Order this document
by MC68HC705E1TS/D

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

# MC68HC705E1

## *Technical Summary*
## 8-Bit Microcontroller Unit

### Introduction

The MC68HC705E1 is a member of the low-cost, high-performance M68HC05 Family of 8-bit microcontroller units (MCUs). The high-density, complementary metal-oxide semiconductor (HCMOS) M68HC05 Family is based on the customer specified integrated circuit (CSIC) design strategy. All MCUs in the family use the popular M68HC05 central processor unit (CPU) and are available with a variety of subsystems, memory sizes and types, and package types.

The MC68HC705E1 is a 28-pin device based on the MC68HC05E1 design. On-chip memory is enhanced with 4112 bytes of user electrically programmable ROM (EPROM) and a bootloader ROM. An on-chip phase-locked loop (PLL) frequency synthesizer can generate the internal clock from a 32.768-kHz oscillator. Other on-chip resources include a custom periodic interrupt circuit for generating real-time interrupts.

Refer to this technical summary for the architecture of the MC68HC705E1 and a brief description of its subsystems and memory space. For MC68HC705E1 evaluation support tools, refer to *Evaluation Products*, Motorola document number BR292/D.

### Features

- Popular M68HC05 CPU
- Memory-Mapped Input/Output (I/O) Registers
- 4112 Bytes of User EPROM
- 368 Bytes of User Static RAM (SRAM)
- 20 Bidirectional I/O Lines
- Fully Static Operation (No Minimum Clock Speed)
- On-Chip Oscillator with Crystal/Ceramic Resonator Connections
- Phase-Locked Loop (PLL) Frequency Synthesizer
- 15-Stage Multifunction Timer
- Programmable Custom Periodic Interrupt Circuit
- Bootloader ROM
- Power-Saving STOP, WAIT, and Data-Retention Modes
- 8 × 8 Unsigned Multiply Instruction
- Illegal Address Reset
- Programmable Edge-Sensitive or Edge- and Level-Sensitive External Interrupt Trigger
- Programmable Computer Operating Properly (COP) Timer
- 28-Pin Dual In-Line Package (DIP)
- 28-Pin Small Outline Integrated Circuit Package (SOIC)

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

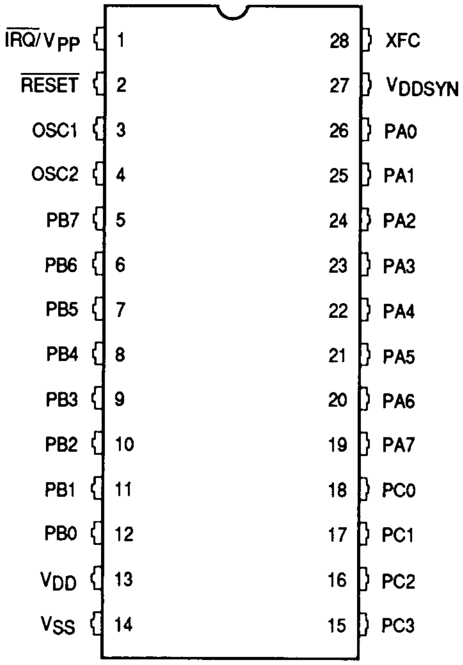
MOTOROLA

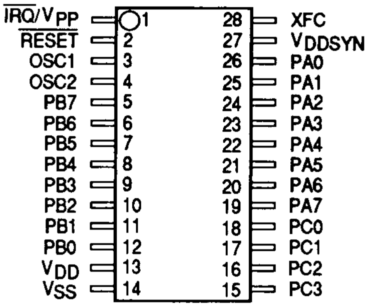
## Table of Contents

## I/O Register Index

## Pin Assignments

The following figures show the pinouts of the DIP/Cerdip and SOIC packages.





**Pin Assignments**

## Order Numbers

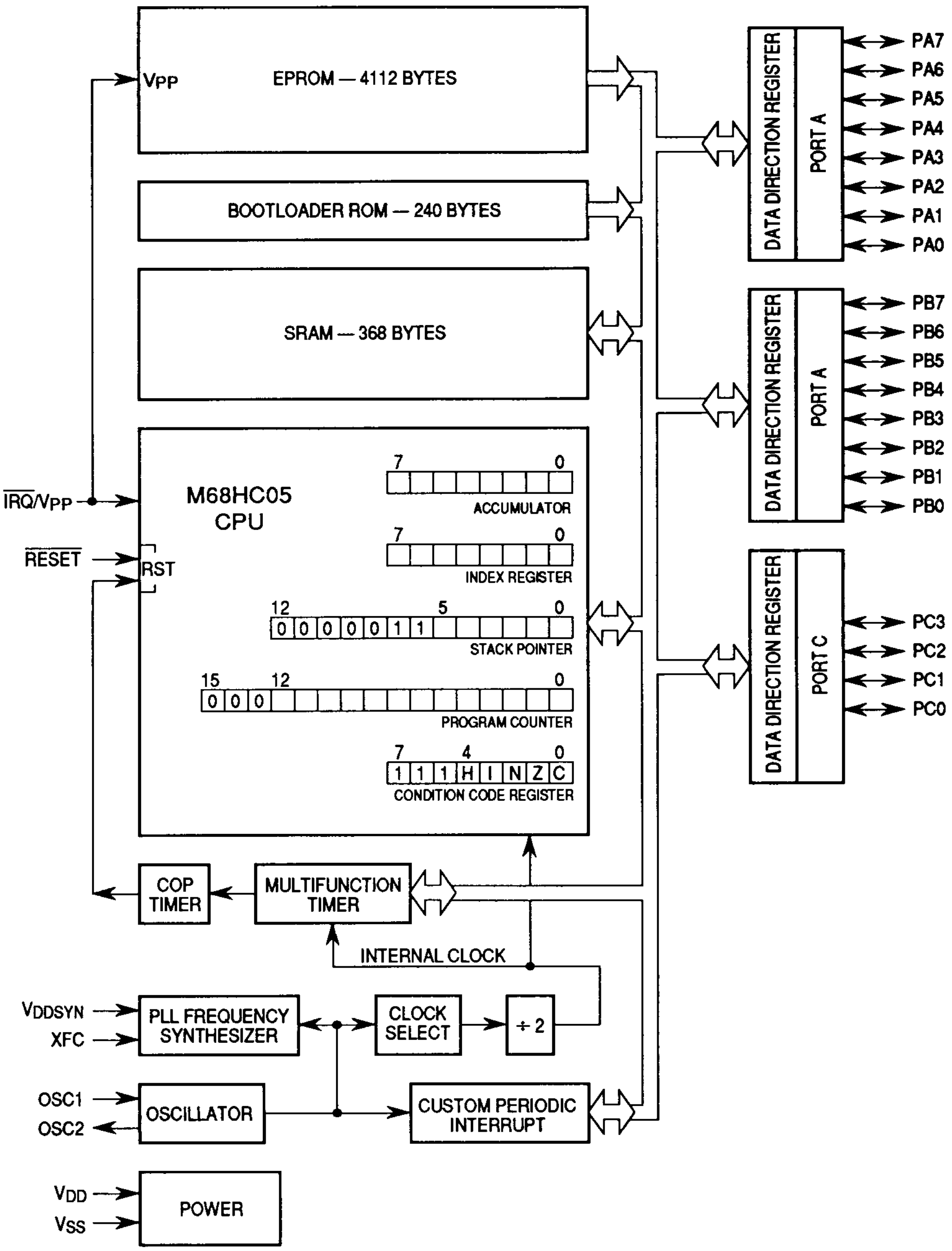
Use the following numbers when ordering these package types.

