ensure that processing is suspended in an orderly manner.

– Recovery from data retention mode, after the power supply has been restored, is by an external interrupt, or by pulling the RESET line high

```
                        ( Reset )
                            │
                            ▼
   Yes               ◇ Is I-bit set? ◇
  ◄────────────────
                            │ No
                            ▼
                  ◇ CIRQ        ◇  Yes    ┌─────────────────┐
                  ◇ interrupt?  ◇────────►│ Clear relevant IRQ │
                            │              │ request latch      │
                            │ No           └─────────────────┘
                            ▼                        │
                  ◇ Core timer? ◇  Yes               ▼
                            │                ┌─────────────────┐
                            │ No             │ Stack            │
                            ▼                │ PC, X, A, CC     │
                  ◇ WOI         ◇  Yes       └─────────────────┘
                  ◇ interrupt?  ◇                    │
                            │                        ▼
                            │ No             ┌─────────────────┐
                            ▼                │ Set I-bit        │
                  ◇ 16-bit timer? ◇  Yes     └─────────────────┘
                            │                        │
                            │ No                     ▼
                            ▼                ┌──────────────────────────┐
                  ┌─────────────────┐       │ Load PC from:            │
                  │ Fetch next       │       │ SWI:        $1FFC-$1FFD  │
                  │ instruction      │       │ CIRQ:       $1FFA-$1FFB  │
                  └─────────────────┘       │ Core timer:$1FF8-$1FF9   │
                            │                │ WOI:        $1FF6-$1FF7  │
                            ▼                │ 16-bit timer:$1FF4-$1FF5 │
                  ◇ SWI         ◇  Yes       └──────────────────────────┘
                  ◇ instruction? ◇────────────────────────────────►
                            │ No
                            ▼
                  ◇ RTI         ◇  Yes    ┌─────────────────┐
                  ◇ instruction? ◇───────►│ Restore registers │──────►
                            │              │ from stack:       │
                            │ No           │ CC, A, X, PC      │
                            ▼              └─────────────────┘
                  ┌─────────────────┐
                  │ Execute          │
                  │ instruction      │
                  └─────────────────┘
```

Figure 2. Reset flow chart

Freescale Semiconductor, Inc.
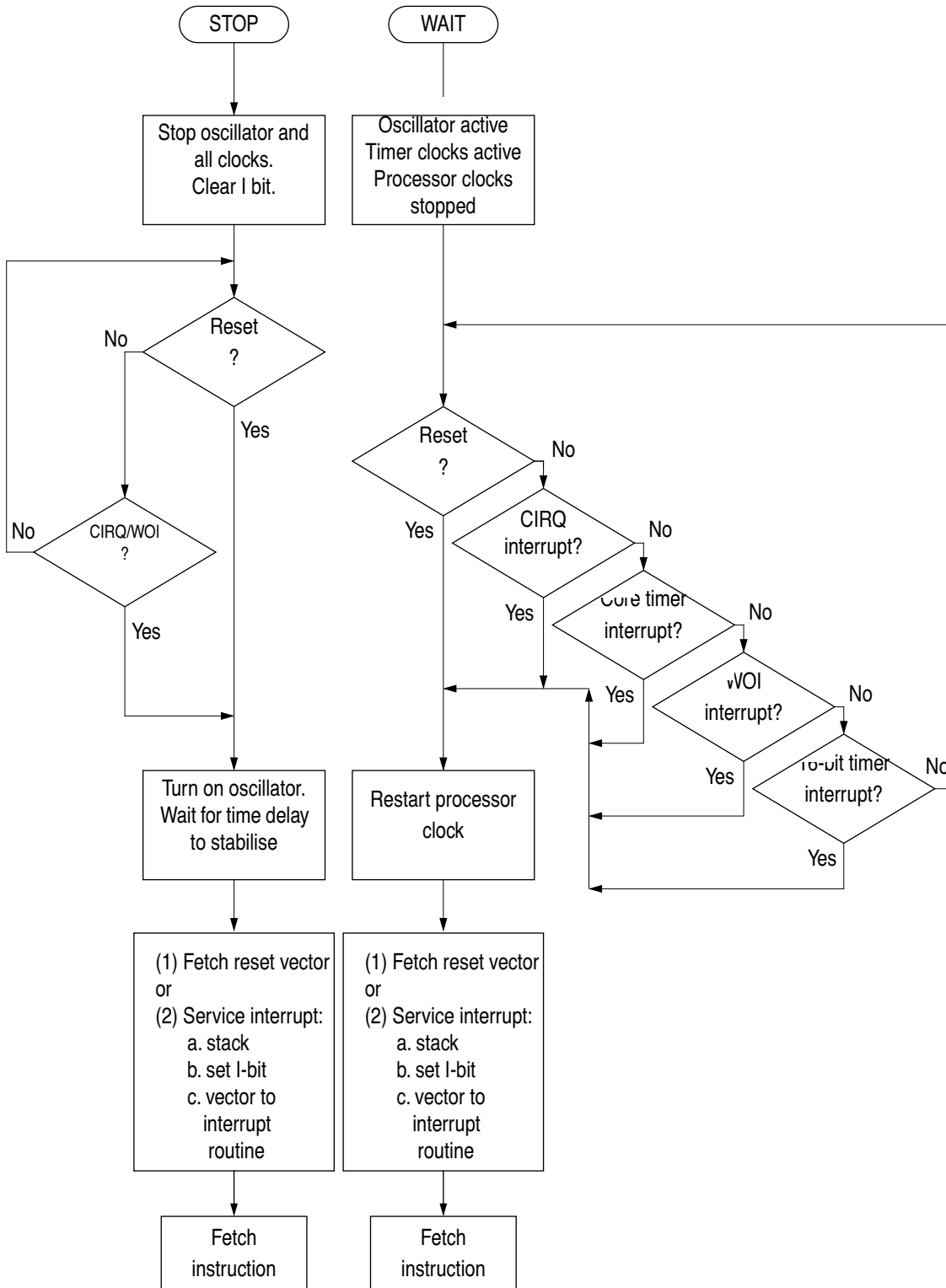
Figure 3. STOP and WAIT flow charts

Memory

Contents

Memory map

The MC68HC05X4 has an 8K byte memory map consisting of MCAN control registers, user ROM *or EPROM*, user RAM, bootloader ROM, and I/O (as illustrated in Figure 1).

RAM

The user RAM consists of 176 bytes of memory space shared with a 64 byte stack area. The stack begins at address $00FF. The stack pointer can access 64 bytes of RAM in the range $00C0 to $00FF.

NOTE: *Using the stack area for data storage or temporary work locations requires care, to prevent the data from being overwritten due to stacking from an interrupt or subroutine call.*

Non-volatile memory (NVM)

The user NVM consists of 4096 bytes of ROM (MC68HC05X4) *or EPROM (MC68HC705X4)* from $0F00 to $1EFF and 16 bytes of user vectors from $1FF0 to $1FFF.

The NVM has two modes of operation: single chip and bootloader (see **Modes of Operation and Pin Descriptions**).

MC68HC05X4

Address	Block
$0000	I/O (32 bytes)
$0020	MCAN control registers (30 bytes)
$003E	Reserved
$0050	RAM I (176 bytes)
$00C0	Stack
$0100	Reserved
$0F00	User ROM (4096 bytes)
$1F00	Reserved
$1F01	Bootloader ROM (239 bytes)
$1FF0–3	Reserved
$1FF4–5	16-bit timer
$1FF6–7	WOI
$1FF8–9	Core timer
$1FFA–B	CIRQ
$1FFC–D	SWI
$1FFE–F	Reset

MC68HC705X4

Block
User EPROM (4096 bytes)
Mask option register
Bootloader ROM (239 bytes)
Reserved
16-bit timer
WOI
Core timer
CIRQ
SWI
Reset

Registers

Register	Address	Group
Port A data	$0000*	
Port B data	$0001	
	$0002	
Port configuration	$0003	
Port A data direction	$0004*	
Port B data direction	$0005	
	$0006–$0007	
Core timer control and status	$0008	
Core timer counter	$0009	
	$000A–$0011	
16-bit timer control	$0012	
16-bit timer status	$0013	
Input capture high	$0014	
Input capture low	$0015	
Output compare high	$0016	
Output compare low	$0017	
Counter high	$0018	
Counter low	$0019	
Alternate counter high	$001A	
Alternate counter low	$001B	
EPROM programming	$001C	
	$001D–$001E	
MCAN control	$0020	
MCAN command	$0021	
MCAN status	$0022	
MCAN interrupt	$0023	MCAN control and status registers
MCAN acceptance code	$0024	
MCAN acceptance mask	$0025	
Bus timing 0	$0026	
Bus timing 1	$0027	
Output control	$0028	
	$0029	
Transmit buffer identifier	$002A	
RTR bit, data length code	$002B	
Transmit data segment 1	$002C	
Transmit data segment 2	$002D	MCAN transmit registers
Transmit data segment 3	$002E	
Transmit data segment 4	$002F	
Transmit data segment 5	$0030	
Transmit data segment 6	$0031	
Transmit data segment 7	$0032	
Transmit data segment 8	$0033	
Receive buffer identifier	$0034	
RTR bit, data length code	$0035	
Receive data segment 1	$0036	
Receive data segment 2	$0037	
Receive data segment 3	$0038	MCAN receive registers
Receive data segment 4	$0039	
Receive data segment 5	$003A	
Receive data segment 6	$003B	
Receive data segment 7	$003C	
Receive data segment 8	$003D	

Reserved

* If AWPS = 1, then the port A WOI enable and port A pull-down enable registers are available at addresses $00 and $04 respectively. They are both cleared on reset.

Figure 1. Memory map of the MC68HC05X4 and the MC68HC705X4

MCU registers

Table 1. MC68HC05X4 and MC68HC705X4 register assignments

Register Name	Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	State on Reset
Port A data/WOI enable (PADAT)	$0000									Undefined/ cleared
Port B data (PBDAT)	$0001									Undefined
(Reserved)										
Port configuration (PCR)	$0003			WOIF	TIMEN	CAF	BPDE	BWIE	AWPS	- - 00 0000
Port A DDR/ Pull-down enable (PADDDR)	$0004									0000 0000
Port B DDR (PBDDR)	$0005									0000 0000
(Reserved)										
Core timer control & status (CTCSR)	$0008	CTOF	RTIF	CTOFE	RTIE	0	0	RT1	RT0	uu00 0011
Core timer counter (CTCR)	$0009									0000 0000
(Reserved)										
Timer control (TCR)	$0012	ICIE	OCIE	TOIE	0	0	0	IEDG	OLVL	0000 00u0
Timer status (TSR)	$0013	ICF	OCF	TOF	0	0	0	0	0	uuu0 0000
Input capture high	$0014	(bit 15)							(bit 8)	Undefined
Input capture low	$0015									Undefined
Output compare high	$0016	(bit 15)							(bit 8)	Undefined
Output compare low	$0017									Undefined
Counter high	$0018	(bit 15)							(bit 8)	1111 1111
Counter low	$0019									1111 1100
Alternate counter high	$001A	(bit 15)							(bit 8)	1111 1111
Alternate counter low	$001B									1111 1100
EPROM programming (EPROG)	$001C	0	0	0	0	0	ELAT	0	EPGM	0000 0000

u = Undefined

- = Unimplemented

NOTE: *Shaded areas represent either unimplemented bits or reserved bits where stated.*

Freescale Semiconductor, Inc.

MCAN registers

Table 2. MCAN register outline

Register name	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Control (CCNTRL)	$0020	MODE	SPD		OIE	EIE	TIE	RIE	RR	0u - u uuu1
Command (CCOM)	$0021	RX0	RX1	COMPSEL	SLEEP	COS	RRB	AT	TR	00u0 0000
Status (CSTAT)	$0022	BS	ES	TS	RS	TCS	TBA	DO	RBS	uu00 1100
Interrupt (CINT)	$0023				WIF	OIF	EIF	TIF	RIF	- - - 0 0000
Acceptance code (CACC)[1]	$0024	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Undefined
Acceptance mask (CACM)[1]	$0025	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	Undefined
Bus timing 0 (CBT0)[1]	$0026	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Undefined
Bus timing 1 (CBT1)[1]	$0027	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	Undefined
Output control (COCNTRL)[1]	$0028	OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCM1	OCM0	Undefined
(reserved)	$0029									
Transmit buffer identifier (TBI)	$002A	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Undefined
RTR-bit, data length code (TRTDL)	$002B	ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Undefined
Transmit data segment 1 (TDS1)	$002C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 2 (TDS2)	$002D	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 3 (TDS3)	$002E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 4 (TDS4)	$002F	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 5 (TDS5)	$0030	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 6 (TDS6))	$0031	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 7 (TDS7)	$0032	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Transmit data segment 8 (TDS8)	$0033	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive buffer identifier (RBI)	$0034	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Undefined

Table 2. MCAN register outline

Register name	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
RTR-bit, data length code (RRTDL)	$0035	ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Undefined
Receive data segment 1 (RDS1)	$0036	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 2 (RDS2)	$0037	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 3 (RDS3)	$0038	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 4 (RDS4)	$0039	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 5 (RDS5)	$003A	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 6 (RDS6)	$003B	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 7 (RDS7)	$003C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined
Receive data segment 8 (RDS8)	$003D	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined

1. These registers can only be accessed when the reset request bit in the control register is set.

Freescale Semiconductor, Inc.

Parallel input/output ports

Contents

Introduction

In single chip mode there are 16 lines arranged as two 8-bit I/O ports. The I/O ports are programmable as either inputs or outputs under software control of the data direction registers. Wired-OR Interrupt capability (refer to **Resets**) and/or a pull-down device can be activated under software control on each I/O pin.

To avoid glitches on the output pins data should be written to the I/O port data register before setting the pin to output mode, by writing a '1' to the corresponding data direction register.

Freescale Semiconductor, Inc.

Input/output programming

Bi-directional port lines may be programmed as inputs or outputs under software control. The direction of each pin is determined by the state of the corresponding bit in the port data direction register (DDR). Each port has an associated DDR. Any I/O port pin is configured as an output if its corresponding DDR bit is set. A pin is configured as an input if its corresponding DDR bit is cleared.

At power-on or reset, all DDRs are cleared, thus configuring all port pins as inputs. The data direction registers can be written to or read by the processor. During the programmed output state, a read of the data register actually reads the value of the output data latch and not the I/O pin. Refer to Figure 1 and Table 1.

