

# Motorola Semiconductor Application Note

---

## AN1286

### MC68HC05C0 Bus Structure Design

By David Yoder and Mark Thompson  
CSIC Applications  
Austin, Texas

#### Introduction

---

This application note explains the basics of designing a system with the MC68HC05C0, a ROM-less expanded bus microcontroller. This document is arranged in two sections. The first section is a question-and-answer format that answers questions that have been asked by people accustomed to designing with single-chip MCUs. The second section has two example schematics that illustrate the use of nonmuxed mode and muxed mode.



## Common Questions

---

Doesn't an expanded bus mean using glue logic?

Due to the carefully chosen chip selects of the MC68HC05C0, glue logic is not required to interface with many common peripherals. The schematic in [Figure 2](#) shows a minimum system with 32 Kbytes of EPROM memory not using glue logic.

What is a minimum system for the MC68HC05C0?

With on-board parallel input/output (I/O), two timers, serial communications interface (SCI), and RAM, the MC68HC05C0 lacks only EPROM memory to execute. Using the  $\overline{A15}$  chip select, an industry standard EPROM, such as a 27C256, can be connected easily in nonmuxed mode. If muxed mode is used, an octal latch is required to latch the low byte of the address bus.

What's the difference between muxed mode and nonmuxed mode?

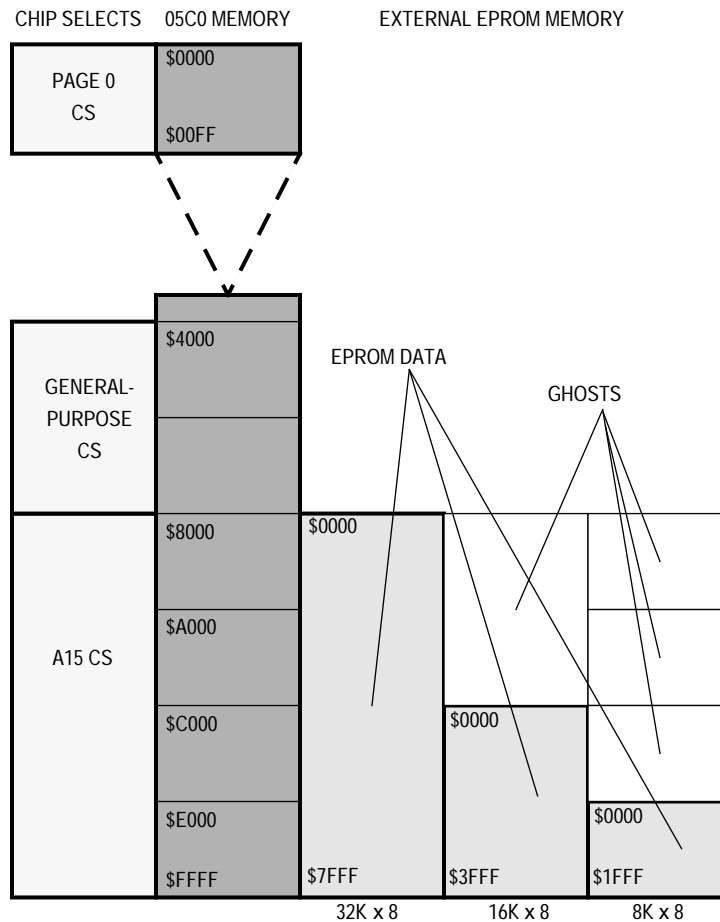
The nonmuxed mode of operation brings the data and address buses out separately. This allows for a pin-to-pin connection with a standard 27Cxxx EPROM chip. The maximum clock frequency is 16 MHz, yielding a bus frequency of 4 MHz.

The muxed mode uses the same eight pins to transmit the low order byte of the address and to transmit or receive data on the data bus. An extra pin ( $AS/\overline{CS2}$ ) is used to latch the low byte of the address into a latch such as a 74HC573. Muxed mode also lowers the maximum oscillator frequency to 8 MHz. This yields a maximum bus frequency of 2 MHz.

What can the chip selects do?

Three chip selects are provided on the MC68HC05C0. (See [Figure 1](#).)

Page 0 chip select is useful for external peripherals. It goes active only for addresses that are mapped as external within the first 256 bytes of the address. This includes \$00, \$04, \$1C–\$29, and \$30–\$3F (clock stretched addresses). In nonmuxed mode, \$03 and \$07 are also mapped external. The  $\overline{CS1}/PB5$  pin can be used as a page 0 chip select in both muxed and nonmuxed modes.



**Figure 1. Comparison of MC68HC05C0 Chip Selects, the Memory Map, and the Memory Maps of Three Differently Sized External EPROMs**

The general-purpose chip select is active from \$4000 to \$7FFF. It is used for other external peripherals that do not require clock stretch or for external volatile memory, such as a 16K x 8 SRAM. This chip select is useful in applications requiring large amounts of data manipulation. The  $\overline{CS1}/PB5$  pin can be used as a general-purpose chip select in both muxed and nonmuxed modes. The  $\overline{CS2}/AS$  pin is always a general-purpose chip select in nonmuxed mode and is always AS in muxed mode.

The  $\overline{A15}$  chip select proves to be the most useful as it selects the entire upper half of the address map. Since memory is required at \$FFF0–\$FFFF for the RESET vectors, it makes sense to use this chip

select for EPROM memory space. Notice that if an EPROM smaller than 32 Kbytes is used, it is ghosted to fill the upper half of address space (see [Figure 1](#)). The  $\overline{A15}$  chip select is, of course, on the  $\overline{A15}$  pin.