**MCU Order Numbers**

| Package Type | Order Number |
|---|---|
| 28-Pin DIP | MC68HC705E1P |
| 28-Pin Cerdip | MC68HC705E1S |
| 28-Pin SOIC | MC68HC705E1DW |

## MCU Structure

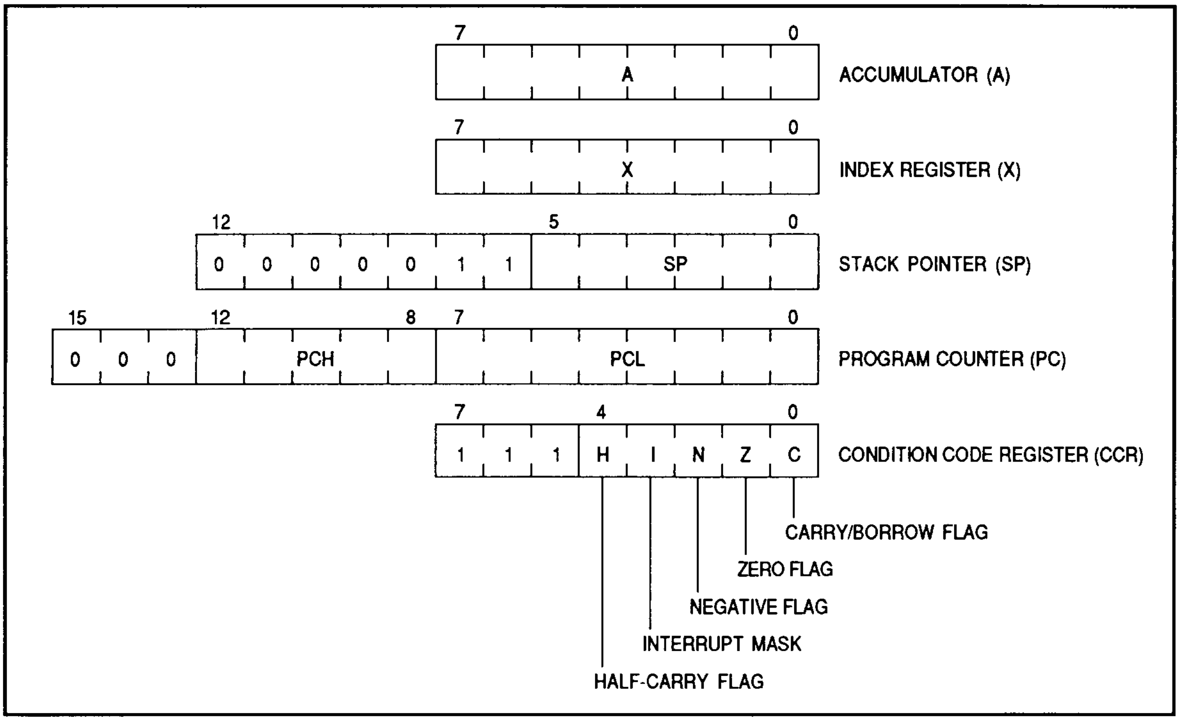
The following figure shows the organization of the MC68HC705E1.



**MCU Structure**

## CPU Registers

Five CPU registers are available to the programmer.



**CPU Registers**

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

The index register contains a value that indicates an operand address in the indexed addressing modes. The index register can also be used as an auxiliary accumulator for temporary storage.