DDRn	DATA	I/O pin
1	0	0
1	1	1
0	0	tristate
0	1	tristate

Output applies to the first two rows; *Input* applies to the last two rows.
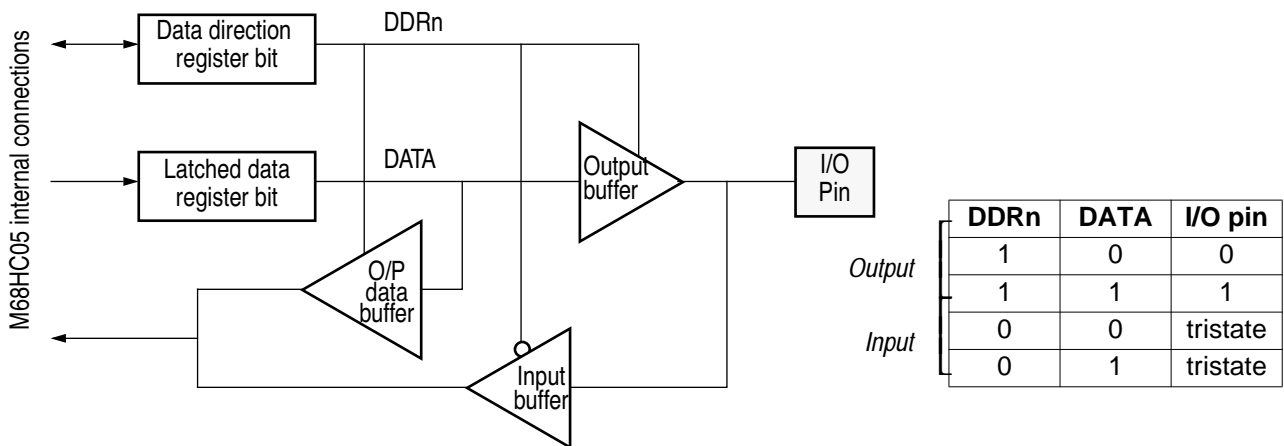
Figure 1. Standard I/O port structure

Port A

Port A is an 8-bit bi-directional port. The port A data register is at $0000 and the data direction register (DDR) is at $0004. Reset does not affect the data register, but clears the data direction register, thereby returning the ports to inputs. Writing a '1' to a DDR bit sets the corresponding port pin to output mode.

MC68HC05X4 Rev 1.0

2-ports

Parallel input/output ports
For More Information On This Product,
Go to: www.freescale.com

WOI enable and pull-down enable registers control each pin individually; these registers are memory mapped to the same addresses as port A data register and data direction register (i.e $00 and $04). When the AWPS bit in the port configuration register (PCR) is set, port A WOI enable and pull-down enable registers are selected instead of port A data and DDR registers. Data and DDR registers are selected when the AWPS bit in PCR is cleared; this is also cleared at reset. Note that WOI can be programmed independently of the DDR contents (input or output).

Port B

Port B is an 8-bit bi-directional port. The port B data register is at $0001 and the data direction register (DDR) is at $0005. Reset does not affect the data register, but clears the data direction register, thereby returning the ports to inputs. Writing a '1' to a DDR bit sets the corresponding port bit to output mode. On port B a single WOI enable bit and a single pull-down enable bit are provided to control all 8 bits together. These two bits are respectively bit 1 and bit 2 of port configuration register (PCR). When bit 1 of the PCR is set, WOI is enabled only on those port B pins that have been programmed as inputs. Note that port B shares pin 1 of the device with the TCMP function of the 16-bit programmable timer. If bit 4 of the PCR is set then pin 1 is TCMP and the wired-OR functions are disabled on this pin. If bit 4 in the PCR is '0' then pin 1 is PB7. On reset, bits 0–4 of the PCR are cleared.

Table 1. I/O pin functions

R/W[†]	DDR	I/O pin function
0	0	The I/O pin is in input mode. Data is written into the output data latch.
0	1	Data is written into the output data latch, and output to the I/O pin.
1	0	The state of the I/O pin is read.
1	1	The I/O pin is in output mode. The output data latch is read.

† Note that R/W is an internal signal, not available to the user.

Port registers

The following sections explain in detail the individual bits in the data and control registers associated with the ports.

Port A data register (PADR)

Each bit can be configured as input or output via the corresponding data direction bit in the port A DDR.

Reset does not affect the state of this register.

NOTE: *If the AWPS bit in the PCR is set then this location becomes the port A wired-OR interrupt enable register. Writing a '1' to any bit enables WOI on the corresponding port A line.*

Port B data register (PBDR)

Each bit can be configured as input or output via the corresponding data direction bit in the port B DDR.

Reset does not affect the state of this register.

Port configuration register (PCR)

Address: $0005

	Bit 7	6	5	4	3	2	1	Bit 0
			WOIF	TIMEN	CAF	BPDE0	BWE	AWPS
Reset:	-	-	0	0	0	0	0	0

Figure 2. Port Configuration Register (PCR)

WOIF — Wired-OR interrupt flag
  1 = Indicates that a wired-OR interrupt has been received. A CPU interrupt request is generated if WOIE is set on port A or port B.
  0 = The flag is cleared by writing a '0' to it.

TIMEN — Timer enable

CAF — Indicates when MCAN is asleep

BPDE — Port B pull-down enable
1 = Enables pull-down on port B.
0 = Disables pull-down on port B.

BWE — Port B WOI enable
1 = Enables wired-OR interrupt on port B.
0 = Disables wired-OR interrupt on port B.

AWPS — Port A WOI and pull-down select

Addresses $00 and $04 in the memory map are shared by two pairs of registers. The state of the AWPS bit determines which pair of registers are accessible at any time. When AWPS is clear the port A data register is found at $00, and the port A data direction register at $04. When AWPS is set, $00 becomes the port A WOI enable register, and $04 the port A pull-down enable register. See **Input/output programming**.

1 = The port A WOI enable and pull-down enable registers are accessible.
0 = The port A data and data direction registers are accessible.

Port A data direction register (PADDR)

Writing a '1' to any bit configures the corresponding bit in the port A data register as an output; conversely, writing any bit to '0' configures the corresponding port A bit as an input.

Reset clears this register.

NOTE: *If the AWPS bit in the PCR is set then this location becomes the port A pull-down enable register. Writing a '1' to any bit enables the pull-down on the corresponding port A line.*

Port B data direction register (PBDDR)

Writing a '1' to any bit configures the corresponding bit in the port B data register as an output. Conversely, writing any bit to '0' configures the corresponding port B bit as an input.

Reset clears this register.

5-ports

MC68HC05X4 Rev 1.0

Parallel input/output ports
For More Information On This Product,
Go to: www.freescale.com

Parallel input/output ports

Motorola CAN

Contents

Introduction

The MCAN includes all hardware modules necessary to implement the CAN transfer layer, which represents the kernel of the CAN bus protocol as defined by BOSCH GmbH, the originators of the CAN specification. For full details of the CAN protocol please refer to the published specifications.

Up to the message level, the MCAN is totally compatible with the full CAN implementation. Functional differences are related to the object layer only. Whereas a full CAN controller provides dedicated hardware for handling a set of messages, the MCAN is restricted to receiving and/or transmitting messages on a message by message basis.

The MCAN will never initiate an overload frame. If the MCAN starts to receive a valid message (one that passes the acceptance filter) and there is no receive buffer available for it then the overrun flag in the CPU status register will be set. The MCAN will respond to overload frames generated by other CAN nodes, as required by the CAN protocol. A summary of all the MCAN frame formats is given in Figure 2 for reference. A diagram of the major blocks of the MCAN is shown in Figure 1.

MC68HC05X4
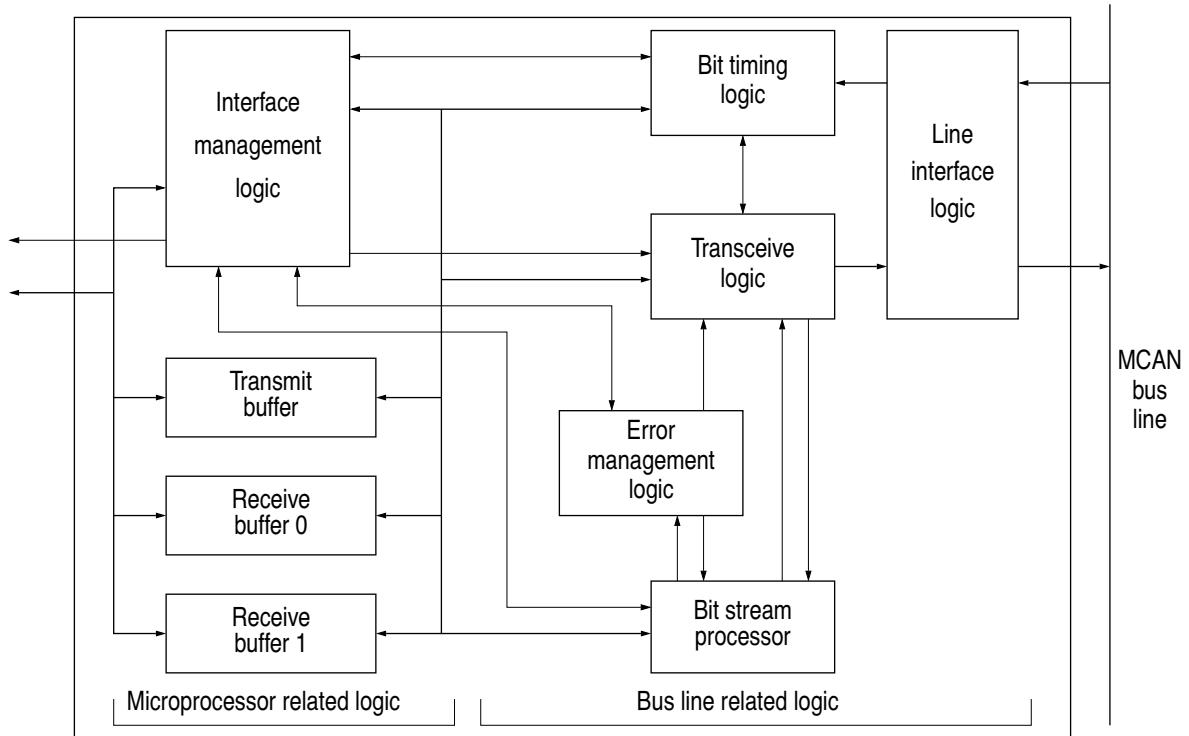
Figure 1. MCAN block diagram

Figure 2. MCAN frame formats

Freescale Semiconductor, Inc.

Note: An error frame can start anywhere in the middle of a frame.

Note: INT = Intermission
Suspend transmission is only for error passive nodes.

Note: An overload frame can only start at the end of a frame.
Maximum echo of overload flag is one bit.

Error frame

Data frame or remote frame

Inter-frame space or overload frame

| 6 | ≤6 | 8 |
| Error flag | Echo error flag | Error delimiter |

d d d d d d   d d r r r r r r   r r r r r r r r

Inter-frame space

Any frame

Data frame or remote frame

| 3 | 8 | | |
| INT | Suspend transmit | Bus idle | Start of frame |

r r r r r r r r r r r r r r r r r   r r r d

Overload frame

End of frame or error delimiter or overload delimiter

Inter-frame space or error frame

| 6 | 8 |
| Overload flag | Overload delimiter |

d d d d d d   r r r r r r r r r

Figure 2. MCAN frame formats (Continued)

TBF – Transmit buffer

The transmit buffer is an interface between the CPU and the bit stream processor (BSP) and is able to store a complete message. The buffer is written by the CPU and read by the BSP. The CPU may access this buffer whenever transmit buffer access is set to released. On requesting a transmission (by setting transmission request in the MCAN command register to present) transmit buffer access is set to locked, giving the BSP exclusive access to this buffer. The transmit buffer is released after the message transfer has been completed or aborted.

The TBF is 10 bytes long and holds the identifier (1 byte), the control field (1 byte) and the data field (maximum length 8 bytes). The buffer is implemented as a single-ported RAM, with mutually exclusive access by the CPU and the BSP.

RBF – Receive buffer

The receive buffer is an interface between the BSP and the CPU and stores a message received from the bus line. Once filled by the BSP and allocated to the CPU (by the IML), the receive buffer cannot be used to store subsequent received messages until the CPU has acknowledged the reading of the buffer's contents. Thus, unless the CPU releases a receive buffer within a protocol defined time frame, future messages to be received may be lost.

To reduce the requirements on the CPU, two receive buffers (RBF0 and RBF1) are implemented. While one receive buffer is allocated to the CPU, the BSP may write to the other buffer. RBF0 and RBF1 are each 10 bytes long and hold the identifier (1 byte), the control field (1 byte) and the data field (maximum length 8 bytes). The buffers are implemented as single-ported RAMs with mutually exclusive access from the CPU and the BSP. The BSP signals the CPU to read the receive buffer only when the message being received has an identifier that passes the acceptance filter. Note that a message being transmitted will be automatically written to the receive buffer. This is because it cannot be

MC68HC05X4

known, until after the first byte has been stored, whether or not the transmitting node will lose arbitration to another node.

Interface to the MC68HC05X4 CPU

The MCAN handles all the communication transactions flowing across the serial bus. For example, the CPU merely places a message to be transmitted into the transmit buffer and sets the TR bit. The MCAN will begin transmitting the message when it has determined that the bus is idle. In the event of a transmission error, the MCAN will initiate a repeated transmission automatically.

In a similar manner, the CPU module is notified that a message has been received only if it was error free. If any error occurs, the MCAN signals the error within the CAN protocol without CPU intervention.

The MCAN within the MC68HC05X4 is controlled using a block of 30 registers. This comprises 10 control registers, 10 Transmit buffer registers and 10 receive buffer registers. These registers are memory mapped between $20 and $3D (see Figure 3).

NOTE: *There is an offset of $20 between the MC68HC05X4 addresses and the MCAN internal addresses, i.e. MCAN addresses $00 to $1D, as defined in the BOSCH CAN specification, are mapped to MC68HC05X4 addresses $20 to $3D.*

MCAN register blocks   MCAN registers

$0020	
MCAN control registers 10 bytes	
$0029	
$002A	
MCAN transmit buffer 10 bytes	
$0033	
$0034	
MCAN receive buffer 10 bytes	
$003D	

Register	Address
Control register	$0020
Command register	$0021
Status register	$0022
Interrupt register	$0023
Acceptance code register	$0024
Acceptance mask register	$0025
Bus timing register 1	$0026
Bus timing register 2	$0027
Output control register	$0028
Test register	$0029
Identifier	$002A
RTR-bit, data length code	$002B
Data segment byte 1	$002C
Data segment byte 2	$002D
Data segment byte 3	$002E
Data segment byte 4	$002F
Data segment byte 5	$0030
Data segment byte 6	$0031
Data segment byte 7	$0032
Data segment byte 8	$0033
Identifier	$0034
RTR-bit, data length code	$0035
Data segment byte 1	$0036
Data segment byte 2	$0037
Data segment byte 3	$0038
Data segment byte 4	$0039
Data segment byte 5	$003A
Data segment byte 6	$003B
Data segment byte 7	$003C
Data segment byte 8	$003D

Figure 3. MCAN module memory map

MCAN control register (CCNTRL)

This register may be read or written to by the MCU; only the RR bit is affected by the MCAN.

Address: $0020

	Bit 7	6	5	4	3	2	1	Bit 0
	MODE	SPD		OIE	EIE	TIE	RIE	RR
External Reset:	0	u	-	u	u	u	u	1
Reset: with RR bit set	0	u	-	u	u	u	u	1

Figure 4. MCAN Control Register (CCNTRL)

NOTE: *Only the RR bit in this register can be written when the RR bit is set.*

MODE — Undefined mode

This bit must never be set by the CPU as this would result in the transmit and receive buffers being mapped out of memory. The bit is cleared on reset, and should be left in this state for normal operation.