What does clock stretching do?

The external addresses \$30–\$3F can be selected as clock-stretch addresses. This mode increases the read and write line low time from  $4 t_{OSC}$  to  $24 t_{OSC}$ . This is useful in a system that requires a high bus frequency, but must access a slow peripheral.

A bus takes a lot of routing. Can it possibly be put on a single layer board?

Yes it can. This is possible because the layout of the address and data bus pins on the MC68HC05C0 in non-muxed mode mirrors that of a 27C256 industry standard EPROM. See the layout of the nonmuxed minimum system in [Figure 3](#).

How is an SRAM connected to the external bus?

Connecting an SRAM without any glue logic to the MC68HC05C0 is possible because of the accommodating bus structure of the MC68HC05C0. For example, the MCM6206C 32K x 8 SRAM can be connected directly to the MC68HC05C0 address and data lines. In this case, an extra port pin could be used on the high order address line so that the entire 32K SRAM could be paged into the 16K space of the general-purpose chip select (\$4000–\$8000).

How can the number of port pins be maximized?

If the slower bus rate of muxed mode is acceptable, this frees eight extra I/O pins. Multiplexing the data and the lowest eight bits of the address leaves port D free for parallel I/O. Also note that the various package types have different numbers of I/O pins. The DIP has 14 in muxed mode. The SDIP package adds two pins and the PLCC adds two more, for a total of 18.

Extra port pins can be added in several ways. One way involves using the SCI to synchronously communicate with a set of serial shift registers. A 74HC164 could be used for output and a 74HC165 for input. Software serial peripheral interface (SPI) routines would need to be written for the

input, as the SCI on the MC68HC05C0 does only synchronous transmits, not receives.

Another way to add extra port pins is to use the external bus. This method takes advantage of the speed of a parallel interface. The bus structure makes it easy to use a part such as the 82C55 without glue logic. In addition, using the page 0 chip select to map the 82C55 into low memory allows the software programmer to take advantage of bit manipulation instructions.

How do I ORG my code and position my reset vectors correctly?

**Figure 1** shows how three common EPROM sizes fit into the memory map. For example, consider the 27C256. This is a 32K x 8 EPROM. When mapped to the  $\overline{A15}$  chip select, it uses the entire top half of the memory map. Therefore, you can begin coding at \$8000 and continue through \$FFF1, where the interrupt vectors begin.

The interrupt vectors reside from \$FFF2 to \$FFFF as shown below:

<u>Interrupt</u>	<u>Vector Address</u>
Keyboard Scan	\$FFF2-\$FFF3
Multi-Function Timer	\$FFF4-\$FFF5
Serial Communication Interface	\$FFF6-\$FFF7
16-Bit Timer	\$FFF8-\$FFF9
External Interrupt	\$FFFA-\$FFFB
Software	\$FFFC-\$FFFD
Reset	\$FFFE-\$FFFF

The RESET vector is always used, making it mandatory that the EPROM memory occupy this space.

It is important to note a difference between the physical location of the code in the EPROM's address space and the mapped location in the part's address space. The physical address of a 32K EPROM will consist of an address space from \$0000 to \$7FFF. However, this address space is mapped into the part's memory at \$8000 to \$FFFF. Similarly, for smaller EPROMs, the physical address space of that EPROM is mapped into the memory map of the part such that the last EPROM address is located at address \$FFFF in the part. The size of the EPROM will determine where its first address will map into the part (see **Figure 1**).

Therefore, absolute code written with ORG statements that match the part's memory map must be programmed into an external EPROM. Use the proper offset on the EPROM programmer such that the first EPROM address is mapped to the proper CPU address and the last EPROM address is mapped to the CPU address of \$FFFF. Always check that the last two locations of the EPROM contain the RESET vector.

What is the purpose of the IRV bit?

The internal read visibility (IRV) bit, located in the system configuration register (CNFGR), is used to control what memory and register accesses are visible on the external bus pins. The reset state of this bit is a zero, which is off. When this bit is off, any access to an internal memory location is not visible on the external bus. Thus, the address bus, the data bus, the chip select, and  $\overline{RD}$  and  $\overline{WR}$  pins will remain in the state they were during the cycle previous to the cycle containing the internal access. This is useful for reducing power consumption and radiated electromagnetic interference (EMI) emissions. The fewer pins that are switching states, the lower the noise and EMI that is generated.

If this bit is turned on, all cycles are displayed on the external pins. Although this is not optimal for noise considerations, it is helpful when debugging software. A logic analyzer or bus state analyzer can be attached to the part and every cycle can be monitored, thus making it easier to find errors.

How do I minimize stop mode current?

EPROMs draw significant current when they are enabled. For this reason, it is not desirable to have the EPROM enabled while the MCU is in stop mode. To minimize current draw, the EPROM should be put in its standby mode by bringing  $\overline{CE}$  high. This is accomplished by executing code out of internal RAM, therefore de-selecting the EPROM.

By default, internal bus cycles are not driven to the external bus. As stated above, this is to reduce radiated noise. This has the effect of leaving the external address bus in the same state it was during the last external cycle. Therefore, the  $\overline{A15}$  chip select is still active. The IRV bit in the configuration register can enable internal cycles to be driven out. With this enabled,  $\overline{A15}$  will be driven to an inactive state while executing in internal RAM. The EPROM will be put into standby mode.