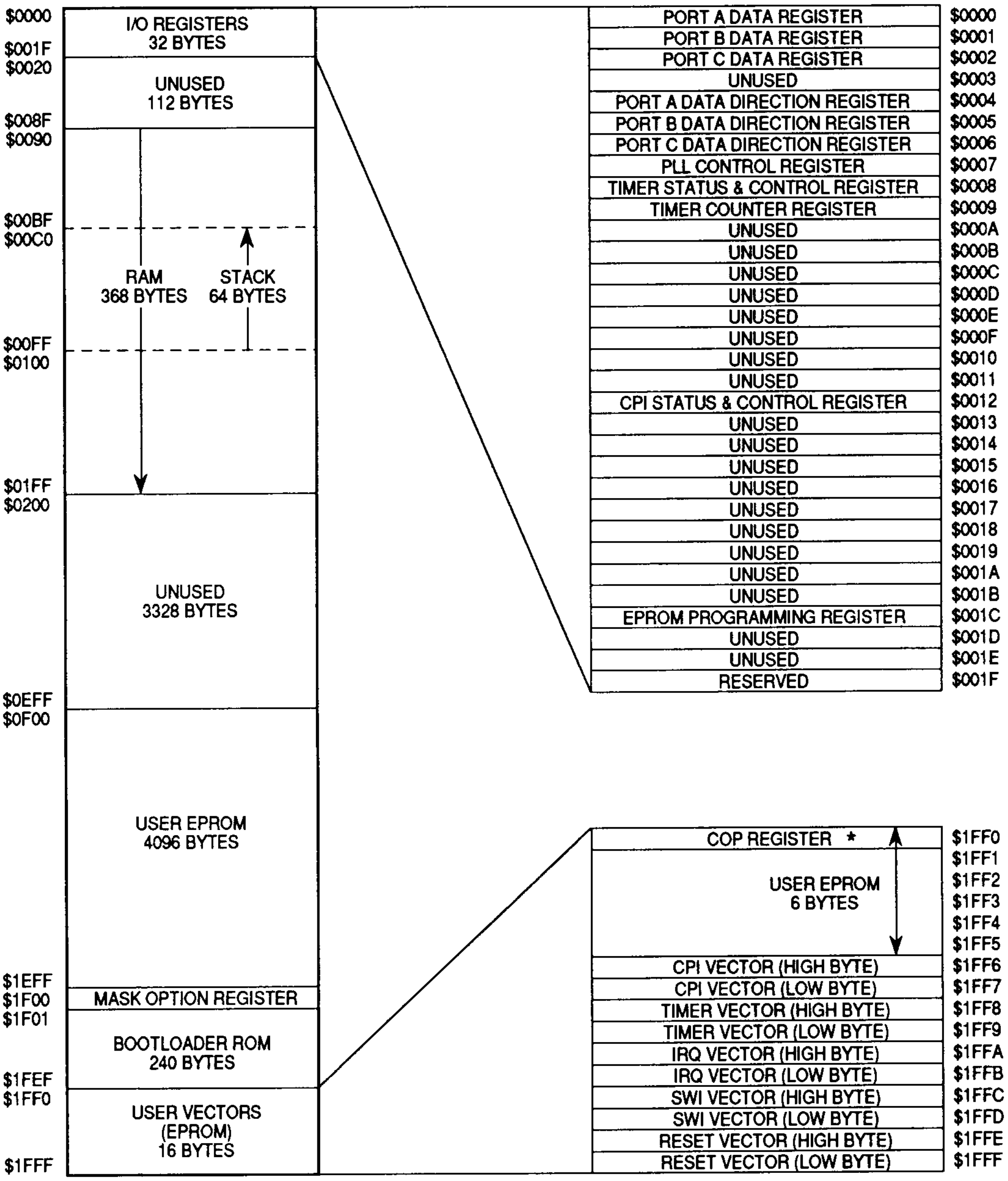
The stack pointer contains the address of the next free location on the stack. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

The program counter contains the address of the next byte for the CPU to fetch.

The condition code register has four status bits that indicate the results of the instruction just executed. A fifth bit is the interrupt mask.

## Memory Map

The MC68HC705E1 CPU can address 8 Kbytes of memory space. The following figures show the organization of the on-chip memory.



| | |
|---|---|
| PORT A DATA REGISTER | $0000 |
| PORT B DATA REGISTER | $0001 |
| PORT C DATA REGISTER | $0002 |
| UNUSED | $0003 |
| PORT A DATA DIRECTION REGISTER | $0004 |
| PORT B DATA DIRECTION REGISTER | $0005 |
| PORT C DATA DIRECTION REGISTER | $0006 |
| PLL CONTROL REGISTER | $0007 |
| TIMER STATUS & CONTROL REGISTER | $0008 |
| TIMER COUNTER REGISTER | $0009 |
| UNUSED | $000A |
| UNUSED | $000B |
| UNUSED | $000C |
| UNUSED | $000D |
| UNUSED | $000E |
| UNUSED | $000F |
| UNUSED | $0010 |
| UNUSED | $0011 |
| CPI STATUS & CONTROL REGISTER | $0012 |
| UNUSED | $0013 |
| UNUSED | $0014 |
| UNUSED | $0015 |
| UNUSED | $0016 |
| UNUSED | $0017 |
| UNUSED | $0018 |
| UNUSED | $0019 |
| UNUSED | $001A |
| UNUSED | $001B |
| EPROM PROGRAMMING REGISTER | $001C |
| UNUSED | $001D |
| UNUSED | $001E |
| RESERVED | $001F |

| | |
|---|---|
| COP REGISTER * | $1FF0 |
| USER EPROM<br>6 BYTES | $1FF1<br>$1FF2<br>$1FF3<br>$1FF4<br>$1FF5 |
| CPI VECTOR (HIGH BYTE) | $1FF6 |
| CPI VECTOR (LOW BYTE) | $1FF7 |
| TIMER VECTOR (HIGH BYTE) | $1FF8 |
| TIMER VECTOR (LOW BYTE) | $1FF9 |
| IRQ VECTOR (HIGH BYTE) | $1FFA |
| IRQ VECTOR (LOW BYTE) | $1FFB |
| SWI VECTOR (HIGH BYTE) | $1FFC |
| SWI VECTOR (LOW BYTE) | $1FFD |
| RESET VECTOR (HIGH BYTE) | $1FFE |
| RESET VECTOR (LOW BYTE) | $1FFF |

* WRITING 0 TO BIT 0 OF $1FF0 CLEARS COP TIMER. READING $1FF0 RETURNS USER EPROM DATA.