SPD — Speed mode
> 1 = Slow – Bus line transitions from both 'recessive' to 'dominant' and from 'dominant' to 'recessive' will be used for resynchronization.
> 0 = Fast – Only transitions from 'recessive' to 'dominant' will be used for resynchronization.

OIE — Overrun interrupt enable
> 1 = Enabled – The CPU will get an interrupt request whenever the Overrun Status bit gets set.
> 0 = Disabled – The CPU will get no overrun interrupt request.

EIE — Error interrupt enable
> 1 = Enabled – The CPU will get an interrupt request whenever the error status or bus status bits in the CSTAT register change.
> 0 = Disabled – The CPU will get no error interrupt request.

TIE — Transmit interrupt enable
> 1 = Enabled – The CPU will get an interrupt request whenever a message has been successfully transmitted, or when the transmit buffer is accessible again following an ABORT command.
> 0 = Disabled – The CPU will get no transmit interrupt request.

NOTE: *When setting TIE while TCS or TBA is set, no interrupt will be requested. Successful transmission must occur after setting the TIE bit to get an interrupt flag.*

RIE — Receive interrupt enable
    1 = Enabled – The CPU will get an interrupt request whenever a message has been received free of errors.
    0 = Disabled – The CPU will get no receive interrupt request.

RR — Reset request

When the MCAN detects that RR has been set it aborts the current transmission or reception of a message and enters the reset state. A reset request may be generated by either an external reset or by the CPU or by the MCAN. The RR bit can be cleared only by the CPU. After the RR bit has been cleared, the MCAN will start normal operation in one of two ways. If RR was generated by an external reset or by the CPU, then the MCAN starts normal operation after the first occurrence of 11 recessive bits. The MCAN module is not synchronized to the bus until the 11 recessive bits have been received. When the SLEEP bit is set during this non synchronous state, an immediate wake-up wil be generated by the MCAN module. If, however, the RR was generated by the MCAN due to the BS bit being set (see **MCAN status register (CSTAT)**) the MCAN waits for 128 occurrences of 11 recessive bits before starting normal operation.

A reset request should not be generated by the CPU during a message transmission. Ensure that a message is not being transmitted as follows:

    if TCS in CSTAT is clear – set AT in CCOM (use STA or STX), read CSTAT.

    if TS in CSTAT is set – wait until TS is clear.

Note that a CPU-generated reset request does not change the values in the transmit and receive error counters.
    1 = Present – MCAN will be reset.
    0 = Absent – MCAN will operate normally.

NOTE: *The following registers may only be accessed when reset request = present: CACC, CACM, CBT0, CBT1, and COCNTRL.*

MCAN command register (CCOM)

This is a write only register; a read of this location will always return the value $FF.

This register may be written only when the RR bit in CCNTRL is clear.

Do not use read-modify-write instructions on this register (e.g. BSET, BCLR).

Address: $0021

	Bit 7	6	5	4	3	2	1	Bit 0
	RX0	RX1	COMPSEL	SLEEP	COS	RRB	AT	TR
External Reset:	0	0	u	0	0	0	0	0
Reset: with RR bit set	0	0	u	0	0	0	0	0

Figure 5. MCAN Command Register (CCOM)

RX0 — Receive pin 0 (passive) (Refer to Figure 21)
   1 = VDD/2 will be connected to the input comparator. The RX0 pin is disconnected.
   0 = The RX0 pin will be connected to the input comparator. VDD/2 is disconnected.

RX1 — Receive pin 1 (passive) (Refer to Figure 21)
   1 = VDD/2 will be connected to the input comparator. The RX1 pin is disconnected.
   0 = The RX1 pin will be connected to the input comparator. VDD/2 is disconnected.

NOTE: *If both RX0 and RX1 are set, or both are clear, then neither of the RX pins will be disconnected.*

COMPSEL — Comparator selector
   1 = RX0 and RX1 will be compared with VDD/2 during sleep mode (see Figure 21).
   0 = RX0 will be compared with RX1 during sleep mode.

SLEEP — Go to sleep

When the SLEEP bit is set during a non synchronous state, an immediate wake-up wil be generated by the MCAN module. (See RR bit description for more details.)

    1 = Sleep – The MCAN will go into sleep mode, as long as there is no activity on the bus. Otherwise the MCAN will issue a wake-up interrupt.

    0 = Wake-up – The MCAN will function normally. If SLEEP is cleared by the CPU then the MCAN will waken up, but will not issue a wake-up interrupt.

NOTE: *If SLEEP is set during the reception or transmission of a message, the MCAN will generate an immediate wake-up interrupt. (This allows for a more orthogonal software implementation on the CPU.) This will have no effect on the transfer layer, i.e. no message will be lost or corrupted.*

The CAF flag in the port configuration register indicates whether or not sleep mode was entered successfully.

A node that was sleeping and has been awakened by bus activity will not be able to receive any messages until its oscillator has started and it has found a valid end of frame sequence (11 recessive bits). The designer must take this into consideration when planning to use the sleep command.

COS — Clear overrun status

    1 = This clears the read-only data overrun status bit in the CSTAT register (see **MCAN status register (CSTAT)**). It may be written at the same time as RRB.

    0 = No action.

RRB — Release receive buffer

When set this releases the receive buffer currently attached to the CPU, allowing the buffer to be reused by the MCAN. This may result in another message being received, which could cause another receive interrupt request (if RIE is set). This bit is cleared automatically when a message is received, i.e. when the RS bit (see **MCAN status register (CSTAT)**) becomes set.

1 = Released – receive buffer is available to the MCAN.

0 = No action.

AT — Abort transmission

When this bit is set a pending transmission will be cancelled if it is not already in progress, allowing the transmit buffer to be loaded with a new (higher priority) message when the buffer is released. If the CPU tries to write to the buffer when it is locked, the information will be lost without being signalled. The status register can be checked to see if transmission was aborted or is still in progress.

1 = Present – Abort transmission of any pending messages.

0 = No action.

TR — Transmission request

1 = Present – Depending on the transmission buffer's content, a data frame or a remote frame will be transmitted.

0 = No action. This will not cancel a previously requested transmission; the abort transmission command must be used to do this.

MCAN status register (CSTAT)

This is a read only register; only the MCAN can change its contents.

Address: $0022

	Bit 7	6	5	4	3	2	1	Bit 0
	BS	ES	TS	RS	TCS	TBA	DO	RBS
External Reset:	0	0	0	0	1	1	0	0
Reset: with RR bit set	u	u	0	0	1	1	0	0

Figure 6. MCAN Status Register (CSTAT)

BS — Bus status

This bit is set (off-bus) by the MCAN when the transmit error counter reaches 256. The MCAN will then set RR and will remain off-bus until the CPU clears RR again. At this point the MCAN will wait for 128 successive occurrences of a sequence of 11 recessive bits before clearing BS and resetting the read and write error counters. While off-bus the MCAN does not take part in bus activities.

1 = Off-bus – The MCAN is not participating in bus activities.

0 = On-bus – The MCAN is operating normally.

ES — Error status

1 = Error – Either the read or the write error counter has reached the CPU warning limit of 96.

0 = Neither of the error counters has reached 96.

TS — Transmit status

1 = Transmit – The MCAN has started to transmit a message.

0 = Idle – If the receive status bit is also clear then the MCAN is idle; otherwise it is in receive mode.

RS — Receive status

1 = Receive – The MCAN entered receive mode from idle, or by losing arbitration during transmission.

0 = Idle – If the transmit status bit is also clear then the MCAN is idle; otherwise it is in transmit mode.

NOTE: *RS will not be set during a bus failure with a permanent dominant bus level.*

MC68HC05X4

14-mcan

TCS — Transmission complete status

This bit is cleared by the MCAN when TR becomes set. When TCS is set it indicates that the last requested transmission was successfully completed. If, after TCS is cleared, but before transmission begins, an abort transmission command is issued then the transmit buffer will be released and TCS will remain clear. TCS will then only be set after a further transmission is both requested and successfully completed.

1 = Complete – Last requested transmission successfully completed.

0 = Incomplete – Last requested transmission not complete.

TBA — Transmit buffer access

When clear, the transmit buffer is locked and cannot be accessed by the CPU. This indicates that either a message is being transmitted, or is awaiting transmission. If the CPU writes to the transmit buffer while it is locked, then the bytes will be lost without this being signalled.

1 = Released – The transmit buffer may be written to by the CPU.

0 = Locked – The CPU cannot access the transmit buffer.

DO — Data overrun

This bit is set when both receive buffers are full and there is a further message to be stored. In this case the new message is dropped, but the internal logic maintains the correct protocol. The MCAN does not receive the message, but no warning is sent to the transmitting node. The MCAN clears DO when the CPU sets the COS bit in the CCOM register.

Note that data overrun can also be caused by a transmission, since the MCAN will temporarily store an outgoing frame in a receive buffer in case arbitration is lost during transmission.

1 = Overrun – Both receive buffers were full and there was another message to be stored.

0 = Normal operation.

RBS — Receive buffer status

This bit is set by the MCAN when a new message is available. When clear this indicates that no message has become available since the last RRB command. The bit is cleared when RRB is set. However, if the second receive buffer already contains a message, then control of that buffer is given to the CPU and RBS is immediately set again. The first receive buffer is then available for the next incoming message from the MCAN.

1 = Full – A new message is available for the CPU to read.

0 = Empty – No new message is available.

MCAN interrupt register (CINT)

All bits of this register are read only; all are cleared by a read of the register.

This register must be read in the interrupt handling routine in order to enable further interrupts.

Address: $0023

	Bit 7	6	5	4	3	2	1	Bit 0
				WIF	OIF	EIF	TIF	RIF
External Reset:	-	-	-	0	0	0	0	0
Reset: with RR bit set	-	-	-	u	0	u	0	0

Figure 7. MCAN Interrupt Register (CINT)

WIF — Wake-up interrupt flag

If the MCAN detects bus activity whilst it is asleep, it clears the SLEEP bit in the CCOM register; the WIF bit will then be set. WIF is cleared by reading the MCAN interrupt register (CINT), or by an external reset.

1 = MCAN has detected activity on the bus and requested wake-up.

0 = No wake-up interrupt has occurred.

MC68HC05X4

OIF — Overrun interrupt flag

When OIE is set then this bit will be set when a data overrun condition is detected. Like all the bits in this register, OIF is cleared by reading the register, or when reset request is set.

1 = A data overrun has been detected.
0 = No data overrun has occurred.

EIF — Error interrupt flag

When EIE is set then this bit will be set by a change in the error or bus status bits in the MCAN status register. Like all the bits in this register, EIF is cleared by reading the register, or by an external reset.

1 = There has been a change in the error or bus status bits in CSTAT.
0 = No error interrupt has occurred.

TIF — Transmit interrupt flag

The TIF bit is set at the end of a transmission whenever both the TBA and TIE bits are set. Like all the bits in this register, TIF is cleared by reading the register, or when reset request is set.

1 = Transmission complete, the transmit buffer is accessible.
0 = No transmit interrupt has occurred.

RIF — Receive interrupt flag

The RIF bit is set by the MCAN when a new message is available in the receive buffer, and the RIE bit in CCNTRL is set. At the same time RBS is set. Like all the bits in this register, RIF is cleared by reading the register, or when reset request is set. After sending a message by the MCAN module, the RIF bit will not be set, even though this message has been written into the receive buffer following successful transmission.

1 = A new message is available in the receive buffer.
0 = No receive interrupt has occurred.

MCAN acceptance code register (CACC)

On reception each message is written into the current receive buffer. The MCU is only signalled to read the message however, if it passes the criteria in the acceptance code and acceptance mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

NOTE: *This register can be accessed only when the RR bit in CCNTRL is set.*

Address: $0024

	Bit 7	6	5	4	3	2	1	Bit 0
	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Reset: Undefined

Figure 8. MCAN Acceptance Code (CACC)

AC7 – AC0 — Acceptance code bits

AC7 – AC0 comprise a user defined sequence of bits with which the 8 most significant bits of the data identifier (ID10 – ID3) are compared. The result of this comparison is then masked with the acceptance mask register. Once a message has passed the acceptance criterion the respective identifier, data length code and data are sequentially stored in a receive buffer, providing there is one free. If there is no free buffer, the data overrun condition will be signalled.

On acceptance the receive buffer status bit is set to full and the receive interrupt bit is set (provided RIE = enabled).

**MCAN
acceptance mask
register (CACM)**

The acceptance mask register specifies which of the corresponding bits in the acceptance code register are relevant for acceptance filtering.

NOTE: *This register can be accessed only when the RR bit in CCNTRL is set.*

Address: $0025

	Bit 7	6	5	4	3	2	1	Bit 0
	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0

Reset: Undefined

Figure 9. MCAN Acceptance Mask (CACM)

AM0 – AM7 — Acceptance mask bits

When a particular bit in this register is clear this indicates that the corresponding bit in the acceptance code 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. When a bit is set, it indicates that the state of the corresponding bit in the acceptance code register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.
0 = Match corresponding acceptance code register and identifier bits.

MCAN bus timing register 0 (CBT0)

Address: $0026

Bit 7	6	5	4	3	2	1	Bit 0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0

Reset: Undefined
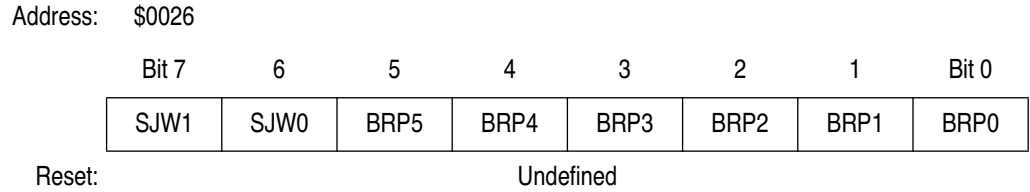
Figure 10. MCAN Bus Timing 0 (CBT0)

NOTE: *This register can be accessed only when the RR bit in CCNTRL is set.*

SJW1, SJW0 — Synchronization jump width bits

The synchronization jump width defines the maximum number of system clock (t_{SCL}) cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 1).

Table 1. Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 t_{SCL} cycle
0	1	2 t_{SCL} cycles
1	0	3 t_{SCL} cycles
1	1	4 t_{SCL} cycles

BRP5 – BRP0 — Baud rate prescaler bits

These bits determine the MCAN system clock cycle time (t_{SCL}), which is used to build up the individual bit timing, according to Table 2 and the formula in Figure 11.