Therefore, before jumping to the STOP instruction in RAM, turn the IRV bit on. See the code sample, C0STOP.MRT, for an example of this method. If desired, the appropriate interrupt handler used to service the interrupt that brings the part out of STOP mode should turn the IRV bit back off.

**What should be considered when interfacing to external memory devices?**

Most users of Motorola M68HC05 microcontrollers are used to accessing on-chip memories, and, therefore, don't worry about timing considerations when accessing these memories. The parts are designed so that all timing considerations are handled in the chip's hardware. When using a microcontroller with an external memory device, these timing considerations must be considered to select a memory device that will work reliably and correctly with the microcontroller.

For example, a typical Motorola SRAM device has connections for the address bus, data bus, and three other signals: chip enable, write enable, and output enable. To connect this device to the MC68HC05C0, the 8-bit data buses would have to be connected, as would all appropriate address lines. If the memory device does not fill the entire memory map of the MC68HC05C0, then some of the MC68HC05C0 upper address lines may not be connected to the memory device. The SRAM chip enable could be connected to the general-purpose chip enable on the MC68HC05C0. The SRAM write enable and output enable signals would then be connected to the MC68HC05C0  $\overline{WR}$  and  $\overline{RD}$  signals respectively.

Connecting the signals correctly isn't the only thing that must be considered. The external memory has minimum requirements for setup and hold times when reading or writing to it. Care must be taken to ensure that the MC68HC05C0 presents the necessary signals (address, data, chip enable,  $\overline{RD}$  and  $\overline{WR}$ ) for a long enough time for the external memory to reliably perform. To ensure compatibility between the devices, consult the MC68HC05C0 specification and the specification for the selected external memory device before designing any application.

## Bus Design

---

### Nonmuxed Mode

**Figure 2** demonstrates the use of a 27C256 32K x 8 EPROM as the program memory in a minimum system. Nonmuxed mode is shown. This mode requires only the MC68HC05C0 and the 27C256, a 2-chip solution.

In **Figure 3**, a layout example also is shown, demonstrating the possibilities of a single layer layout. This is due to the MC68HC05C0 pinout, which is a mirror image of a 27C256.

The simple program, CONME.MRT, checks that a few selected addresses contain the proper data. If this data is correct, the LED on port B pin 0 blinks twice per second. If the test fails, the LED lights steadily. If the LED does not light, something is wrong with the circuit or the EPROM.

### Muxed Mode

**Figure 4** demonstrates the use of a 27C256 32K x 8 EPROM as the program memory in a minimum system. Muxed mode is shown. This mode requires the addition of an HC573 transparent latch, but gives eight extra I/O port pins.

The same program, CONME.MRT, is used to demonstrate the system. It simply checks that a few selected addresses contain the proper data. If this data is correct, the LED on port B pin 0 blinks twice per second. If the test fails, the LED lights steadily. If the LED does not light, something is wrong with the circuit or the EPROM.