**Memory Map**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |  |
|---|---|---|---|---|---|---|---|---|---|
| $0000 | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 | PORTA |
| $0001 | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 | PORTB |
| $0002 | 0 | 0 | 0 | 0 | PC3 | PC2 | PC1 | PC0 | PORTC |
| $0003 | — | — | — | — | — | — | — | — | UNUSED |
| $0004 | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 | DDRA |
| $0005 | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 | DDRB |
| $0006 | 0 | 0 | 0 | 0 | DDRC3 | DDRC2 | DDRC1 | DDRC0 | DDRC |
| $0007 | 0 | BCS | AUTO | BWC | PLLON | VCOTST | PS1 | PS0 | PLLCR |
| $0008 | TOF | RTIF | TOFE | RTIE | 0 | 0 | RT1 | RT0 | TSCR |
| $0009 | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | TCNTR |
| $000A | — | — | — | — | — | — | — | — | UNUSED |
| ⋮ |  |  |  | ⋮ |  |  |  |  | ⋮ |
| $0011 | — | — | — | — | — | — | — | — | UNUSED |
| $0012 | — | CPIF | — | CPIE | — | — | — | — | CPISCR |
| $0013 | — | — | — | — | — | — | — | — | UNUSED |
| ⋮ |  |  |  | ⋮ |  |  |  |  | ⋮ |
| $001B | — | — | — | — | — | — | — | — | UNUSED |
| $001C | 0 | 0 | 0 | 0 | 0 | LATCH | 0 | EPGM | EPROGR |
| $001D | — | — | — | — | — | — | — | — | UNUSED |
| $001E | — | — | — | — | — | — | — | — | UNUSED |
| $001F | — | — | — | — | — | — | — | — | RESERVED |
| $1F00 | — | — | — | — | CPI1 | CPI0 | IRQ | COPE | MOR |
| $1FF0 | — | — | — | — | — | — | — | COPC | COPR |

**I/O Registers**

## I/O Ports

The MCU's 20 bidirectional pins form three I/O ports. The contents of the data direction registers determine whether each I/O pin is an input or an output. The following figures detail the three I/O port data registers and their data direction registers.

**PORTA** — Port A Data Register **$0000**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| RESET: | NOT CHANGED BY RESET | | | | | | | |

Port A is an eight-bit general-purpose I/O port.

**DDRA** — Port A Data Direction Register **$0004**

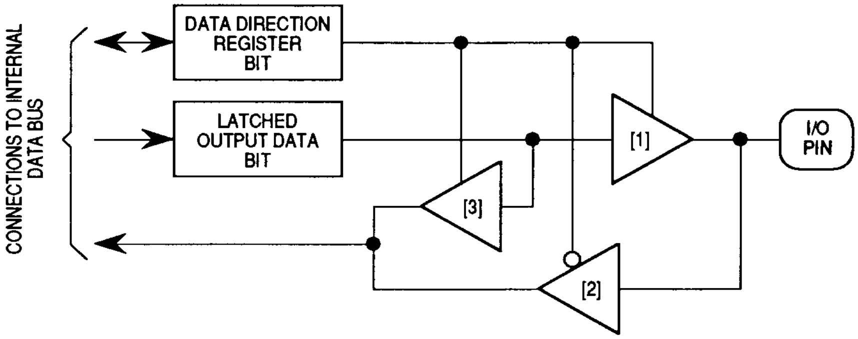
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DDRA7–DDRA0 — Port A Data Direction Bits

These read/write bits determine whether the port A pins are inputs or outputs.

1 = Corresponding port pin configured as output

0 = Corresponding port pin configured as input



[1] This output buffer enables the latched data to drive the pin when DDR bit is 1 (output mode).

[2] This input buffer is enabled when DDR bit is 0 (input mode).

[3] This input buffer is enabled when DDR bit is 1 (output mode).

When an I/O pin is programmed as an output, reading the associated port bit actually reads the value of the output data latch and not the voltage on the pin itself. When a pin is programmed as an input, reading the port bit reads the voltage level on the I/O pin. The output data latch can always be written, regardless of the state of its DDR bit.

To avoid an unintentional state on an output pin, write data to the I/O port data register before setting the corresponding data direction register.

### I/O Pin Functions

| R/$\overline{W}$* | DDR | I/O Pin Functions |
|---|---|---|
| 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, which drives 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. |

*R/$\overline{W}$ is an internal signal.

**PORTB** — Port B Data Register **$0001**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| RESET: | NOT CHANGED BY RESET |  |  |  |  |  |  |  |

Port B is an eight-bit general-purpose I/O port.

**DDRB** — Port B Data Direction Register **$0005**

|  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
|  | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DDRB7–DDRB0 — Port B Data Direction Bits

These read/write bits determine whether the port B pins are inputs or outputs.

1 = Corresponding port pin configured as output

0 = Corresponding port pin configured as input

## **PORTC** — Port C Data Register **$0002**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | PC3 | PC2 | PC1 | PC0 |
| RESET: | NOT CHANGED BY RESET | | | | | | | |

Port C is a four-bit general-purpose I/O port.

## **DDRC** — Port C Data Direction Register **$0006**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DDRC3–DDRC0 — Port C Data Direction Bits

These read/write bits determine whether the port C pins are inputs or outputs.

1 = Corresponding port pin configured as output

0 = Corresponding port pin configured as input

## Resets and Interrupts

There are four reset conditions that force the MCU to a user-defined starting address. There are also four ways to interrupt normal processing to service an external or subsystem event.

### Resets

A reset occurs under the following conditions:

- Power-on reset (POR) — A POR is generated when a positive transition occurs on the $V_{DD}$ pin.
- External reset — A reset is generated when a zero is applied to the $\overline{\text{RESET}}$ pin.
- Computer operating properly (COP) timer reset — When the COP timer is enabled, it resets the CPU if not cleared by a program sequence within a specific period of time. See **COP Timer**.
- Illegal address reset — An attempt to fetch an opcode from an address outside of RAM ($0090–$01FF) or EPROM ($0F00–$1FFF) causes an illegal address reset.

The following internal actions occur on reset:

- All implemented data direction bits are cleared, making all I/O pins inputs.
- The stack pointer is loaded with $FF.
- The interrupt mask (I bit) in the condition code register is set, inhibiting interrupts, and the IRQ request latch is cleared.
- The timer divide-by-four prescaler is cleared.
- The timer is cleared.
- The RT1 and RT0 bits in the timer status and control register are set, selecting the lowest real-time interrupt rate.
- The timer interrupt enable bits, TOFE and RTIE, in the timer status and control register are cleared, disabling timer interrupts.
- The EPROM programming register, EPROGR, is cleared.
- The STOP latch is cleared to enable MCU clocks.
- The WAIT latch is cleared to wake the CPU from the WAIT mode.
- The program counter is loaded with the user-defined reset vector.