Table 2. 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
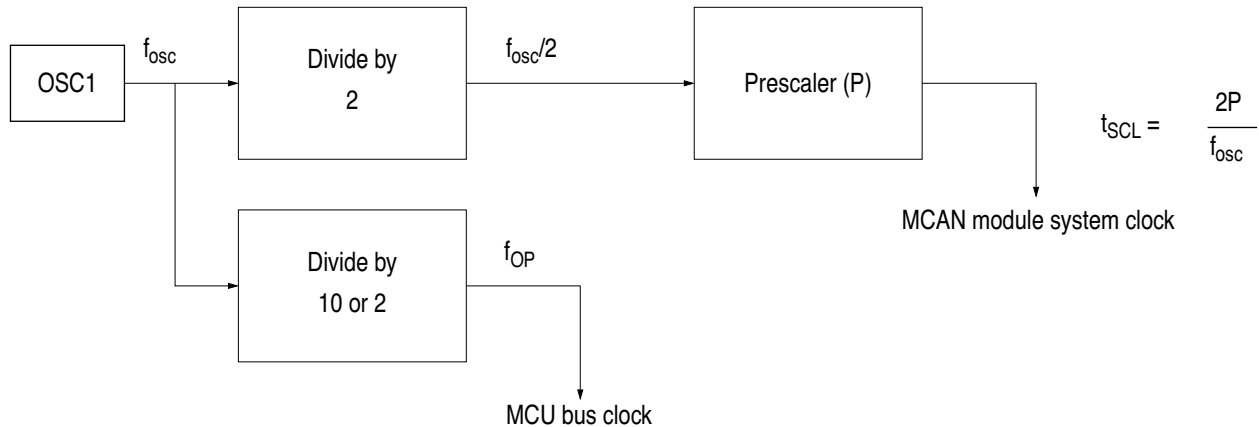
Figure 11. Oscillator block diagram

MCAN bus timing register 1 (CBT1)

This register can only be accessed only when the RR bit in CCNTRL is set.

Address: $0027

Bit 7	6	5	4	3	2	1	Bit 0
SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10

Reset: Undefined

Figure 12. MCAN Bus Timing 1 (CBT1)

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. When set three samples per bit are taken. This sample rate gives better rejection of noise on the bus, but introduces a one bit delay to the bus sampling. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit.
0 = One sample per bit.

TSEG22 – TSEG10 — Time segment bits

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

Freescale Semiconductor, Inc.
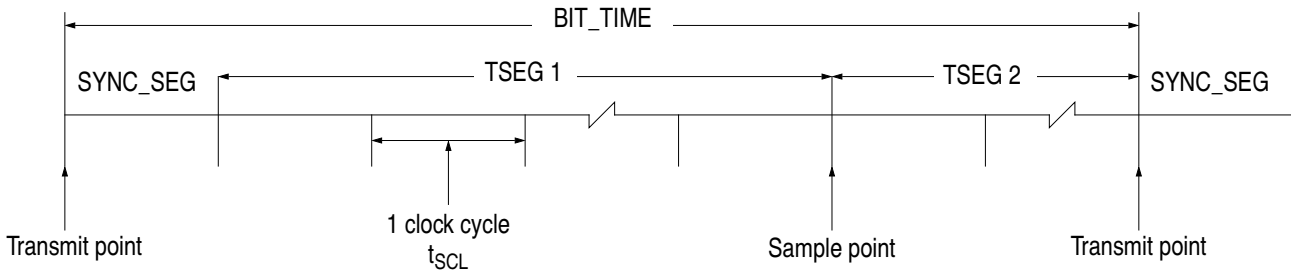
Figure 13. Segments within the bit time

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 MCAN 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 3.

Table 3. Time segment values

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	1	2 t_{SCL} cycles
0	0	1	0	3 t_{SCL} cycles
0	0	1	1	4 t_{SCL} cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 t_{SCL} cycles

TSEG22	TSEG21	TSEG20	Time segment 2
0	0	1	2 t_{SCL} cycles
.	.	.	.
.	.	.	.
1	1	1	8 t_{SCL} cycles

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of bus clock cycles (t_{SCL}) per bit (as shown above).

Calculation of the bit time

$$\text{BIT\_TIME} = \text{SYNC\_SEG} + \text{TSEG1} + \text{TSEG2}$$

NOTE: *TSEG2 must be at least 2 t_{SCL}, i.e. the configuration bits must not be 000. (If three samples per bit mode is selected then TSEG2 must be at least 3 t_{SCL}.)*

TSEG1 must be at least as long as TSEG2.

The synchronization jump width (SJW) may not exceed TSEG2, and must be at least t_{SCL} shorter than TSEG1 to allow for physical propagation delays.

i.e. in terms of t_{SCL}:

$$\text{SYNC\_SEG} = 1$$

$$\text{TSEG1} \geq \text{SJW} + 1$$

$$\text{TSEG1} \geq \text{TSEG2}$$

$$\text{TSEG2} \geq \text{SJW}$$

and $\text{TSEG2} \geq 2$ (SAMP = 0)

or $\text{TSEG2} \geq 3$ (SAMP = 1)

These boundary conditions result in minimum bit times of 5 t_{SCL}, for one sample, and 7 t_{SCL}, for three samples per bit.

MCAN output control register (COCNTRL)

This register allows the setup of different output driver configurations under software control. The user may select active pull-up, pull-down, float or push-pull output.

NOTE: *This register can be accessed only when the RR bit in CCNTRL is set.*

Address: $0028

Bit 7	6	5	4	3	2	1	Bit 0
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCM1	OCM0

Reset: Undefined

Figure 14. MCAN Output Control (COCNTRL)

OCM1 and OCM0 — Output control mode bits

The values of these two bits determine the output mode, as shown in Table 4.

Table 4. Output control modes

OCM1	OCM0	Function
0	0	Biphase mode
0	1	Not used
1	0	Normal mode 1 Bit stream transmitted on both TX0 and TX1
1	1	Normal mode 2 TX0 – bit sequence TX1 – bus clock (t_{xclk})

NOTE: *The transmit clock (t_{xclk}) is used to indicate the end of the bit time and will be high during the SYNC_SEG.*

For all the following modes of operation, a dominant bit is internally coded as a zero, a recessive as a one. The other output control bits are used to determine the actual voltage levels transmitted to the MCAN bus for dominant and recessive bits.

MC68HC05X4

24-mcan

Biphase mode

If the CAN modules are isolated from the bus lines by a transformer then the bit stream has to be coded so that there is no resulting dc component. There is a flip-flop within the MCAN that keeps the last dominant configuration; its direct output goes to TX0 and its complement to TX1. The flip-flop is toggled for each dominant bit; dominant bits are thus sent alternately on TX0 and TX1; i.e. the first dominant bit is sent on TX0, the second on TX1, the third on TX0 and so on. During recessive bits, all output drivers are deactivated (i.e. high impedance).

Normal mode 1

In contrast to biphase mode the bit representation is time invariant and not toggled.

Normal mode 2

For the TX0 pin this is the same as normal mode 1, however the data stream to TX1 is replaced by the transmit clock. The rising edge of the transmit clock marks the beginning of a bit time. The clock pulse will be t_{SCL} long.

Other output control bits

The other six bits in this register control the output driver configurations, to determine the format of the output signal for a given data value (see Figure 21).

OCTP0/1 – These two bits control whether the P-type output control transistors are enabled.

OCTN0/1 – These two bits control whether the N-type output control transistors are enabled.

OCPOL0/1 – These two bits determine the driver output polarity for each of the MCAN bus lines (TX0, TX1).

TP0/1 and TN0/1 – These are the resulting states of the output transistors.

TD – This is the internal value of the data bit to be transferred across the MCAN bus. (A zero corresponds to a dominant bit, a one to a recessive.)

The actions of these bits in the output control register are as shown in Table 5.

Table 5. MCAN driver output levels

Mode	TD	OCPOLi	OCTPi	OCTNi	TPi	TNi	TXi output level
Float	0	0	0	0	Off	Off	Float
	1	0	0	0	Off	Off	Float
	0	1	0	0	Off	Off	Float
	1	1	0	0	Off	Off	Float
Pull-down	0	0	0	1	Off	On	Low
	1	0	0	1	Off	Off	Float
	0	1	0	1	Off	Off	Float
	1	1	0	1	Off	On	Low
Pull-up	0	0	1	0	Off	Off	Float
	1	0	1	0	On	Off	High
	0	1	1	0	On	Off	High
	1	1	1	0	Off	Off	Float
Push-pull	0	0	1	1	Off	On	Low
	1	0	1	1	On	Off	High
	0	1	1	1	On	Off	High
	1	1	1	1	Off	On	Low

Transmit buffer identifier register (TBI)

Address: $002A

Bit 7	6	5	4	3	2	1	Bit 0
ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
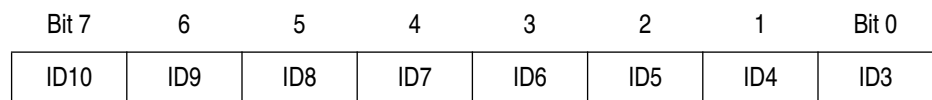
Reset: Undefined

Figure 15. Transmit Buffer Identifier Register (TBI)

ID10 – ID3 — Identifier bits

The identifier consists of 11 bits (ID10 – ID0). ID10 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. The three least significant bits are contained in the TRTDL register. The seven most significant bits must not all be recessive.

Remote transmission request and data length code register (TRTDL)

Address: $002B

Bit 7	6	5	4	3	2	1	Bit 0
ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
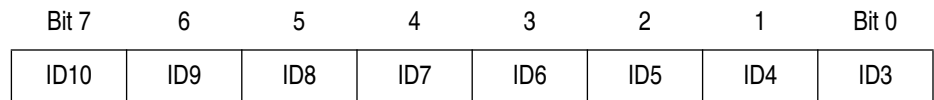
Reset: Undefined

Figure 16. RTR and Data Length Code Register (TRTDL)

ID2 – ID0 — Identifier bits

These bits contain the least significant bits of the transmit buffer identifier.

RTR — Remote transmission request
1 = A remote frame will be transmitted.
0 = A data frame will be transmitted.

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 ignored, forcing the number of bytes to be 0. The data byte count ranges from 0 to 8 for a data frame. Table 6 shows the effect of setting the DLC bits.

Table 6. 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

**Transmit data
segment registers
(TDS) 1 – 8**

Address: $002C–$0033

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Reset: Undefined

Figure 17. Transmit Data Segment Registers (TDS)

DB7 – DB0 — data bits

These data bits in the eight data segment registers make up the bytes
of data to be transmitted. The number of bytes to be transmitted is
determined by the data length code.

**Receive buffer
identifier register
(RBI)**

The layout of this register is identical to the TBI register (see **Transmit
buffer identifier register (TBI)**).

Address: $0034

Bit 7	6	5	4	3	2	1	Bit 0
ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3

Reset: Undefined

Figure 18. Receive Buffer Identifier Register (RBI)

(Note that there are actually two receive buffer register sets, but
switching between them is handled internally by the MCAN.)

**For More Information On This Product,
Go to: www.freescale.com**

Remote transmission request and data length code register (RRTDL)

The layout of this register is identical to the TRTDL register (see **Remote transmission request and data length code register (TRTDL)**).

Address: $0035

Bit 7	6	5	4	3	2	1	Bit 0
ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0

Reset: Undefined

Figure 19. RTR and Data Length Code Register (RRTDL)

Receive data segment registers (RDS) 1 – 8

The layout of these registers is identical to the TDSx registers (see **Transmit data segment registers (TDS) 1 – 8**).

Address: $0036–$003D

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Reset: Undefined

Figure 20. Receive Data Segment Registers (RDS)

(Note that there are actually two receive buffer register sets, but switching between them is handled internally by the MCAN.)

Interface to the MCAN bus

Physically, the MCAN bus may be composed of two wires. The bus can take on one of two values: dominant or recessive. During simultaneous transmission of dominant and recessive bits by two or more CAN modules the resulting bus value will be dominant. (For example, with a wired-AND implementation of the bus, the dominant level would correspond to a logic 0, and the recessive level to a logic 1.)

The two wires of the MCAN bus are designated CANH and CANL. The voltage levels appearing on these lines are designated V_{CANH} and V_{CANL}. A simple termination network is required for each wire. Figure 21 shows the physical interface circuitry within the MCAN module, and its connection to the MCAN bus with a typical low speed (<125 kbaud) hardware interface. (Note that the suggested values shown in the diagram are subject to change in the future.)

For the voltage and resistor values shown in Figure 21 the voltages on the MCAN bus are:

- Recessive level: $V_{CANH} = 3.25$ V $\quad V_{CANL} = 1.75$ V
- Dominant level: $V_{CANH} = 1.00$ V $\quad V_{CANL} = 4.00$ V

If several CAN modules are driving a dominant level on the bus at the same time then the values for V_{CANH} and V_{CANL} can go to 0.3 and 4.7 volts respectively. The residual 0.3 V is due to the voltage drop across the diodes and driver transistors in the transmission circuit.

The receiver part of the network uses two identical voltage divider networks, with a divide ratio of 6:1 (resistor values of 150kΩ and 30kΩ) referenced to $V_{DD}/2$. This increases the common mode range of the input comparator on the physical bus lines. If the common mode range of the comparator at its inputs is 1.5 to 3.5 volts then, for $V_{DD} = 5.0$ V, the common mode range will be increased to −3.5 to +8.5 volts on the bus lines.
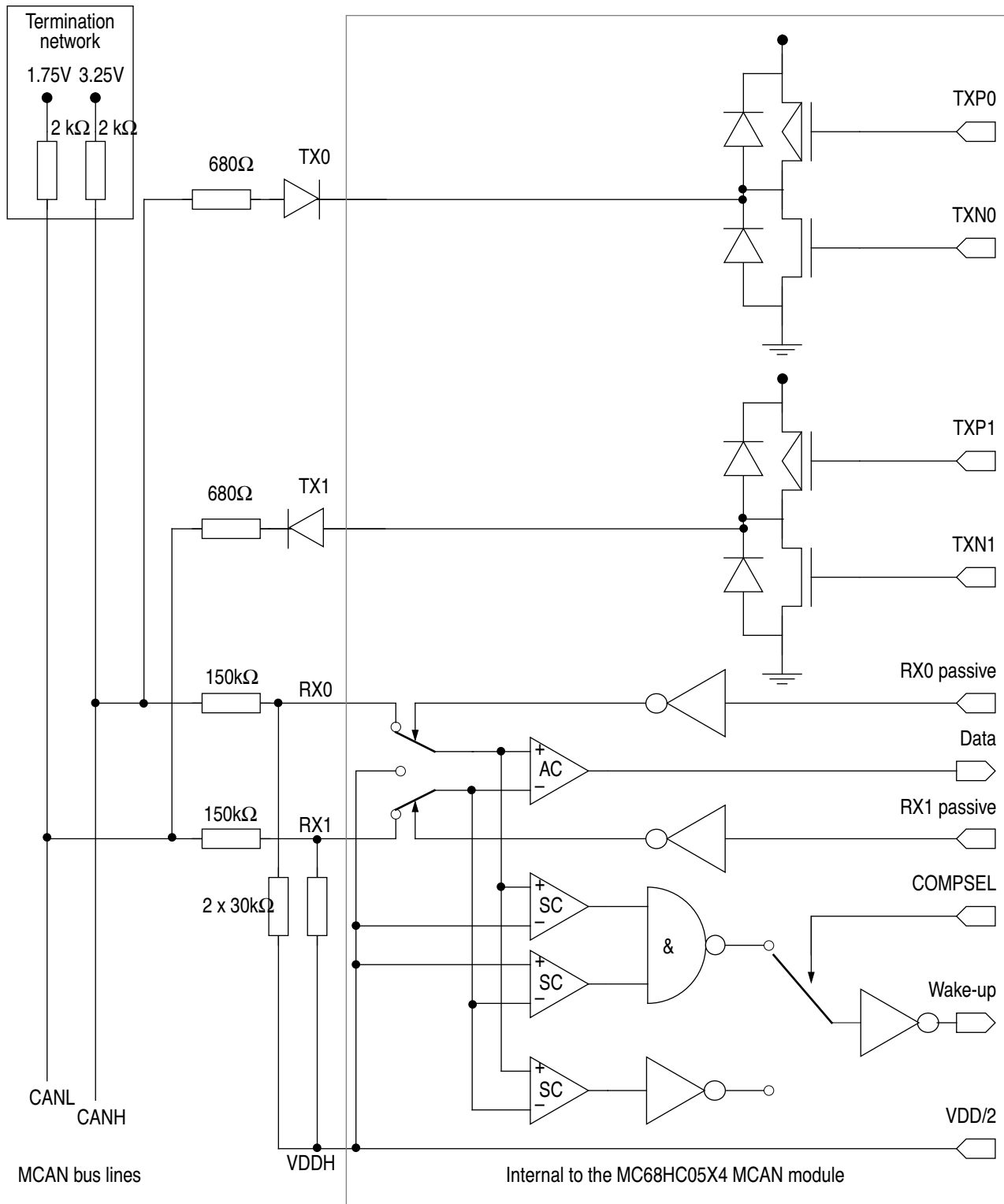
Figure 21. A typical physical interface between the MCAN and the MCAN bus lines

Single wire operation

In the event of a bus fault occurring, limited operation of the MCAN bus may still be possible, depending on the nature of the fault. If the fault is due to a short circuit between the two bus lines or between one of the lines and ground, battery voltage or some other potential, it is possible to identify (using a special software procedure) the line on which the fault exists and to switch the corresponding comparator input from the faulty line to the $V_{DD}/2$ reference supply. At the same time the driver transistors to the faulty line should also be switched off. This will allow communication to continue on the bus. One result of this mode of one wire transmission is a significant reduction in the common mode range of the input comparator.