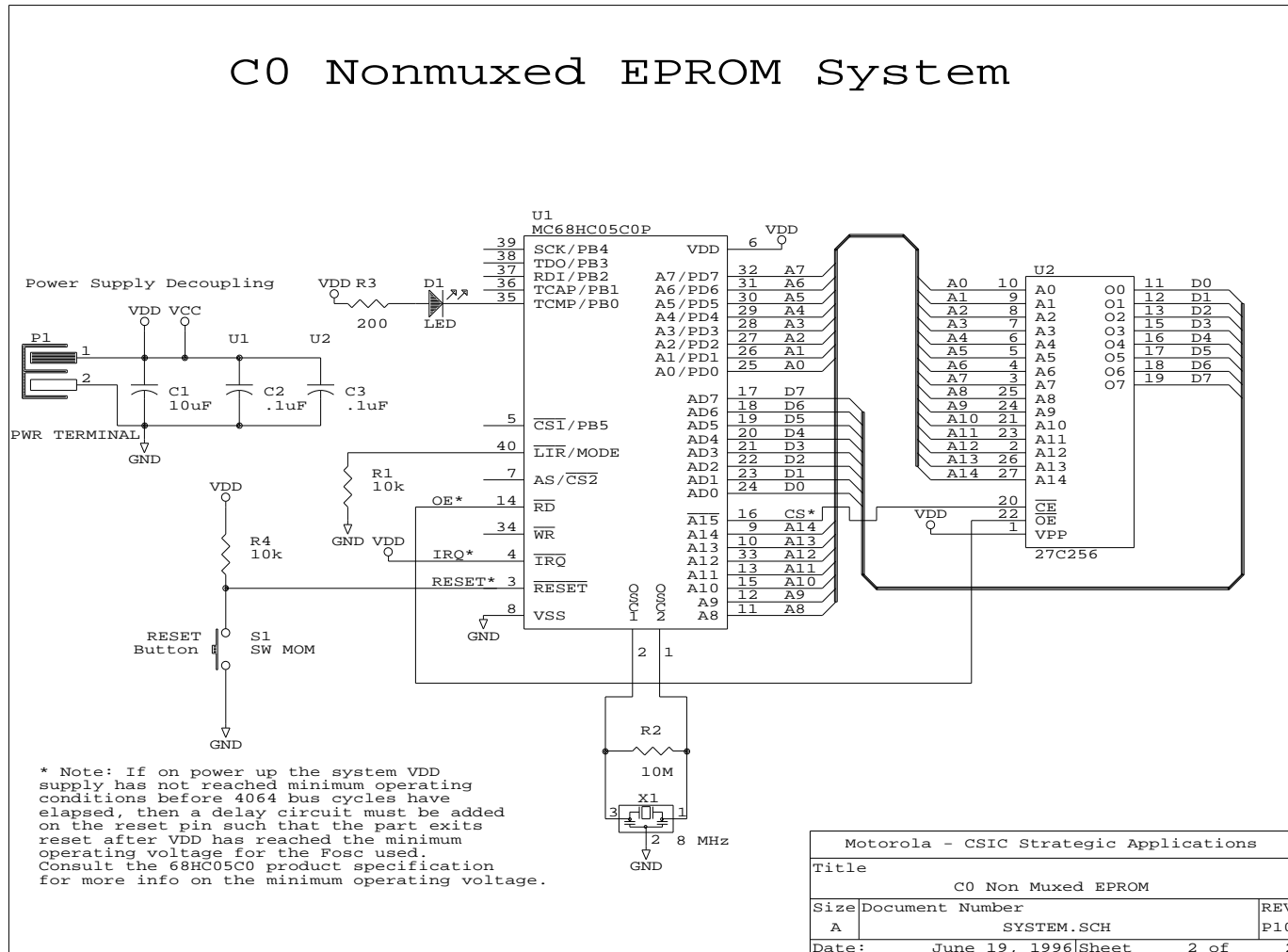


Figure 2. MC68HC05C0 Nonmuxed Mode Schematic

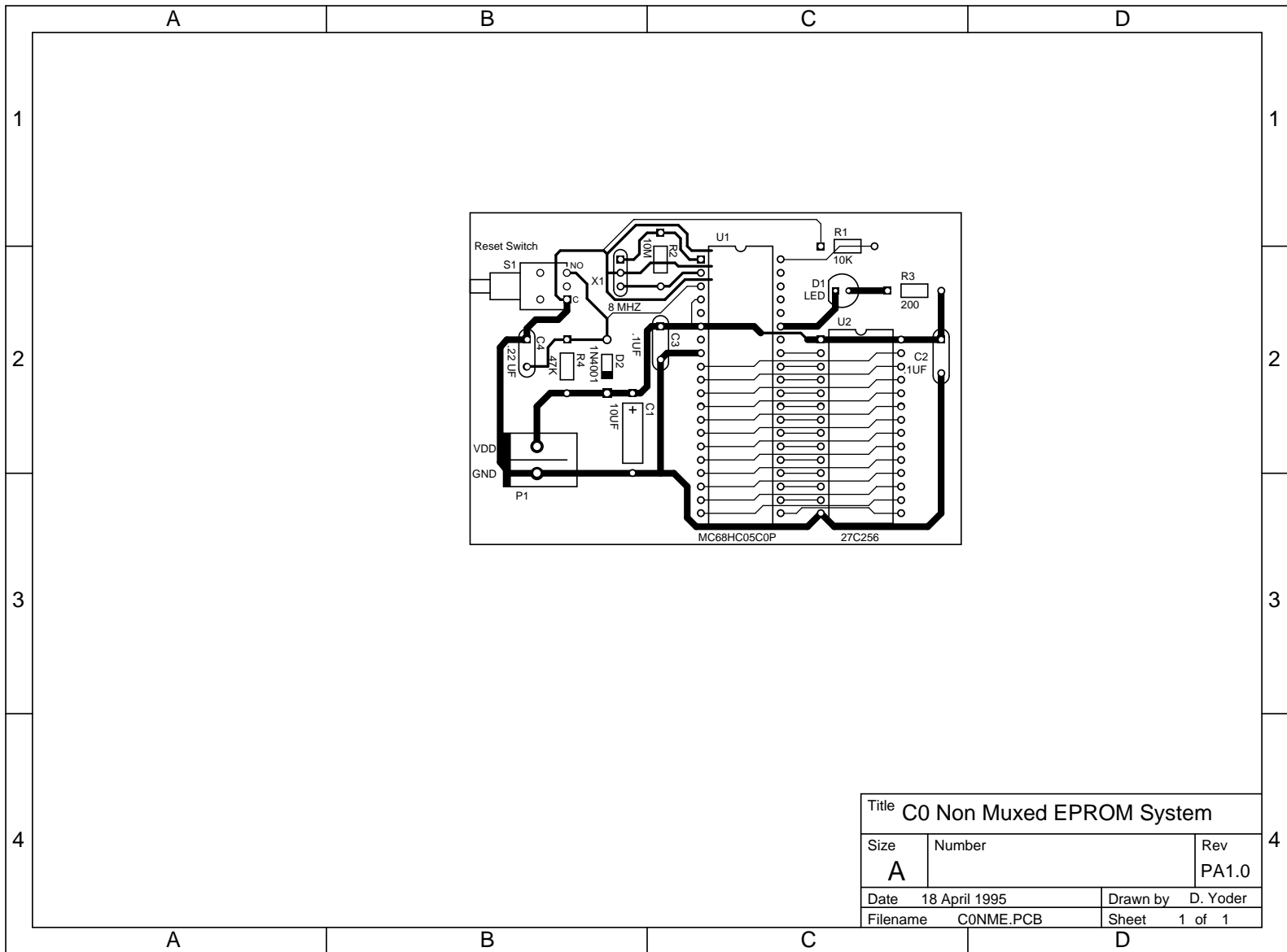


Figure 3. MC68HC05C0 Nonmuxed-Mode Layout

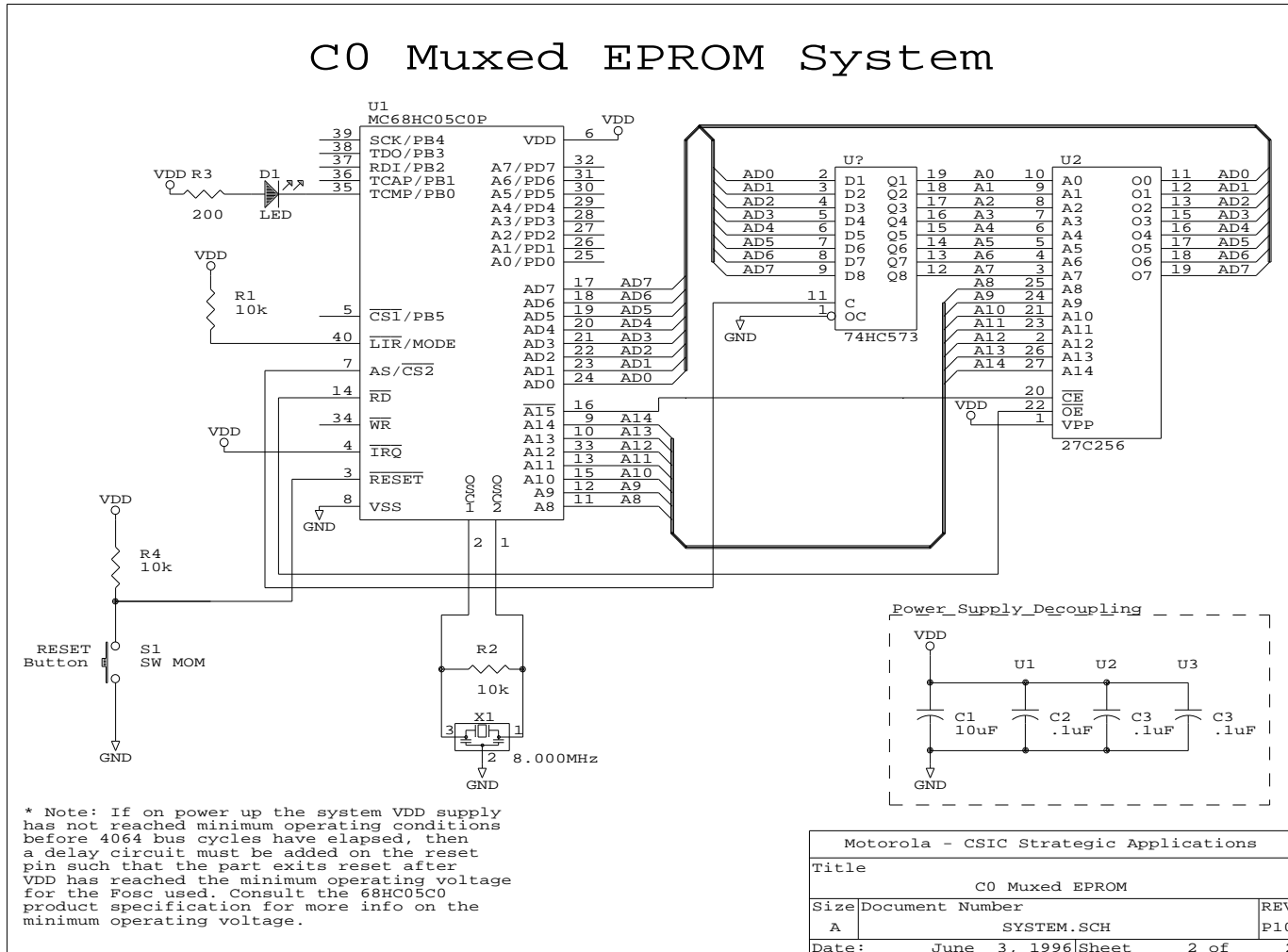


Figure 4. MC68HC05C0 Muxed Mode Schematic

## MC68HC05C0 to 27C256 Interface Test Program

---

```

*****
*****
*
*      CONME - C0 to 27C256 Interface Test Program
*
*****
*
* File Name: CONME.MRT                      Copyright (c) Motorola 1994
*
* Current Revision: 1.0
* Current Release Level: PA
* Current Revision Release Date: 04/20/95
*
* Current Release Written By: David Yoder
*                               Motorola CSIC Applications - Austin, Texas
*
* Assembled Under: IASM05 (P&E Microcomputer Systems, Inc.)   Ver.: 3.02m
*
* Project Folder Name: C0AppN
*
* Part Family Software Routine Works With: HC05
* Part Module(s) Software Routine Works With:
*
* Routine Size (Bytes):      156
* Stack Space Used (Bytes):  0
* RAM Used (Bytes):         0
*
* Global Variables Used:    None
* Subroutines Called:      DelaymS
*
* Full Functional Description Of Routine Design:
*
*      The intent of this routine is to test the external bus connections
*      of the schematic CONME.SCH which is an example of interfacing the
*      MC68HC05C0 to a 27C256 EPROM.
*
*      This is accomplished by accessing addresses that take each of the
*      address lines high individually. Data in these addresses is unique
*      ensuring that the correct address was accessed.
*
*      Use DB directives to define the following data:
*
*      Address:      Data:
*      8001          FE
*      8002          FD
*      8004          FC
*      8008          FB