### Interrupts

The MCU can be interrupted in the following ways:

- Software interrupt (SWI) — The SWI instruction causes a nonmaskable interrupt to be executed.
- Timer interrupt — A timer interrupt is requested if one of the two timer interrupt flags, TOF or RTIF, is set.
- External interrupt — The CPU recognizes an external interrupt when a user-defined interrupt condition occurs on the $\overline{\text{IRQ}}$ pin. See **Mask Option Register (MOR)**.
- CPI interrupt — A custom periodic interrupt is requested when the CPI counter rolls over from $7FFF to $0000. See **Mask Option Register (MOR)**.

The following actions occur as a result of an interrupt:

- The CPU registers are stored in the stack in the order PCL, PCH, X, A, CCR.
- The interrupt mask (I bit) in the condition code register is set to prevent additional interrupts.
- The program counter is loaded with the appropriate interrupt vector (SWI, external, timer, or CPI).
- The RTI (return from interrupt) instruction causes the CPU registers to be recovered from the stack in the order CCR, A, X, PCH, PCL. Normal processing resumes.

**Reset/Interrupt Vector Assignments**

| Function | Source | Local Mask | Global Mask | Priority (1 = High) | Vector Address |
|---|---|---|---|---|---|
| Reset | COP Timer<br>$\overline{\text{RESET}}$ Pin<br>Power-On<br>Illegal Address | None<br>None<br>None<br>None | COPE Bit<br>None<br>None<br>None | 1<br>1<br>1<br>1 | $1FFE–$1FFF<br>$1FFE–$1FFF<br>$1FFE–$1FFF<br>$1FFE–$1FFF |
| Software Interrupt (SWI) | User Code | None | None | Same Priority As Instruction | $1FFC–$1FFD |
| External Interrupt | $\overline{\text{IRQ}}$ Pin | None | I Bit | 2 | $1FFA–$1FFB |
| Timer Interrupt | TOF Bit<br>RTIF Bit | TOFE Bit<br>RTIE Bit | I Bit<br>I Bit | 3<br>3 | $1FF8–$1FF9<br>$1FF8–$1FF9 |
| CPI Interrupt | CPIF Bit | CPIE Bit | I Bit | 4 | $1FF6–$1FF7 |

### Custom Periodic Interrupt (CPI)

The CPI circuit is based on a 15-bit counter driven by the OSC1 pin. The counter is MOR-programmable for interrupt rates of 0.25 second, 0.5 second, or 1 second. See **Mask Option Register (MOR)**.

**CPISCR** — CPI Status and Control Register **$0012**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | CPIF | 0 | CPIE | 0 | 0 | 0 | 0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CPIF — CPI Flag

This clearable flag is set when the CPI counter rolls over from $7FFF to $0000. Rollover generates a CPI interrupt request if the CPI enable bit, CPIE, is set. Clear the CPIF bit by writing a zero to it. Writing a one to CPIF has no effect.

CPIE — CPI Enable

This read/write bit enables CPI interrupts. Clearing the CPIE bit clears the CPI counter.