Switching to one wire operation is achieved using the control bits RX0-passive and RX1-passive in the MCAN command register, located at address $21. Setting either of these bits will result in the corresponding input being disconnected from the bus and connected to $V_{DD}/2$.

Sleep mode

If the SLEEP bit in the MCAN command register is set by the processor the MCAN will go to sleep, unless it is active. If there is activity on the MCAN bus lines, or there is an interrupt pending, the MCAN is deemed to be active and will not go to sleep; a wake-up interrupt will be generated by the MCAN in these circumstances. The SLEEP bit may also be cleared by the processor, in which case no wake-up interrupt will be generated. Note that this bit is write-only by the CPU, and it is not possible therefore to check whether sleep mode has been entered by reading it. However, the CAF bit in the port configuration register is set when the MCAN is asleep, and cleared when it is woken up (see **Port configuration register (PCR)**).

Port configuration register (PCR)

Address: $0003

Bit 7	6	5	4	3	2	1	Bit 0
		WOIF	TIMEN	CAF	BPDE	BWE	AWPS

Reset: -   -   0   0   0   0   0   0
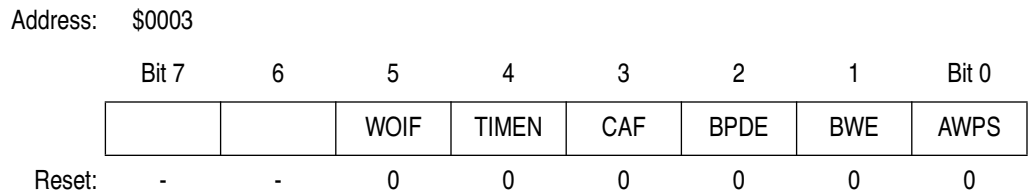
Figure 22. Port Configuration Register (PCR)

CAF — MCAN asleep flag
   1 = The MCAN is in sleep mode
   0 = The MCAN is awake

In order to minimize power consumption, the active comparator is switched off and the sleep comparator circuitry is used to detect activity on the bus. When in sleep mode the MCAN stops its own clocks, leaving the MCU in normal run mode. (Similarly a STOP instruction will stop the processor clocks, leaving the MCAN in run mode.) The on-chip oscillator will stop only if the MCAN is in sleep mode and the MCU executes a STOP instruction. There is a time delay between the STOP instruction being executed and the oscillator stopping. During this time it is possible that the MCAN will come out of sleep mode, and hence prevent the oscillator from stopping.

When a dominant level is detected on the MCAN bus, the MCAN is woken up and a wake-up interrupt is generated.

Sleep comparator reference

When the COMPSEL bit in the MCAN command register ($21) is cleared the sleep comparator inputs are the same as for the active comparator. However, when the COMPSEL bit is set each input is compared with $V_{DD}/2$ (VDDH – see Figure 21) to detect a dominant level. For further details of the active comparator, the sleep comparator and VDDH, refer to **ELECTRICAL SPECIFICATIONS**.

Core Timer

Contents

Introduction

The MC68HC05X4 has a 15-stage ripple counter called the core timer (CTIMER). Features of this timer are: timer overflow, power-on reset (POR), real time interrupt (RTI) with four selectable interrupt rates, and a computer operating properly (COP) watchdog timer.

As shown in Figure 1, the timer is driven by the internal bus clock divided by four with a fixed prescaler. This signal drives an 8-bit ripple counter. The value of this 8-bit ripple counter can be read by the CPU at any time, by accessing the CTIMER counter register (CTCR) at address $09. A timer overflow function is implemented on the last stage of this counter, giving a possible interrupt at the rate of $f_{OP}/1024$. The POR signal (t_{PORL}) is also derived from this register, at $f_{OP}/32$. The counter register circuit is followed by four more stages, with the resulting clock ($f_{OP}/16384$) driving the real time interrupt circuit. The RTI circuit consists of three divider stages with a 1-of-4 selector. The output of the RTI circuit is further divided by eight to drive the COP watchdog timer circuit. The RTI rate selector bits, and the RTI and CTIMER overflow enable bits and

Internal bus
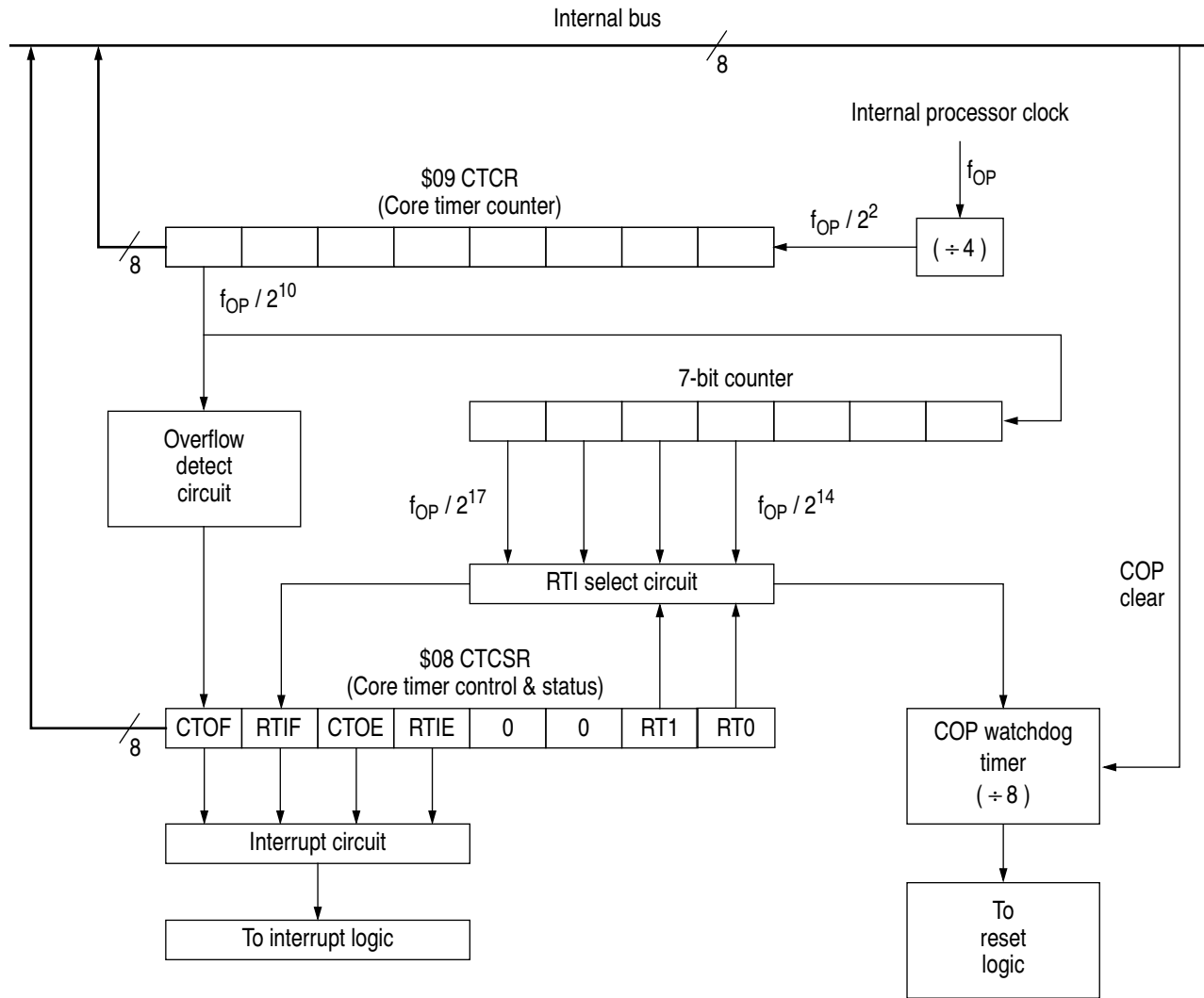
Figure 1. Core timer block diagram

flags, are located in the CTIMER control and status register (CTCSR) at location $08.

CTOF (core timer overflow flag) is a read-only status bit which is set when the 8-bit ripple counter rolls over from $FF to $00. A CPU interrupt request will be generated if CTOE is set. Clearing CTOF is done by writing a '0' to it . Writing a '1' to CTOF has no effect on the bit's value. Reset clears CTOF.

MC68HC05X4 Rev 1.0

2-ctimer

When CTOE (core timer overflow enable) is set, a CPU interrupt request is generated when the CTOF bit is set. Reset clears CTOE.

The core timer counter register (CTCR) is a read-only register that contains the current value of the 8-bit ripple counter at the beginning of the timer chain. This counter is clocked at $f_{OP}/4$ and can be used for various functions including a software input capture. Extended time periods can be attained using the CTIMER overflow function to increment a temporary RAM storage location thereby simulating a 16-bit (or more) counter.

The power-on cycle clears the entire counter chain and begins clocking the counter. After t_{PORL} cycles, the power-on reset circuit is released, which again clears the counter chain and allows the device to come out of reset. At this point, if RESET is not asserted, the timer will start counting up from zero and normal device operation will begin. When RESET is asserted at any time during operation (other than POR), the counter chain is cleared.

Real time interrupts (RTI)

The real time interrupt circuit consists of a three stage divider and a 1-of-4 selector. The clock frequency that drives the RTI circuit is $f_{OP}/2^{14}$ (or $f_{OP}/16384$), with three additional divider stages, giving a maximum interrupt period of 4 seconds at a bus frequency (f_{OP}) of 32.768 kHz. Register details are given in **Core timer registers**.

Computer operating properly (COP) watchdog timer

The COP watchdog timer function is implemented by taking the output of the RTI circuit and further dividing it by eight, as shown in Figure 1. Note that the minimum COP timeout period is seven times the RTI period. This is because the COP will be cleared asynchronously with respect to the value in the core timer counter register/RTI divider, hence the actual COP timeout period will vary between 7x and 8x the RTI period.

If the COP circuit times out, an internal reset is generated and the normal reset vector is fetched. COP timeout is prevented by writing a '0' to bit 0 of address $1FF0. When the COP is cleared, only the final divide-by-eight stage is cleared (see Figure 1).

The COP function is a mask option, enabled or disabled during device manufacture.

Core timer registers

Core timer control and status register (CTCSR)

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
	CTOF	RTIF	CTOE	RTIE	0	0	RT1	RT0
Reset:	u	u	0	0	0	0	1	1

Figure 2. Core Timer Control/Status Register (CTCSR)

CTOF — Core timer overflow
> 1 = This read-only flag is set whenever a core timer overflow occurs.
> 0 = No core timer overflow has occurred.

This bit is set when the core timer counter register rolls over from $FF to $00; an interrupt request will be generated if CTOE is set. When set, the bit may be cleared by writing a '0' to it.

RTIF — Real time interrupt flag
> 1 = This read-only flag is set when the pre-selected RTI period has elapsed. The RTI period is selected using the RT0 and RT1 bits as shown in Table 1.
> 0 = The pre-selected RTI period has not elapsed.

Freescale Semiconductor, Inc.

This bit is set when the output of the chosen stage becomes active; an interrupt request will be generated if RTIE is set. When set, the bit may be cleared by writing a '0' to it.

CTOE — Core timer overflow interrupt enable

1 = A core timer overflow interrupt will be generated if CTOF is set and the I-bit in the CCR is clear.

0 = No core timer overflow interrupt will be generated regardless of the state of the CTOF flag and the I-bit.

RTIE — Real time interrupt enable

0 = A real time interrupt will be generated if RTIF is set and the I-bit in the CCR is clear.

0 = No real time interrupt will be generated regardless of the state of the RTIF flag and the I-bit.

RT1, RT0 — Real time interrupt rate select

These two bits select one of four taps from the real time interrupt circuitry. Reset sets both RT0 and RT1 to one, selecting the lowest periodic rate and therefore the maximum time in which to alter them if necessary. The COP reset times are also determined by these two bits. Care should be taken when altering RT0 and RT1 if a timeout is imminent, or if the timeout period is uncertain. If the selected tap is modified during a cycle in which the counter is switching, an RTIF could be missed or an additional one could be generated. To avoid problems, the COP should be cleared before changing the RTI taps. See Table 1 for some example RTI periods.

Table 1. Example RTI periods

RT1	RT0	Division ratio	Bus frequency f_{OP} = 500 kHz		Bus frequency f_{OP} = 1 MHz		Bus frequency f_{OP} = 2 MHz	
			RTI period	Minimum COP period	RTI period	Minimum COP period	RTI period	Minimum COP period
0	0	2^{14}	31.3ms	218.8ms	15.6ms	109.4ms	7.8ms	54.7ms
0	1	2^{15}	62.5ms	437.5ms	31.3ms	218.8ms	15.6ms	109.4ms
1	0	2^{16}	125ms	875.0ms	62.5ms	437.5ms	31.3ms	218.8ms
1	1	2^{17}	250ms	1.75s	125.1ms	875.0ms	62.5ms	437.5ms

Core timer counter register (CTCR)

Address: $0009

Bit 7	6	5	4	3	2	1	Bit 0

Reset: 0   0   0   0   0   0   0   0
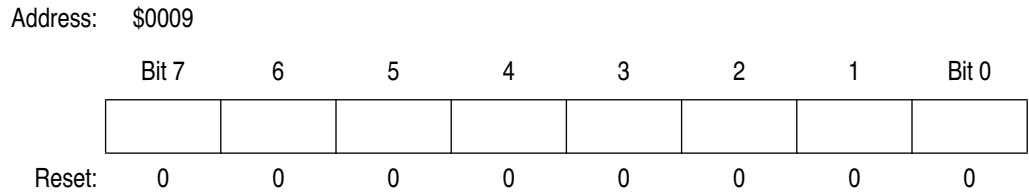
Figure 3. Core Timer Counter Register (CTCR)

The core timer counter register is a read-only register, which contains the current value of the 8-bit ripple counter at the beginning of the timer chain. Reset clears this register.