```

```
*           8010           FA                               *
*           8020           F9                               *
*           8040           F8                               *
*           8080           F7                               *
*           8100           F6                               *
*           8200           F5                               *
*           8400           F4                               *
*           8800           F3                               *
*           9000           F2                               *
*           A000           F1                               *
*           C000           F0                               *
```

```
*           All other address locations in the 27C256 EPROM should be erased *
*           to FF.                                         *
```

```
*           Get data from 8001.                             *
*           Compare to FE                                   *
*           If not equal, branch to error routine          *
*           Repeat for all table entries.                  *
*           Blink LED with 512mS period if correct.       *
*           Error routine: Solid LED if incorrect.        *
```

```
*****
```

```
*
* Motorola reserves the right to make changes without further notice to any *
* product herein to improve reliability, function, or design. Motorola does *
* not assume any liability arising out of the application or use of any *
* product, circuit, or software described herein; neither does it convey any *
* license under its patent rights nor the rights of others. Motorola *
* products are not designed, intended, or authorized for use as components *
* in systems intended for surgical implant into the body, or other *
* applications intended to support life, or for any other application in *
* which the failure of the Motorola product could create a situation where *
* personal injury or death may occur. Should Buyer purchase or use Motorola *
* products for any such intended or unauthorized application, Buyer shall *
* indemnify and hold Motorola and its officers, employees, subsidiaries, *
* affiliates, and distributors harmless against all claims, costs, damages, *
* and expenses, and reasonable attorney fees arising out of, directly or *
* indirectly, any claim of personal injury or death associated with such *
* unintended or unauthorized use, even if such claim alleges that Motorola *
* was negligent regarding the design or manufacture of the part. Motorola *
* and the Motorola Logo are registered trademarks of Motorola Inc. *
*
*
*
```

```

*****
*
*           Part Specific Framework Includes Section
*
* Place the assembler statement ($INCLUDE) to include the part specific
* framework for the target part.
*
*****

```

```

$NOLIST
$INCLUDE 'H05C0.FRK'           ;Include equates for the HC05C0.
$LIST

```

```

*****
*
*           Macros for Main Routine
*
*****

```

```

$MACRO Check                    ;Macro to test one address.
    lda    %1                    ;Get the value.
    cmp    #%2                    ;Check the value.
    bne    CONME_090             ;Branch if not what expected.
$MACROEND

```

```

*****
*
*           Constant Definitions for Main Routine
*
*****

```

```

    org    $8001                 ;Define bytes to test the address
    db     $FE                   ; bus.
    org    $8002
    db     $FD
    org    $8004
    db     $FC
    org    $8008
    db     $FB
    org    $8010
    db     $FA
    org    $8020
    db     $F9
    org    $8040
    db     $F8
    org    $8080
    db     $F7
    org    $8100

```

```
db    $F6
org   $8200
db    $F5
org   $8400
db    $F4
org   $8800
db    $F3
org   $9000
db    $F2
org   $A000
db    $F1
org   $C000
db    $F0
```

```
*****
*
*           Program Initialization
*
* Code needed to initialize processor resources is placed here.
* For any subroutines that have initialization routines which need
* to be executed once out of RESET, a jump (JSR) to the initialization
* section of the subroutine should be placed in this section.
*
*****
```

```
org    $E000
Start:
CONME_Init:  bset    0,portb    ;Preset PortB0 to high(LED off) to
                ; prevent glitches.
                bset    0,ddrb   ;Enable PortB0 to output.
```

```

*****
*
*                               Main Program Loop                               *
*
* This section will contain the main program loop. It's purpose is to check *
* the values placed in the external EPROM and then activate an LED according *
* to whether the data verified correctly or not. If the data verifies         *
* correctly, the LED will blink. If the data fails verify, the LED will stay *
* lit all the time.                                                           *
*
*****

```

Main:

CONME\_Body:

```

Check    $8001, $FE
Check    $8002, $FD
Check    $8004, $FC
Check    $8008, $FB
Check    $8010, $FA
Check    $8020, $F9
Check    $8040, $F8
Check    $8080, $F7
Check    $8100, $F6
Check    $8200, $F5
Check    $8400, $F4
Check    $8800, $F3
Check    $9000, $F2
Check    $A000, $F1
Check    $C000, $F0

CONME_070:    bclr    0,PORTB    ;Pass signal - blinking LED
              lda     #$00      ;Turn on LED
              jsr    DelayMS_Body ;Set up acc for 256mS delay
              ;Delay for 256mS
CONME_080:    bset    0,PORTB    ;Turn off LED
              lda     #$00      ;Set up acc for 256mS delay
              jsr    DelayMS_Body ;Delay for 256mS
              lda     #$00
              sta    COPR       ;Clear the COP Watchdog timer
              bra    CONME_070   ;Infinite loop blinking LED

CONME_090:    bclr    0,PORTB    ;Fail signal - solid LED
CONME_095:    lda     #$00      ;Turn on LED
              sta    COPR       ;Clear the COP Watchdog timer
              bra    CONME_095   ;Infinite loop with solid LED