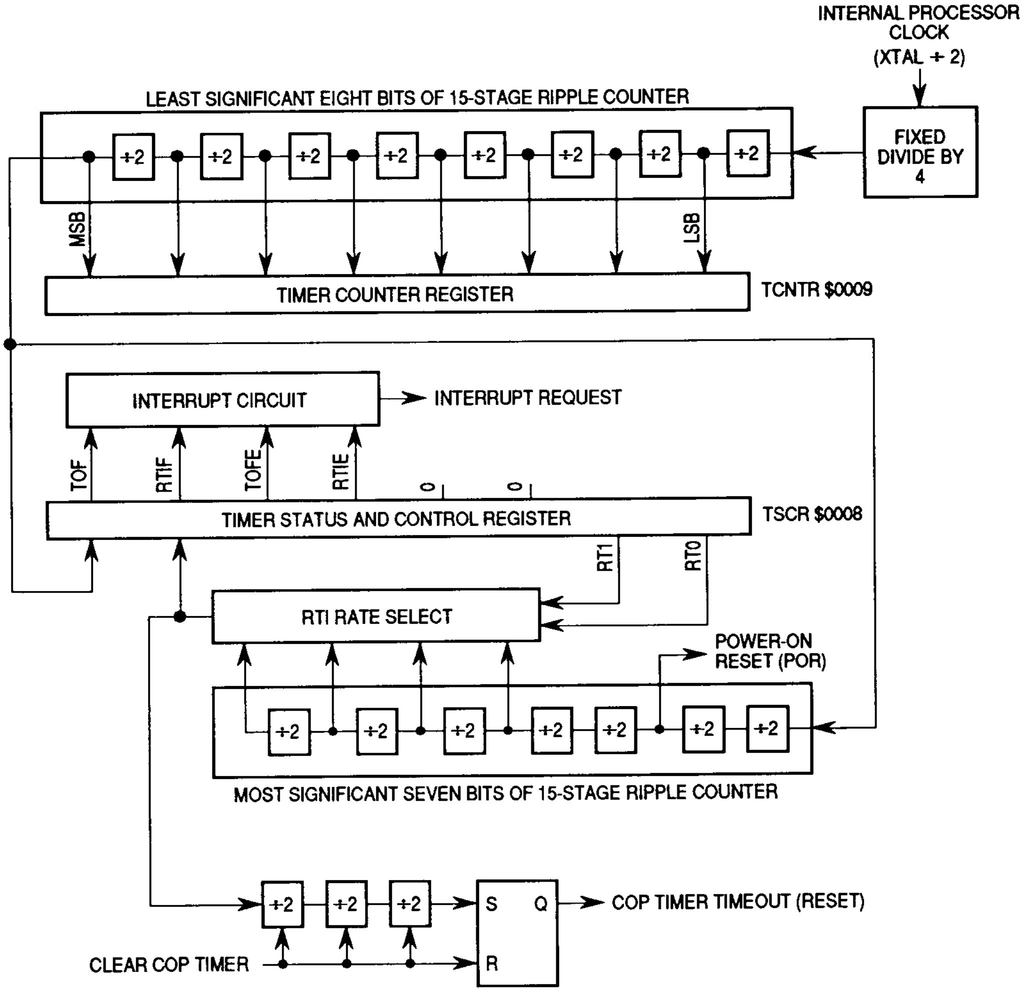
1 = CPI interrupts enabled

0 = CPI interrupts disabled

## Multifunction Timer

The multifunction timer is based on a 15-stage ripple counter. The first eight stages are readable at any time from the timer counter register (TCR). A timer overflow function at the eighth stage allows a timer interrupt every 1024 internal clock cycles. The next four stages lead to the real-time interrupt (RTI) circuit. The RT1 and RT0 bits in the timer status and control register (TSCR) allow a timer interrupt every 16,384; 32,768; 65,536; or 131,072 internal clock cycles.

Three additional counter stages comprise the mask-optional computer operating properly (COP) timer.



INTERNAL PROCESSOR CLOCK (XTAL ÷ 2)
FIXED DIVIDE BY 4
LEAST SIGNIFICANT EIGHT BITS OF 15-STAGE RIPPLE COUNTER
÷2
MSB
LSB
TIMER COUNTER REGISTER
TCNTR $0009
INTERRUPT CIRCUIT
INTERRUPT REQUEST
TOF
RTIF
TOFE
RTIE
0
0
TIMER STATUS AND CONTROL REGISTER
TSCR $0008
RT1
RT0
RTI RATE SELECT
POWER-ON RESET (POR)
MOST SIGNIFICANT SEVEN BITS OF 15-STAGE RIPPLE COUNTER
S
Q
R
COP TIMER TIMEOUT (RESET)
CLEAR COP TIMER

**Multifunction Timer**

**TSCR** — Timer Status and Control Register **$0008**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | TOF | RTIF | TOFE | RTIE | 0 | 0 | RT1 | RT0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

TOF — Timer Overflow Flag

This clearable, read-only bit is set when the first eight stages of the counter roll over from $FF to $00. Rollover also generates a timer interrupt request if the timer overflow enable bit, TOFE, is set. Clear the TOF bit by writing a zero to bit 7 of the timer status and control register. Writing a one to TOF has no effect.

RTIF — Real-Time Interrupt Flag

This clearable, read-only bit is set when the chosen output of the real-time interrupt (RTI) circuit becomes active. The RTIF bit also generates a timer interrupt request if the real-time interrupt enable bit (RTIE) is set. Clear RTIF by writing a zero to bit 6 of the timer status and control register. Writing a one to RTIF has no effect.

TOFE — Timer Overflow Enable

This read/write bit enables timer overflow interrupts.

1 = TOF interrupt enabled
0 = TOF interrupt disabled

RTIE — Real-Time Interrupt Enable

This read/write bit enables RTI interrupts.

1 = RTI interrupts enabled
0 = RTI interrupts disabled

Bits 3 and 2 — Not used

Bits 3 and 2 always read as zeros.

RT1 and RT0 — Real-Time Interrupt Rate Select Bits

These read/write bits change the real-time interrupt rate by selecting one of the four outputs of the real-time interrupt circuit. Because the selected RTI output drives the COP timer, changing the RTI output also changes the counting rate of the COP timer.

### Real-Time Interrupt Rate Selection

| RT1:RT0 | RTI Rate | RTI Period ($f_{op}$ = 2.097 MHz) | COP Timeout Period (-0/+1 RTI Period) | Minimum COP Timeout Period ($f_{op}$ = 2.097 MHz) |
|---|---|---|---|---|
| 0 0 | $f_{op} \div 2^{14}$ | 7.8 ms | 7 × RTI Period | 54.7 ms |
| 0 1 | $f_{op} \div 2^{15}$ | 15.6 ms | 7 × RTI Period | 109.4 ms |
| 1 0 | $f_{op} \div 2^{16}$ | 31.3 ms | 7 × RTI Period | 218.8 ms |
| 1 1 | $f_{op} \div 2^{17}$ | 62.5 ms | 7 × RTI Period | 437.5 ms |

**TCR** — Timer Counter Register **$0009**

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

TCR7–TCR0 — Timer Counter Bits 7–0

These read-only bits show the current value of the first eight stages of the 15-stage multifunction timer.

## COP Timer

Three counter stages at the output of the 15-stage multifunction timer comprise the computer operating properly (COP) timer. The COP timer is a software error detection system that automatically times out if not cleared by a program sequence within a specific time. If the COP timer times out, an internal reset is generated. At an internal clock rate of 2 MHz, minimum timeout periods of 54.7 ms, 109.4 ms, 218.8 ms, and 437.5 ms are selectable. To prevent a COP timeout, write a zero to bit 0 (COPC) of the COP register at location $1FF0. Writing a zero to COPC resets the COP timer and begins the timeout period again.

The state of the COP bit in the mask option register (MOR) determines whether the COP timer output is enabled or disabled. See **Mask Option Register (MOR)**.