Core timer during WAIT

The CPU clock halts during the WAIT mode, but the core timer remains active. If the CTIMER interrupts are enabled, then a CTIMER interrupt will cause the processor to exit the WAIT mode.

Core timer during STOP

The timer is cleared when going into STOP mode. When STOP is exited by an external interrupt or an external reset, the internal oscillator will restart, followed by an internal processor stabilization delay (t_{PORL}). The timer is then cleared and operation resumes.

16-Bit Programmable Timer

Contents

Introduction

Besides the core timer the MC68HC05X4 has a 16-bit programmable timer. This timer consists of a 16-bit read-only free-running counter, with a fixed divide-by-four prescaler, plus the input capture/output compare circuitry. Selected input edges cause the current counter value to be latched into a 16-bit input capture register so that software can later read this value to determine when the edge occurred. When the free running counter value matches the value in the output compare registers, the programmed pin action takes place. Refer to Figure 1 for a block diagram of the timer. The timer must be enabled by setting the TIMEN bit in the port configuration register (PCR).

The timer has a 16-bit architecture, hence each specific functional segment is represented by two 8-bit registers. These registers contain the high and low byte of that functional segment. Accessing the low byte of a specific timer function allows full control of that function; however, an access of the high byte inhibits that specific timer function until the low byte is also accessed.

NOTE: *The I-bit in the CCR should be set while manipulating both the high and low byte register of a specific timer function to ensure that an interrupt does not occur.*

The timer must be enabled by setting the TIMEN bit in the port configuration register (PCR)

Port configuration register (PCR)

Address: $0003

		Bit 7	6	5	4	3	2	1	Bit 0
				WOIF	TIMEN	CAF	BPDE0	BWE	AWPS
Reset:		-	-	0	0	0	0	0	0

Figure 1. Port Configuration Register (PCR)

TIMEN — Timer enable

When set, this bit enables the 16-bit programmable timer and selects the TCMP function on the shared PB7/TCMP pin. When this bit is cleared the TCMP function is disabled, but the timer is still enabled and may produce interrupts from an output compare event.

> 1 = 16-bit timer enabled
> 0 = 16-bit timer disabled

MC68HC05X4 Rev 1.0

2-ptimer

Freescale Semiconductor, Inc.

Internal bus

8

Internal processor clock

8-bit buffer

High byte | Low byte

f_{OP}

High byte | Low byte

High byte | Low byte

Output compare register $16 $17

(÷ 4)

16-bit free-running counter $18 $19

Alternate counter register $1A $1B

Input capture register $14 $15

Output compare circuit

Overflow detect circuit

Edge detect circuit

Output level register

CLK

D C Q

ICF | OCF | TOF $13 TSR (Timer status register)

RESET

ICIE | OCIE | TOIE | 0 | 0 | 0 | IEDG | OLVL

$12 TCR (Timer control register)

TCMP (Output level)

Interrupt circuit

TCAP (Input edge)

Table 1. 16-bit programmable timer block diagram

Counter

The key element in the programmable timer is a 16-bit, free-running counter, or counter register, preceded by a prescaler that divides the internal processor clock by four. The prescaler gives the timer a resolution of $2\mu s$ if the internal bus clock is 2MHz. The counter is incremented during the low portion of the internal bus clock. Software can read the counter at any time without affecting its value.

Counter registers

Address: $0018

Bit 15	14	13	12	11	10	9	Bit 8

Reset: 1 1 1 1 1 1 1 1
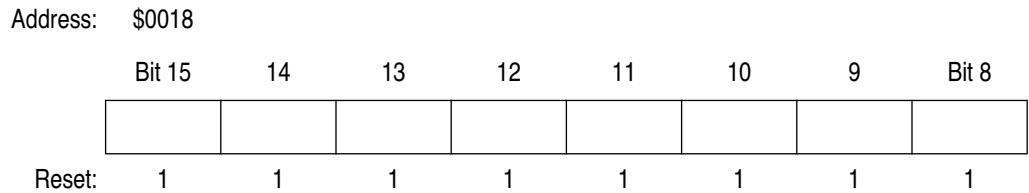
Figure 2. Timer Counter High Register (TCH)

Address: $0019

Bit 7	6	5	4	3	2	1	Bit 0

Reset: 1 1 1 1 1 1 0 0
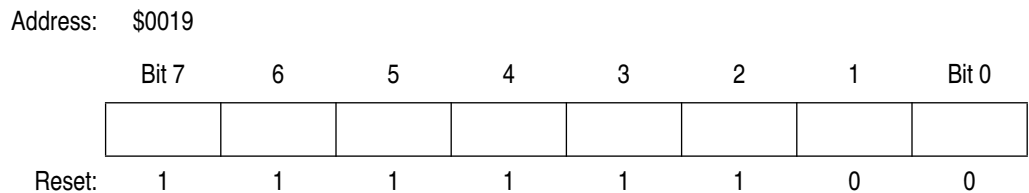
Figure 3. Timer Counter Low Register (TCL)

Address: $001A

Bit 15	14	13	12	11	10	9	Bit 8

Reset: 1   1   1   1   1   1   1   1
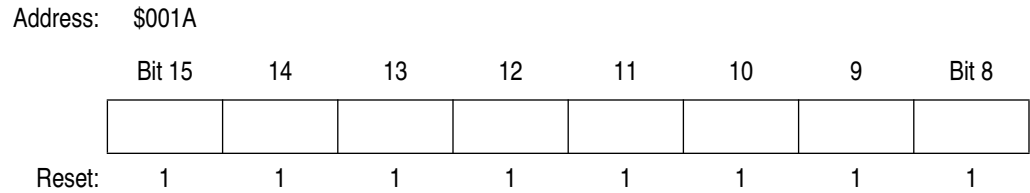
Figure 4. Alternate Counter High Register (ACH)

Address: $001B

Bit 7	6	5	4	3	2	1	Bit 0

Reset: 1   1   1   1   1   1   0   0
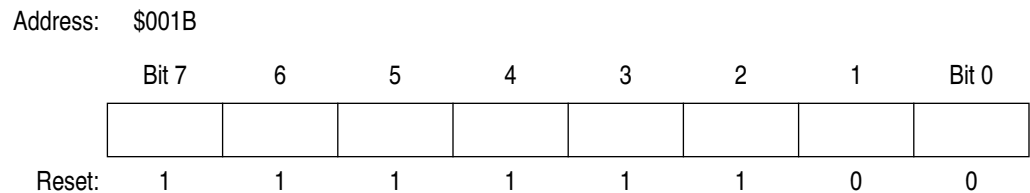
Figure 5. Alternate Counter High Register (ACL)

The double-byte, free-running counter can be read from either of two locations, $18 – $19 (counter register) or $1A – $1B (alternate counter register). A read from only the less significant byte (LSB) of the free-running counter ($19 or $1B) receives the count value at the time of the read. If a read of the free-running counter or alternate counter register first addresses the more significant byte (MSB) ($18 or $1A), the LSB is transferred to a buffer. This buffer value remains fixed after the first MSB read, even if the user reads the MSB several times. This buffer is accessed when reading the free-running counter or alternate counter register LSB and thus completes a read sequence of the total counter value. In reading either the free-running counter or alternate counter register, if the MSB is read, the LSB must also be read to complete the sequence. If the timer overflow flag (TOF) is set when the counter register LSB is read, then a read of the TSR will clear the flag.

The alternate counter register differs from the counter register only in that a read of the LSB does not clear TOF. Therefore, to avoid the possibility of missing timer overflow interrupts due to clearing of TOF, the alternate counter register should be used where this is a critical issue.

The free-running counter is set to $FFFC during reset and is always a read-only register. During a power-on reset, the counter is also preset to $FFFC and begins running after the oscillator start-up delay. Because the free-running counter is 16 bits preceded by a fixed divide-by-four

prescaler, the value in the free-running counter repeats every 262144 internal bus clock cycles. TOF is set when the counter overflows (from $FFFF to $0000); this will cause an interrupt if TOIE is set.

Bits 8 – 15 — MSB of counter/alternate counter register

A read of only the more significant byte (MSB) transfers the LSB to a buffer, which remains fixed after the first MSB read, until the LSB is also read.

Bits 0 – 7 — LSB of counter/alternate counter register

A read of only the less significant byte (LSB) receives the count value at the time of reading.

Timer functions

The 16-bit programmable timer is monitored and controlled by a group of ten registers, full details of which are contained in the following paragraphs. An explanation of the timer functions is also given.

Timer control register (TCR)

The timer control register ($12) is used to enable the input capture (ICIE), output compare (OCIE), and timer overflow (TOIE) interrupt enable functions as well as selecting input edge sensitivity (IEDG) and output level polarity (OLVL).

Address: $0012

	Bit 7	6	5	4	3	2	1	Bit 0
	ICIE	OCIE	TOIE	0	0	0	IEDG	OLVL
Reset:	0	0	0	0	0	0	u	0

Figure 6. Timer Control Register (TCR)

ICIE — Input capture interrupt enable

If this bit is set, a timer interrupt is enabled whenever the ICF1 or ICF2 status flag in the timer status register (TSR) is set.

1 = Interrupt enabled.
0 = Interrupt disabled.

OCIE — Output compare interrupt enable

If this bit is set, a timer interrupt is enabled whenever the OCF1 or OCF2 status flag in TSR is set.

1 = Interrupt enabled.
0 = Interrupt disabled.

TOIE — Timer overflow interrupt enable

If this bit is set, a timer interrupt is enabled whenever the TOF status flag in TSR is set.

1 = Interrupt enabled.
0 = Interrupt disabled.

IEDG — Input edge

When IEDG is set, a positive-going edge on the TCAP pin will trigger a transfer of the free-running counter value to the input capture register. When clear, a negative-going edge triggers the transfer.

1 = TCAP is positive-going edge sensitive.
0 = TCAP is negative-going edge sensitive.

OLVL — Output level

When OLVL is set, a high output level is clocked into the output level register by the next successful output compare, and appears on the TCMP pin. When clear, a low level appears on the TCMP pin.

1 = A high output level will appear on the TCMP pin.
0 = A low output level will appear on the TCMP pin.

Timer status register (TSR)

The timer status register ($13) contains the status bits for the interrupt conditions ICF, OCF, and TOF.

Accessing the timer status register satisfies the first condition required to clear the status bits. The remaining step is to access the register corresponding to the status bit.

Address: $0013

	Bit 7	6	5	4	3	2	1	Bit 0
	ICF	OCF	TOF	0	0	0	0	0
Reset:	u	u	u	0	0	0	0	0

Figure 7. Timer Status Register (TSR)

ICF — Input capture flag
        1 = A valid input capture has occurred.
        0 = No input capture has occurred.

This bit is set when the selected polarity of edge is detected by the input capture edge detector; an input capture interrupt will be generated, if ICIE is set. ICF is cleared by reading the TSR and then the input capture low register ($15).

OCF — Output compare flag
        1 = A valid output compare has occurred.
        0 = No output compare has occurred.

This bit is set when the output compare register contents match those of the free-running counter; an output compare interrupt will be generated, if OCIE is set. OCF is cleared by reading the TSR and then the output compare low register ($17).

TOF — Timer overflow flag
        1 = Timer overflow has occurred.
        0 = No timer overflow has occurred.

This bit is set when the free-running counter overflows from $FFFF to $0000; a timer overflow interrupt will occur, if TOIE is set. TOF is cleared by reading the TSR and the counter low register ($19).

When using the timer overflow function and reading the free-running counter at random times to measure an elapsed time, a problem may occur whereby the timer overflow flag is unintentionally cleared if:

1. the timer status register is read or written when TOF is set, **and**

2. the LSB of the free-running counter is read, but not for the purpose of servicing the flag.

Reading the alternate counter register instead of the counter register will avoid this potential problem.

Input capture function

'Input capture' is a technique whereby an external signal (connected to the TCAP pin) is used to trigger a read of the free-running counter. In this way it is possible to relate the timing of an external signal to the internal counter value, and hence to elapsed time.

Input capture registers

Address: $0014

Bit 15	14	13	12	11	10	9	Bit 8

Reset: Undefined
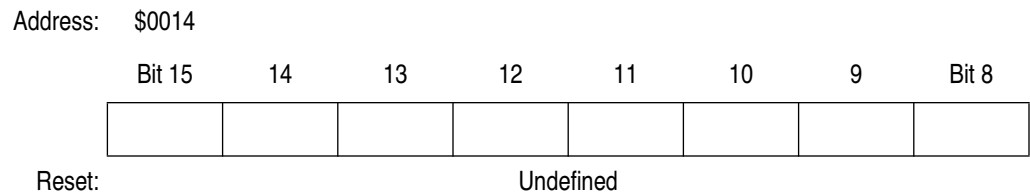
Figure 8. Input Capture High Register (ICH)

Address: $0015

Bit 7	6	5	4	3	2	1	Bit 0

Reset: Undefined
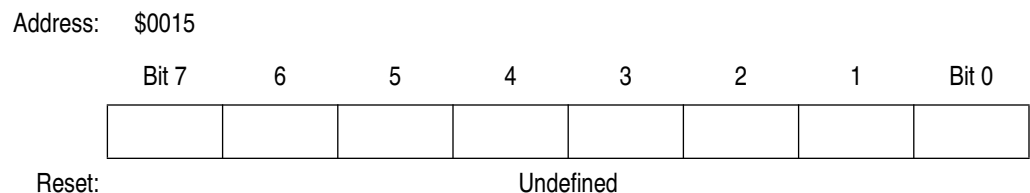
Figure 9. Input Capture Low Register (ICL)

The two 8-bit registers that make up the 16-bit input capture register are read-only, and are used to latch the value of the free-running counter after the input capture edge detector senses a valid transition. The level transition that triggers the counter transfer is defined by the input edge

bit (IEDG). The most significant 8 bits are stored in the input capture high register at $14, the least significant in the input capture low register at $15.

The result obtained from an input capture will be one greater than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization. Resolution is one count of the free-running counter, which is four internal bus clock cycles. The free-running counter contents are transferred to the input capture register on each valid signal transition whether the input capture flag (ICF) is set or clear. The input capture register always contains the free-running counter value that corresponds to the most recent input capture. After a read of the input capture register MSB at location $14, the counter transfer is inhibited until the LSB at location $15 is also read. This characteristic causes the time used in the input capture software routine and its interaction with the main program to determine the minimum pulse period. A read of the input capture register LSB at location $15 does not inhibit the free-running counter transfer since the two actions occur on opposite edges of the internal bus clock.