```



```
*****
*
*           Subroutine Body Includes Section
*
* Place the assembler statement ($INCLUDE) to include any subroutines which
* are called in the main routine.
*
*****
```

```
$INCLUDE 'DelaymS.SRT'           ;Include subroutine for delay.
```

```
*****
*
*           Interrupt and Reset Vectors for Main Routine
*
* Place Vector ORG and FDB statements here. Use the labels defined in the
* part-specific framework file.
*
*****
```

```
org      $FFFE
fdb      START
```

```
*****
*****
*
*           Subroutine DelaymS - Delay for whole number of milliseconds
*
*****
*
* File Name: delayms.SUB           Copyright (c) Motorola 1994
*
* Curent Revision: 1.00
* Current Release Level: PA
* Current Revision Release Date: 04/20/95
*
* Current Release Written By: David Yoder
*                               Motorola CMCU Applications - Austin, Texas
*
* Assembled Under: IASM05 (P&E Microcomputer Systems, Inc.)   Ver.: 3.02m
*
* Documentation File Name: n/a           Revision: n/a
*
* Brief Description of Module Purpose:
*           The routine delays for a whole number of milliseconds.
*           Minimum = 1mS, Maximum = 256mS.
*           Assumes an HC05 CPU with 4MHz crystal, 2MHz bus speed
*
*****
```

```

* Part Family Software Module Works With:  HC05
* Part Module Software Module Works With:  only CPU05 is required
* Part Module Software Module Tested With:  HC705C8A
*
* Calling Sequence: LDA #desired number of milliseconds
*                   JSR DelaymS_Body
*
* Entry Label:      DelaymS_Body
* Module Size (Bytes): 17T bytes
* Stack Space Used (Bytes): 2T bytes (1 jsr stack frame)
* RAM Used (Bytes): 0
* Worst Case Execution (Cycles): 512006T cycles (with $00 in Acc)
* Entry Conditions: # of mS to delay in Acc
* Part Resources Needed: CPU05 running at 2.00MHz bus frequency
* External Variables Used: none
* Subroutines Used: none
* Number of Exit Points: 1
*   Exit Label: DelaymS015
*   Exit Conditions: Acc = 0 after being decremented
*
*
* Full Functional Description of Module Design:
*
* This routine delays operation for a whole number of milliseconds.
* The number of milliseconds to delay is passed in the accumulator
* The routine alters Acc, X and CCR.
* A 4MHz clock (2MHz bus) is assumed.
* The smallest delay is 2011 cycles which occurs when Acc = 1. (1mS)
* The largest delay is 512006 cycles which occurs when Acc = 0. (256mS)
*
* Please note that passing 0 will NOT result in zero delay, but 256mS delay.
*
* The number of milliseconds to delay is passed in the accumulator. The
* routine is formed by two loops. The inner loop (DelaymS020) executes in
* 1989 cycles. The outer loop executes once for each millisecond and adds
* 11 bus cycles each time through the loop. This creates 2000 cycles for
* each millisecond of delay. The RTS used to exit the routine add 06 bus
* cycles to the total time. By my convention, the JSR is not counted in
* this routine, but in the calling routine.
*
* The exact number of cycles for this routine to execute may be calculated
* from:
*
*           cycles = 11 + ( Acc * 2000 )
*
* Upon exit, the accumulator and index register will be zero.
*
*

```

```

*****
*
* Update History:
* Rev:      Author:      Date:      Description of Change:
* ----      -
* PA 1.0    Yoder        04/20/95   Original Release
*
*****
*
* Motorola reserves the right to make changes without further notice to any
* product herein to improve reliability, function, or design. Motorola does
* not assume any liability arising out of the application or use of any
* product, circuit, or software described herein; neither does it convey any
* license under its patent rights nor the rights of others. Motorola
* products are not designed, intended, or authorized for use as components
* in systems intended for surgical implant into the body, or other
* applications intended to support life, or for any other application in
* which the failure of the Motorola product could create a situation where
* personal injury or death may occur. Should Buyer purchase or use Motorola
* products for any such intended or unauthorized application, Buyer shall
* indemnify and hold Motorola and its officers, employees, subsidiaries,
* affiliates, and distributors harmless against all claims, costs, damages,
* and expenses, and reasonable attorney fees arising out of, directly or
* indirectly, any claim of personal injury or death associated with such
* unintended or unauthorized use, even if such claim alleges that Motorola
* was negligent regarding the design or manufacture of the part. Motorola
* and the Motorola Logo are registered trademarks of Motorola Inc.
*
*****
.PAGE
*****
*
* Delay for XmS
*
* This calls the 1mS delay routine below.
* Number of mS in passed through the accumulator.
*
*****
DelaymS_Body    equ    *
DelaymS010     jsr    DelaymS020    ;Jump to 1mS delay counter
               brn    DelaymS010    ;burn 3 bus cycles
               nop    ;burn 2 bus cycles
               deca   ;decrement # of mS to delay
               bne   DelaymS010    ;branch if not done
DelaymS015     rts    ;all done - return to calling routine