**COPR** — Computer Operating Properly Register **$1FF0**

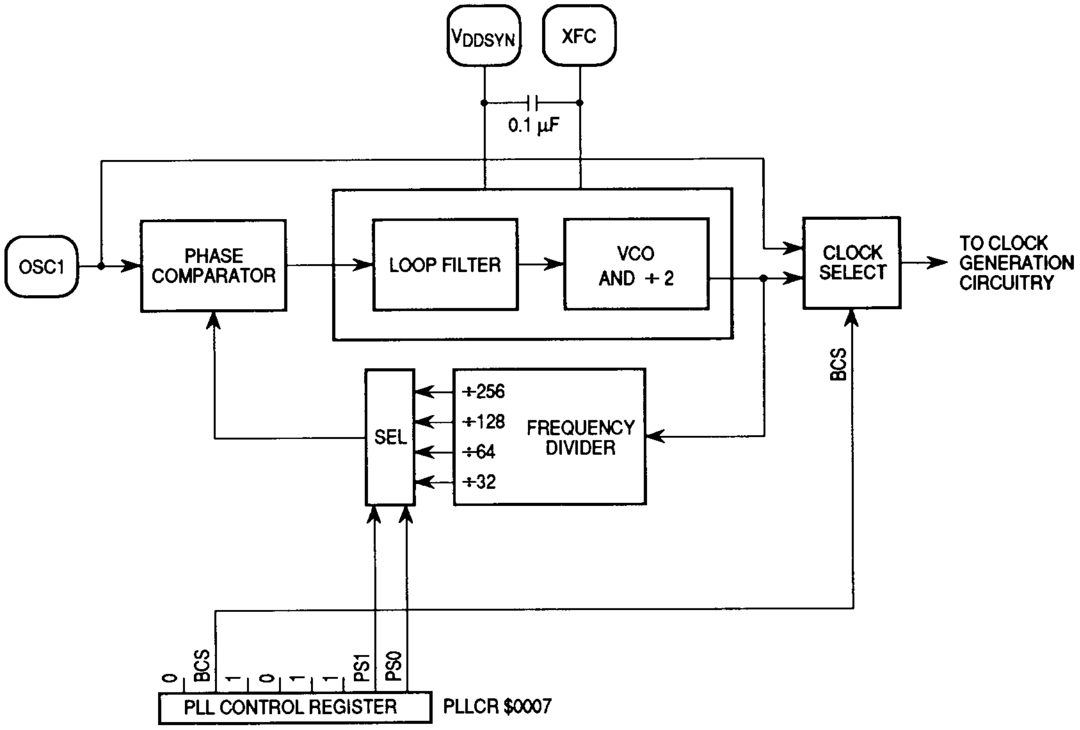
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | COPC |
| RESET: | — | — | — | — | — | — | — | 0 |

COPC — COP Clear

This write-only bit resets the COP timer. Reading address $1FF0 returns the user EPROM data at that address.

## Phase-Locked Loop Frequency Synthesizer

To multiply the frequency of a low-cost external oscillator, the MCU has a programmable phase-locked loop (PLL) frequency synthesizer. The following figure is a block diagram of the PLL frequency synthesizer.



**PLL Frequency Synthesizer**

The phase comparator compares the OSC1 input with the feedback from the frequency divider. The loop filter integrates and amplifies the phase comparator output and controls the voltage controlled oscillator (VCO). The programmable frequency divider selects the feedback frequency for the phase comparator.

The $V_{DDSYN}$ pin provides power to the PLL. $V_{DDSYN}$ should be at the same potential as $V_{DD}$. An external capacitor between the $V_{DDSYN}$ and XFC pins, placed as close to the package as possible, serves to minimize noise.

**PLLCR** — Phase-Locked Loop Control Register **$0007**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | BCS | 1 | 0 | PLLON | 1 | PS1 | PSO |
| RESET: | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

This read/write register controls PLL operation.

BCS — Bus Clock Select

The BCS bit determines whether the PLL output or the OSC1 pin drives the internal clock generator.

1 = PLL drives internal clock generator
0 = OSC1 drives internal clock generator

Bit 5 — Bit 5 must always be set. It is used by Motorola to test the PLL.

Bit 4 — Bit 4 must always be reset. It is used by Motorola to test the PLL.

PLLON — PLL On

The PLLON bit turns on the PLL frequency synthesizer.

1 = PLL frequency synthesizer on
0 = PLL frequency synthesizer off

Bit 2 — Bit 2 must always be set. It is used by Motorola to test the PLL.

**NOTE**

Do not clear the PLLON bit while the PLL frequency synthesizer is driving the internal clock. Clear the BCS bit first; then clear PLLON. Clearing PLLON while the PLL is generating the internal clock causes the internal clock to stop.

PS1, PS0 — PLL Speed Select

These bits select one of the four PLL frequency divider outputs.

**PLL Frequency Selection**

| PS1:PS0 | Internal Clock Frequency (32.768 kHz Crystal) |
|---|---|
| 0 0 | 524 kHz |
| 0 1 | 1.049 MHz (Reset condition) |
| 1 0 | 2.097 MHz (Select only when $V_{DD} \geq 4.5$ V.) |
| 1 1 | 4.194 MHz (For high-speed MCUs only. Select only when $V_{DD} \geq 4.5$ V.) |

## EPROM

Addresses $0F00–$1EFF contain 4096 bytes of EPROM. Sixteen additional EPROM bytes for user-defined interrupt and reset vectors occupy addresses $1FF0–$1FFF.

### EPROM Erasing

The erased state of an EPROM byte is $00. Erase the EPROM with high-density, 2537-angstrom ultraviolet (UV) light before programming it. The recommended dose is 15 Ws/cm$^2$. Position the UV light source 1 inch (2.5 cm) from the EPROM. Do not use a shortwave filter.

### EPROM Programming

The EPROM programming register shown in the next figure contains the control bits for programming the EPROM.

**EPROGR** — EPROM Programming Register **$001C**

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | LATCH | 0 | EPGM |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

LATCH — EPROM Bus Latch

This read/write bit configures the EPROM address bus and data bus for programming.

1 = EPROM address bus and data bus configured for EPROM programming

0 = Buses configured for normal operation

EPGM — EPROM Programming

This read/write bit can be set only when LATCH = 1. EPGM turns on the EPROM programming power and clears automatically when LATCH is cleared.

1 = EPROM programming power switched on

0 = EPROM programming power switched off

Execute the following steps from RAM to program a byte of EPROM:

1. Set the LATCH bit.
2. Write to any EPROM address.
3. Set the EPGM bit for at least 2 ms to apply the programming voltage.
4. Clear the LATCH bit.