Reset does not affect the contents of the input capture register, except when exiting STOP mode.
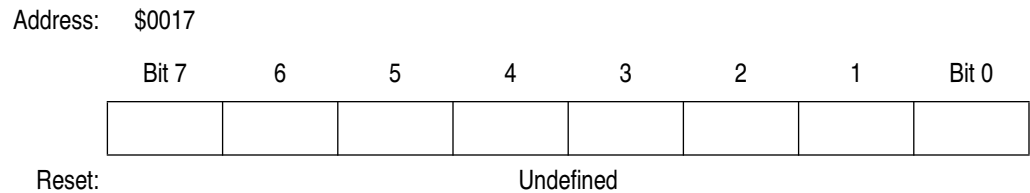
location $17 will not inhibit the compare function. The processor can write to either byte of the output compare register without affecting the other byte. The output level (OLVL) bit is clocked to the output level register whether the output compare flag (OCF) is set or clear. The minimum time required to update the output compare register is a function of the program rather than the internal hardware. Because the output compare flag and the output compare register are not defined at power on, and not affected by reset, care must be taken when initializing output compare functions with software. The following procedure is recommended:

1. write to output compare high to inhibit further compares;
2. read the timer status register to clear OCF (if set);
3. write to output compare low to enable the output compare function.

All bits of the output compare register are readable and writable and are not altered by the timer hardware or reset. If the compare function is not needed, the two bytes of the output compare register can be used as storage locations.

Timer during WAIT mode

All CPU action is suspended, but the timers (core and 16-bit) remain active. An interrupt from either of the timers, if enabled, will cause the MCU to exit WAIT mode.

Timer during STOP mode

In the STOP mode all MCU clocks are stopped, hence the timer stops counting. If STOP is exited by an interrupt the counter retains the last count value. If the device is reset, then the counter is forced to $FFFC. During STOP, if at least one valid input capture edge occurs at the TCAP pin, the input capture detect circuit is armed. This does not set any timer flags nor wake up the MCU. When the MCU does wake up, however, there is an active input capture flag and data from the first valid edge that

occurred during the STOP period. If the device is reset to exit STOP mode, then no input capture flag or data remains, even if a valid input capture edge occurred.

Timer state diagrams

The relationships between the internal clock signals, the counter contents and the status of the flag bits are shown in the following diagrams. It should be noted that the signals labelled 'internal' (processor clock, timer clocks and reset) are not available to the user.

Note: The counter and timer control registers are the only ones affected by power-on or external reset.
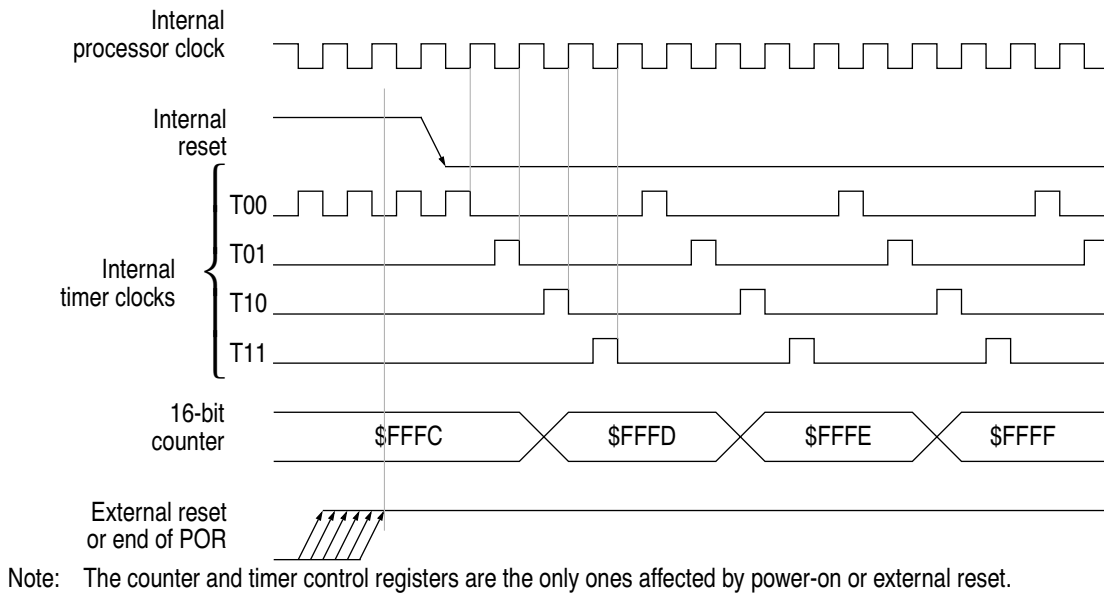
Figure 12. Timer state timing diagram for reset
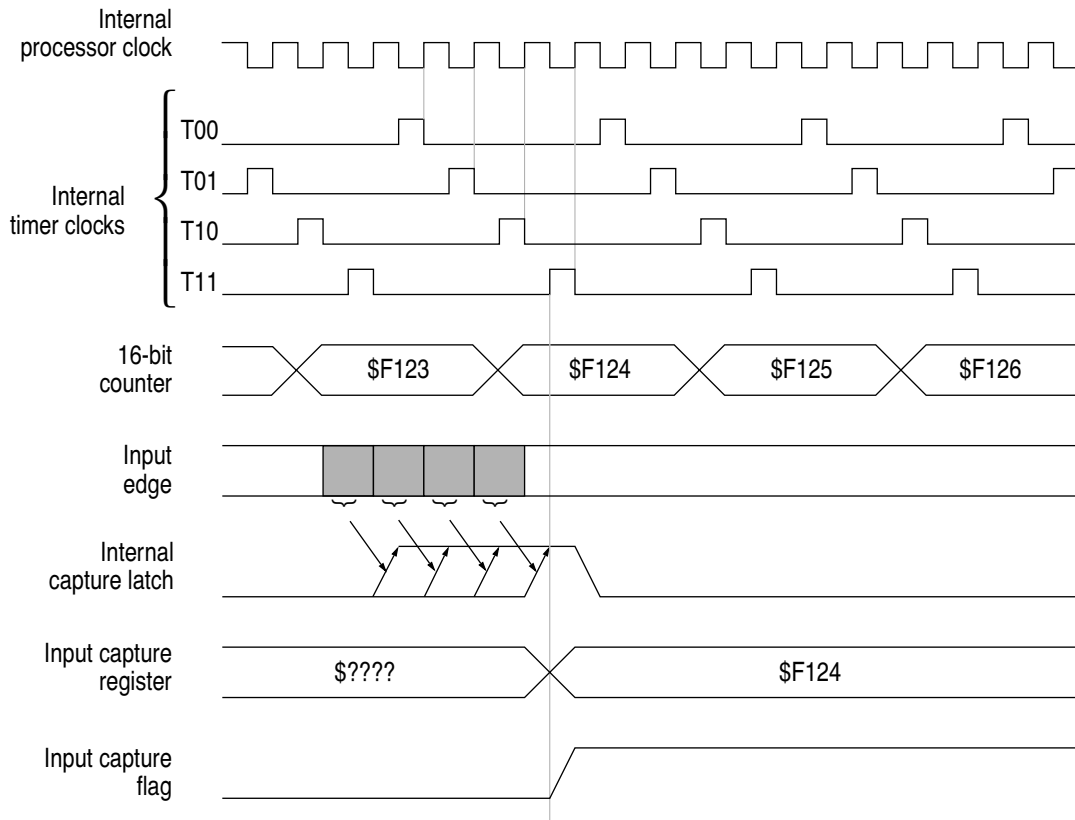
Note: If the input edge occurs in the shaded area from one timer state T10 to the next timer state T10, then the input capture flag will be set during the next T11 state.

Figure 13. Timer state timing diagram for input capture
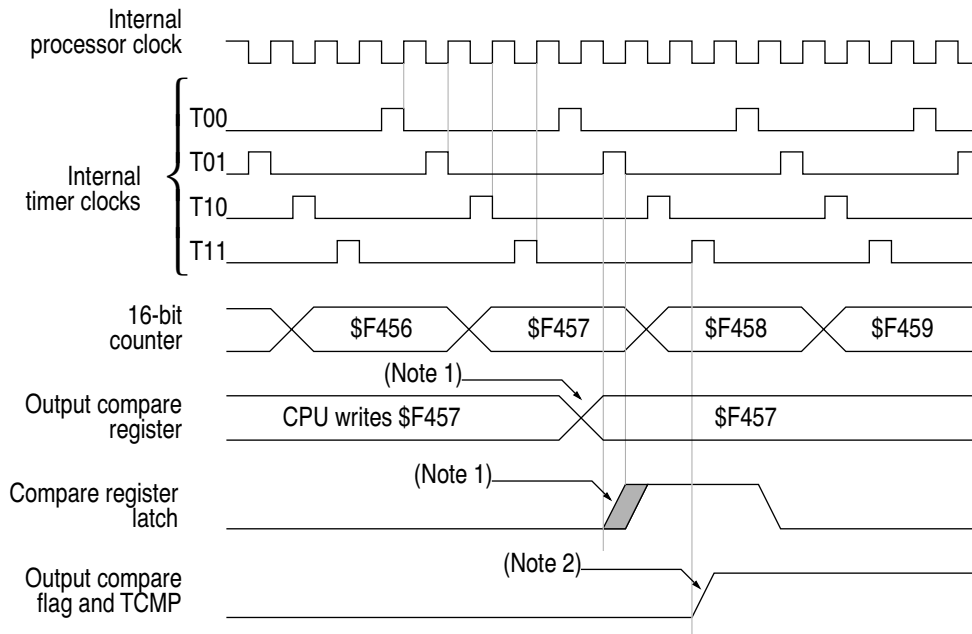
Note: (1) The CPU write to the compare registers may take place at any time, but a compare only occurs at timer state T01. Thus a four cycle difference may exist between the write to the compare register and the actual compare.

Note: (2) The output compare flag is set at the timer state T11 that follows the comparison match ($F457 in this example).

Figure 14. Timer state timing diagram for output compare
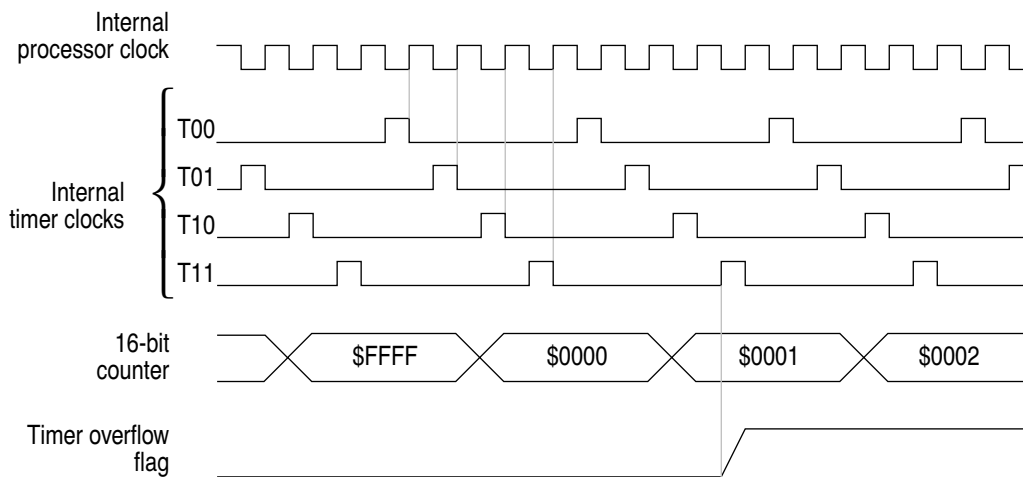
Note: The timer overflow flag is set at timer state T11 (transition of counter from $FFFF to $0000). It is cleared by a read of the timer status register during the internal processor clock high time, followed by a read of the counter low register.

Figure 15. Timer state timing diagram for timer overflow

16-Bit Programmable Timer

Electrical Characteristics

Contents

Introduction

This section contains the electrical specifications and associated timing information for the MC68HC05X4 and *MC68HC705X4*.

NOTE: *Because the MC68HC705X4 has not yet been characterised fully the information and values given for this device are for guidance only.*

Maximum ratings

Figure 1. Maximum ratings

Rating	Symbol	Value	Unit
Supply voltage[1]	V_{DD}	− 0.3 to +7.0	V
Input Voltage: (Ports, OSC1, RESET, RX0/1)	V_{IN}	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	V
Input Voltage: (MDS/TCAP – MC68HC05X4)	V_{IN}	$V_{SS} - 0.3$ to $2V_{DD} + 0.3$	V
Input Voltage: (VPP – MC68HC705X4)	V_{PP}	$V_{SS} - 0.3$ to 16	V
Operating temperature range (standard plastic package)	T_A	T_L to T_H −40 to +125	°C
Storage temperature range	T_{STG}	− 65 to +150	°C
Current drain per pin [2] – excluding VDD and VSS	I_D	28	mA

1. All voltages are with respect to V_{SS}.

2. Maximum current drain per pin is for one pin at a time, limited by an external resistor.

NOTE: *This device contains circuitry designed to protect against damage due to high electrostatic voltages or electric fields. However, it is recommended that normal precautions be taken to avoid the application of any voltages higher than those given in the Maximum Ratings table to this high impedance circuit. For maximum reliability all unused inputs should be tied to either V_{SS} or V_{DD}.*

Thermal characteristics and power considerations

Table 1. Package thermal characteristics

Characteristics	Symbol	Value	Unit
Thermal resistance: – Plastic 28-pin SOIC package	θ_{JA}	60	°C/W

The average chip junction temperature, T_J, in degrees Celsius can be obtained from the following equation:

MC68HC05X4 Rev 1.0

$$T_J = T_A + (P_D \bullet \theta_{JA})$$ [1]

where:

T_A = Ambient temperature (°C)

θ_{JA} = Package thermal resistance, junction-to-ambient (°C/W)

$P_D = P_{INT} + P_{I/O}$ (W)

P_{INT} = Internal chip power = $I_{DD} \bullet V_{DD}$ (W)

$P_{I/O}$ = Power dissipation on input and output pins (user determined)

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = \frac{K}{T_J + 273}$$ [2]

Solving equations [1] and [2] for K gives:

$$K = P_D \bullet (T_A + 273) + \theta_{JA} \bullet P_D^2$$ [3]

where K is a constant for a particular part. K can be determined by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained for any value of T_A by solving the above equations. The package thermal characteristics are shown in Table 1.