### Mask Option Register (MOR)

The mask option register is an EPROM byte at address $1F00 that is programmable only by the bootloader ROM. The mask option register contains four bits that control the following options:

- Custom periodic interrupt rate
- Sensitivity of external interrupt (IRQ) trigger (negative edge sensitive only or both negative edge sensitive and low-level sensitive)
- Computer operating properly (COP) timer (enable or disable)

**MOR** — Mask Option Register **$1F00**

| Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | CPI1 | CPI0 | IRQ | COPE |

This readable EPROM register can be programmed only by using the bootloader ROM to program the EPROM.

CPI1, CPI0 — Custom Periodic Interrupt Select

These bits select one of the four CPI rates.

**CPI Rate Selection**

| CPI1:CPI0 | CPI Interrupt Rate (32.768-kHz Crystal) |
|---|---|
| 0 0 | 1 Second |
| 0 1 | 1 Second |
| 1 0 | 0.5 Second |
| 1 1 | 0.25 Second |

IRQ — Interrupt Request

The IRQ bit determines the sensitivity of the external interrupt trigger.

1 = Edge-sensitive and low level-sensitive external interrupt trigger

0 = Edge-sensitive only external interrupt trigger

COPE — COP Enable

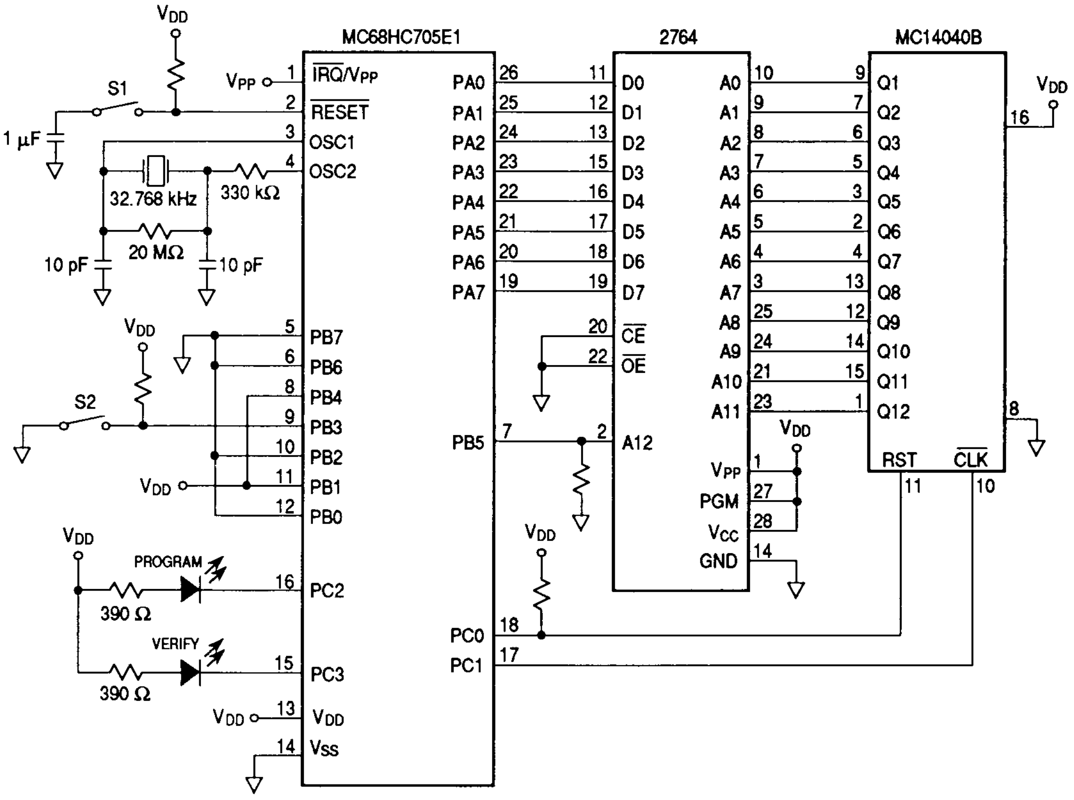
The COPE bit enables the COP timer.

1 = COP timer enabled

0 = COP timer disabled

## Bootloader ROM

The bootloader ROM, located at addresses $0F01–$0FEF, contains a program that can download user code from an external EPROM to the MCU EPROM. The following figure shows the circuit required to copy the contents of a 2764 EPROM to the MCU EPROM. The bootloader circuit includes a 12-bit counter to address the EPROM containing the code to be copied. Use zero-force-insertion sockets for the MCU and the EPROM.



NOTE: Unlabeled resistors are 10 kΩ.

**Bootloader Circuit**

The bootloader copies to the 4-Kbyte space located at user EPROM addresses $0F00–$1EFF and to the 16-byte space at $1FF0–$1FFF. The addresses of the code in the external EPROM must be the same as the MCU addresses to which the code is copied. The bootloader ignores all other addresses. Operation is fastest when unused external EPROM addresses contain $00.

The voltage on the PB3 pin selects the bootloader function, as shown in the following table. The bootloader copies to the EPROM at 4 ms/byte and then performs a verify pass.

**Bootloader Function Selection**

| PB3 | Bootloader Function |
|---|---|
| 1 | Program and Verify |
| 0 | Verify Only |

The bootloader automatically disables the COP timer.

Complete the following steps to use the bootloader ROM:

1. Turn off all power to the circuit.
2. Install the MCU and the EPROM.
3. Select the bootloader function:
   a. Open switch S2 to select the program and verify function.
   b. Close switch S2 to select the verify only operation.
4. Close switch S1 to reset the MCU.
5. Apply the $V_{DD}$ voltage to the circuit.
6. Apply the EPROM programming voltage, $V_{PP}$, to the circuit.
7. Open switch S1 to take the MCU out of reset. During programming, the PROGRAM LED turns on. It turns off when the verification routine begins. If verification is successful, the VERIFY LED turns on. If the bootloader finds an error during verification, it puts the error address on the external address bus and stops running.
8. Close switch S1 to reset the MCU.
9. Remove the $V_{PP}$ voltage.
10. Remove the $V_{DD}$ voltage.