Pins	R1	R2	C
PA0–7, PB0–7	3.26kΩ	2.38kΩ	50pF
TX0, TX1	0.25kΩ	0.25kΩ	50pF
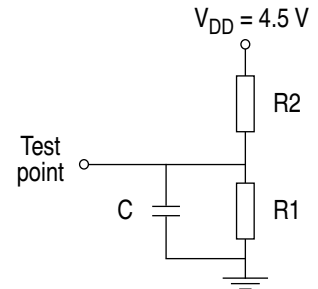
Figure 2. Equivalent test load

DC electrical characteristics

Table 2. DC electrical characteristics (V_{DD} = 5 V)

(V_{DD} = 5.0 V_{DC} ± 10%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H)

Characteristic	Symbol	Min.	Typ.[1]	Max.	Unit
Output voltage I_{LOAD} = −10 µA I_{LOAD} = +10 µA	V_{OH} V_{OL}	V_{DD} − 0.1 —	— —	— 0.1	V V
Output high voltage (I_{LOAD} = 0.8 mA) Ports	V_{OH}	V_{DD} − 0.8	—	—	V
Output low voltage (I_{LOAD} = +1.6 mA) Ports	V_{OL}	—	—	0.4	V
Input high voltage Ports, OSC1, MDS, $\overline{RESET}$	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage Ports, OSC1, MDS, $\overline{RESET}$	V_{IL}	—	—	0.2 x V_{DD}	V
Supply current[2] (Divide-by-2 only) RUN WAIT (CAN active) WAIT (CAN asleep) STOP (CAN active) STOP(CAN asleep)	I_{DD} I_{DD} I_{DD} I_{DD} I_{DD}	— — — — —	3.5 1.4 0.53 0.7 160	5.5 2.6 1.1 1.9 300	mA mA mA mA µA
Supply current[3] (Divide-by-10 only) RUN WAIT (CAN active) WAIT (CAN asleep) STOP (CAN active) STOP (CAN asleep)	I_{DD} I_{DD} I_{DD} I_{DD} I_{DD}	— — — — —	5 3.8 0.53 0.7 160	7.5 4.5 1.1 1.9 300	mA mA mA mA µA
I/O ports hi-Z leakage current Ports, TX0, TX1	I_{IL}	—	–	±10	µA
Input current MDS, OSC1, RX0, RX1, RESET	I_{IN}	—	—	±1	µA
Capacitance All I/Os MDS, RESET	C_{OUT} C_{IN}	— —	— —	12 8	pF pF
MC68HC705X4 EPROM[4]: Programming voltage Programming current Programming Time	V_{PP} I_{PP} t_{PROG}	14.5 1.5 10	15 2 10	15.5 2.5 10	V mA ms

1. Typical values are at mid point of voltage range and at 25°C only.

2. Run (Operating) I_{DD}, Wait I_{DD}: Measured using external square wave clock source (f_{OSC} = 4.4 MHz), all inputs 0.2 V from rail; no dc loads, less than 50pF on all outputs, CL = 20 pF on OSC2, CL = 5pF on XOSC2. Wait, Stop I_{DD}: All ports configured as inputs, V_{IL} = 0.2 V, V_{IH} = V_{DD}–0.2 V. Stop I_{DD} measured with OSC1 = V_{SS}. Wait I_{DD} is affected linearly by the OSC2 capacitance.

3. Run (Operating) I_{DD}, Wait I_{DD}: Measured using external square wave clock source (f_{OSC} = 22 MHz), all inputs 0.2 V from rail; no dc loads, less than 50pF on all outputs, CL = 20 pF on OSC2, CL = 5pF on XOSC2. Wait, Stop I_{DD}: All ports configured as inputs, V_{IL} = 0.2 V, V_{IH} = V_{DD}–0.2 V. Stop I_{DD} measured with OSC1 = V_{SS}. Wait I_{DD} is affected linearly by the OSC2 capacitance.

4. EPROM data is preliminary and cannot be guaranteed.

Table 3. DC electrical characteristics – MCAN module (1)

(V_{DD} = 5.0 V_{DC} $\pm$ 10%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H)

Characteristic	Symbol	Min	Typ	Max	Unit
CAN bus input comparator (RX0–1)					
Input voltage	V_{IN}	– 0.5	—	VDD+0.5	V
Common mode range	CMR	1.5	—	VDD – 1.5	V
Input offset voltage	V_{OFS}	– 20	—	+ 20	mV
Hysteresis	V_{HYS}	5	—	20	mV
VDD/2 generator (VDDH)					
Output voltage difference from VDD/2	DV_{OUT}	– 200	—	+200	mV
(for – 100 < I_{OUT} < +100 µA)					
Output current	I_{OUT}	– 100	—	+100	µA
CAN bus output driver (TX0–1)					
Source current (V_{OUT} = V_{DD} – 1 V)	I_{OH}	– 10	—	—	mA
Sink current (V_{OUT} = 1 V)	I_{OL}	10	—	—	mA

Table 4. DC electrical characteristics – MCAN module (2)

(V_{DD} = 5.0 V_{DC} $\pm$ 2%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H)

Characteristic	Symbol	Min	Typ	Max	Unit
VDD/2 generator (VDDH)					
Output voltage difference from VDD/2	DV_{OUT}	– 180	—	+180	mV
(for – 100 < I_{OUT} < +100 µA)					
Output current	I_{OUT}	– 100	—	+100	µA

Electrical Characteristics

AC electrical characteristics

Table 5. AC electrical characteristics

(V_{DD} = 5.0 V_{DC} ± 10%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H)

Characteristic	Symbol	Min	Typ	Max

Frequency of operation
  Oscillator frequency
  MCAN module bus frequency
  MCU bus frequency
	f_{OSC}
	f_{SCL}
	f_{OP}
		0
		0
		0
			22
			11
			2.2
				MHz

Processor cycle time	t_{CYC}	450	—	ns
MCAN module cycle time	t_{SCL}	90	—	ns
Oscillator clock pulse width	t_{OH}, t_{OL}	18	—	ns
WOI pulse width	t_{WOI}	1.5	—	t_{CYC}
RESET pulse width	t_{RL}	1.5	—	t_{CYC}
Power-on reset delay	t_{PORL}	4064	4064	t_{CYC}

MCAN bus output driver rise
  and fall
  time (TX0, TX1;
  C_{LOAD} = 100 pF)
	trf	—	25	ns

Crystal oscillator start-up time	t_{OXOV}	—	100	ms

NOTE: For the MC68HC705X4 — as with the MC68HC05X4, this device will operate down to dc. However, it should be noted that the power consumption of the EPROM is inversely proportional to frequency. This means that, at frequencies below f_{OP} = 100 kHz, the overall power consumption may exceed the specification limits.

Mechanical Dimensions and Ordering Information

Contents

Freescale Semiconductor, Inc.

Mechanical Dimensions and Ordering Information
For More Information On This Product,
Go to: www.freescale.com

28-pin SOIC package

Figure 1. 28-pin SOIC mechanical dimensions
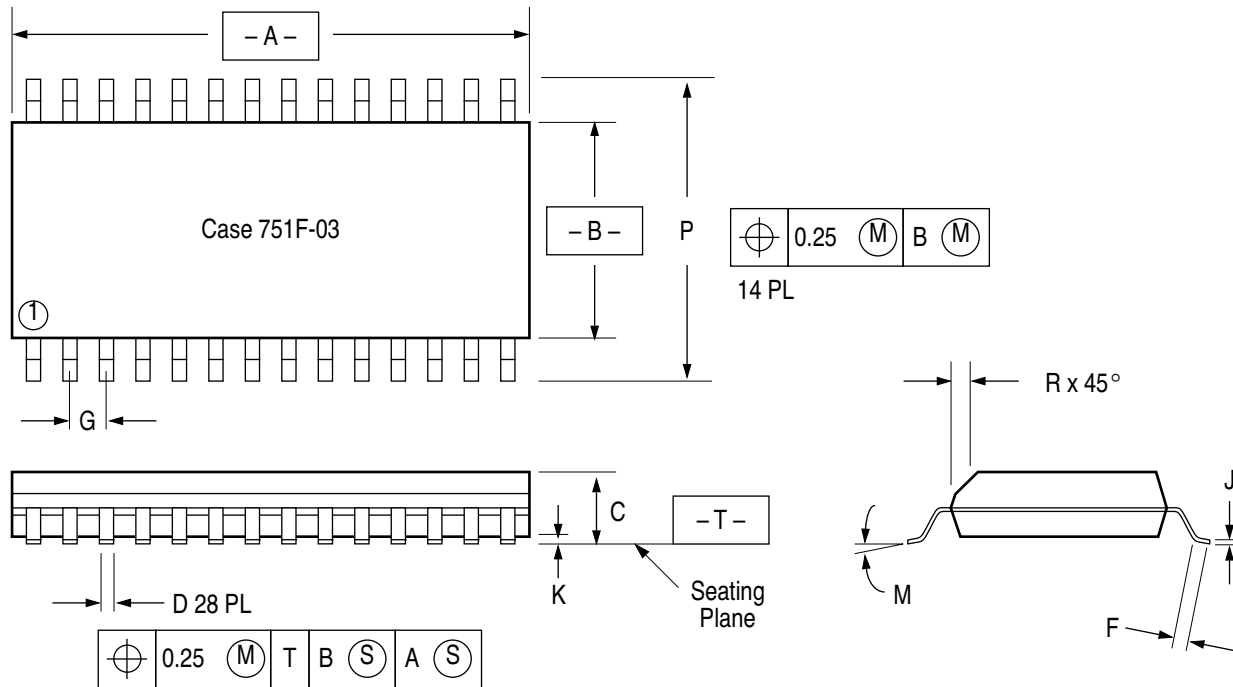
Dim.	Min.	Max.	Notes	Dim.	Min.	Max.
A	17.80	18.05	1.Dimensions 'A' and 'B' are datums and 'T' is a datum surface.	J	0.229	0.317
B	7.40	7.60		K	0.127	0.292
C	2.35	2.65	2.Dimensioning and tolerancing per ANSI Y14.5M, 1982.	M	0°	8°
D	0.35	0.49	3.All dimensions in mm.	P	10.05	10.55
F	0.41	0.90	4.Dimensions 'A' and 'B' do not include mould protrusion.	R	0.25	0.75
G	1.27 BSC		5.Maximum mould protrusion is 0.15 mm per side.	—	—	—

Ordering information

This section describes the information needed to order the MCU.

To initiate a ROM pattern for the MCU, it is necessary to first contact your local field service office, local sales person or Motorola representative. Please note that you will need to supply details such as: mask option

selections; temperature range; oscillator frequency; package type; electrical test requirements; and device marking details so that an order can be processed, and a customer specific part number allocated.

NOTE: *The MC68HC705X4 has no customer specific ROM, or options, and may therefore be ordered as a standard part.*

EPROMs

An 8K byte EPROM programmed with the customer's software (positive logic for address and data) should be submitted for pattern generation. All unused bytes should be programmed to zeros. The EPROM should be clearly labelled, placed in a conductive IC carrier and securely packed.

Verification media

All original pattern media (EPROMs) are filed for contractual purposes and are not returned. A computer listing of the ROM code will be generated and returned with a listing verification form. The listing should be thoroughly checked and the verification form completed, signed and returned to Motorola. The signed verification form constitutes the contractual agreement for creation of the custom mask. If desired, Motorola will program blank EPROMs (supplied by the customer) from the data file used to create the custom mask, to aid in the verification process.

MC order numbers

Device title	Package type	Temperature	Part number
MC68HC05X4 (ROM)	28-pin SOIC	0 to +70°C	MC68HC05X4DW
MC68HC05X4 (ROM)	28-pin SOIC	−40 to +85°C	MC68HC05X4CDW
MC68HC705X4 (EPROM)	28-pin SOIC	0 to +70°C	MC68HC705X4DW
MC68HC705X4 (EPROM)	28-pin SOIC	−40 to +85°C	MC68HC705X4CDW

Freescale Semiconductor, Inc.

Glossary

A — See "accumulator (A)."

accumulator (A) — An 8-bit general-purpose register in the CPU08.   The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation during startup before the PLL locks on a frequency. 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 M68HC08 CPU has 16 addressing modes.

ALU — See "arithmetic logic unit (ALU)."

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.

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)

MC68HC0X4 Rev 1.0

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 — A module in the M68HC08 Family. 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 — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, f_{op}, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

byte — A set of eight bits.

C — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

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.

MC68HC0X4 Rev 1.0

clock generator module (CGM) — A module in the M68HC08 Family. The CGM 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 in the M68HC08 Family that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU08 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.

COP — See "computer operating properly module (COP)."

counter clock — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

CPU — See "central processor unit (CPU)."

CPU08 — The central processor unit of the M68HC08 Family.

CPU clock — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

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 M68HC08 are:

- A (8-bit accumulator)

- H:X (16-bit index register)

- SP (16-bit stack pointer)

- PC (16-bit program counter)

- CCR (condition code register containing the V, H, I, N, Z, and C bits)

CSIC — customer-specified integrated circuit

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

decimal number system — Base 10 numbering system that uses the digits zero through nine.

direct memory access module (DMA) — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

DMA — See "direct memory access module (DMA)."

DMA service request — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

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.

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically 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.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

external interrupt module (IRQ) — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

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.

H — The upper byte of the 16-bit index register (H:X) in the CPU08.

H — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

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.

I — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

index register (H:X) — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

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)."

IRQ — See "external interrupt module (IRQ)."

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.

low voltage inhibit module (LVI) — A module in the M68HC08 Family that monitors power supply voltage.

LVI — See "low voltage inhibit module (LVI)."

M68HC08 — A Motorola family of 8-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.

mask option — A optional microcontroller feature that the customer chooses to enable or disable.

mask option register (MOR) — An EPROM location containing bits that enable or disable certain MCU features.

MCU — Microcontroller unit. See "microcontroller."

memory location — Each M68HC08 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.

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.

monitor ROM — A section of ROM that can execute commands from a host computer for testing purposes.

MOR — See "mask option register (MOR)."

most significant bit (MSB) — The leftmost digit of a binary number.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

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.

MC68HC0X4 Rev 1.0

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 oscillator circuit in which the frequency of the oscillator 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.

MC68HC0X4 Rev 1.0

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 CPU08. 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 in the M68HC08 Family that supports asynchronous communication.

serial peripheral interface module (SPI) — A module in the M68HC08 Family 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 CPU08 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.

TIM — See "timer interface module (TIM)."

timer interface module (TIM) — A module used to relate events in a system to a point in time.

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 — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. 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. Executing an opcode at an unimplemented location causes an illegal address reset.

V —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

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.

X — The lower byte of the index register (H:X) in the CPU08.

Z — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

Index

MC68HC05X4 Rev 1.0

Freescale Semiconductor, Inc.

MC68HC05X4 Rev 1.0

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

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

US & Canada only   http://sps.motorola.com/mfax

Japan   Nippon Motorola Ltd.
                        Tatsumi-SPD-JLDC
                        Toshikatsu Otsuki
                        6F Seibu-Butsuryu Center
                        3-14-2 Tatsumi Koto-Ku
                        Tokyo 135, Japan
                        Phone 03-3521-8315

MC68HC05X4 Rev 1.0

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

Mfax

To access this worldwide faxing service call or contact by electronic mail or the internet:

RMFAX0@email.sps.mot.com

TOUCH-TONE 1-602-244-6609

http://sps.motorola.com/mfax

Motorola SPS World Marketing World Wide Web Server

Use the Internet to access Motorola's World Wide Web server. Use the following URL:

http://design-net.com

Microcontroller Division's Web Site

Directly access the Microcontroller Division's web site with the following URL:

http://design-net.com/csic/CSIC_home.html

MC68HC05X4 Rev 1.0

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

freescale™
semiconductor

MC68HC05X4/D

For More Information On This Product,
Go to: www.freescale.com