```

```

*****
*
* Delay for 1mS
*
* Takes 1989 cycles to execute.
*
*****
DelaymS020      ldx      #$F7          ;Load delay into X
DelaymS030      decx     ;Decrement delay
                nop       ;burn 2 bus cycles
                bne     DelaymS030    ;Branch in not done
                rts
*****

*****
*
*      COSTOP - STOP mode example for external bus HC05C0
*
*****
*
* File Name: COSTOP.MRT                      Copyright (c) Motorola 1994
*
* Current Revision: 1.0
* Current Release Level: PA
* Current Revision Release Date: 04/20/95
*
* Current Release Written By: David Yoder
*                               Motorola CSIC Applications - Austin, Texas
*
* Assembled Under: IASM05 (P&E Microcomputer Systems, Inc.)   Ver.: 3.02m
*
* Project Folder Name: C0AppN
*
* Part Family Software Routine Works With: HC05
* Part Module(s) Software Routine Works With:
*
* Routine Size (Bytes):      26
* Stack Space Used (Bytes):  0
* RAM Used (Bytes):         0
*
* Global Variables Used:    None
* Subroutines Called:      DelaymS
*

```

```
* Full Functional Description Of Routine Design: *
*
*   The intent of this routine is to test the STOP mode of the C0 with *
*   respect to chip selects. That is, the chip selects should be *
*   when a STOP instruction is executed from internal RAM. *
*
*   This will be tested by: *
*       Copying a STOP and RTS instruction to RAM. *
*       Enabling Internal Read Visibility. *
*       Enable STOP Instruction. *
*       JSR to the STOP instruction. *
*       Measure the level of the A15_ pin. *
*
*****
*
* Motorola reserves the right to make changes without further notice to any *
* product herein to improve reliability, function, or design. Motorola does *
* not assume any liability arising out of the application or use of any *
* product, circuit, or software described herein; neither does it convey any *
* license under its patent rights nor the rights of others. Motorola *
* products are not designed, intended, or authorized for use as components *
* in systems intended for surgical implant into the body, or other *
* applications intended to support life, or for any other application in *
* which the failure of the Motorola product could create a situation where *
* personal injury or death may occur. Should Buyer purchase or use Motorola *
* products for any such intended or unauthorized application, Buyer shall *
* indemnify and hold Motorola and its officers, employees, subsidiaries, *
* affiliates, and distributors harmless against all claims, costs, damages, *
* and expenses, and reasonable attorney fees arising out of, directly or *
* indirectly, any claim of personal injury or death associated with such *
* unintended or unauthorized use, even if such claim alleges that Motorola *
* was negligent regarding the design or manufacture of the part. Motorola *
* and the Motorola Logo are registered trademarks of Motorola Inc. *
*
*
*****
*
*           Part Specific Framework Includes Section *
*
* Place the assembler statement ($INCLUDE) to include the part specific *
* framework for the target part. *
*
*****

$NOLIST
$INCLUDE 'H05C0.FRK'           ;Include equates for the HC05C0.
$LIST
```

```
*****
*
*          RAM Variable Definition
*
*****

RAM_Subroutine    org    RAM
                  rmb    2          ;Reserve two bytes for STOP subroutine.

*****
*
*          Program Initialization
*
* Code needed to initialize processor resources is placed here.
* For any subroutines that have initialization routines which need
* to be executed once out of RESET, a jump (JSR) to the initialization
* section of the subroutine should be placed in this section.
*
*****

COSTOP_Init:     org    $E000
                  LDA    #$FF          ;Write all available port pins to
                  STA    PORTB         ; output high. This will reduce STOP
                  STA    PORTD         ; current for measurement.
                  STA    DDRB
                  STA    DDRD
```

```

*****
*
*                               Main Program Loop                               *
*
* This section will contain the main program loop. It's purpose is to set *
* up the CONFIG register to enable STOP mode and the Internal Read *
* Visibility (IRV) option. It then loads the RAM with a STOP instruction and *
* an RTS instruction to create a subroutine in RAM. Since our purpose is *
* to disable the EPROM to minimize current draw in STOP mode, we want to run *
* this code from RAM. When this subroutine is executed, the CE of the EPROM *
* is brought high, thus disabling it. This example program assumes there is *
* some method of supplying the part an interrupt, thus bringing the part out *
* of STOP mode. Upon exiting STOP mode, the main loop simply branches back *
* to the beginning, creating an infinite loop. *
*
*****

```

```

COSTOP_Body:
    lda    #$F3                ;Enable STOP, IRV on, others at
                                ; POR default.
    sta    CNFGR                ;Write to configuration register.

    lda    #$8E                ;Write STOP instruction to RAM.
    sta    RAM_Subroutine
    lda    #$81                ;Write RTS instruction to RAM.
    sta    RAM_Subroutine+1
    jsr    RAM_Subroutine      ;Jump to STOP subroutine.
    bra    COSTOP_Body        ;Infinite loop.

```

```

COSTOP_ISR:
    rti                ;Just return from interrupt.

```

```

*****
*
*                               Interrupt and Reset Vectors for Main Routine *
*
* Place Vector ORG and FDB statements here. Use the labels defined in the *
* part-specific framework file. *
*
*****

```

```

    org    IRQ_INT
    dw    COSTOP_ISR

    org    RESET
    fdb    COSTOP_Init

```

```

*****

```

```

*****
*
*           HC705J1A Part Specific FrameWork
*
*****
*
* File Name: H05C0.FRK           Copyright (c) Motorola 1994
*
* Curent Revision: 1.1
* Current Release Level: ES
* Current Revision Release Date: 04/11/95
*
* Current Release Written By: David Yoder
*           Motorola CMCU Applications - Austin, Texas
*
* Assembled Under: IASM05 (P&E Microcomputer Systems, Inc.)
*           Version: 2.2, 3.01m, 3.02m
*           CASM05 (P&E Microcomputer Systems, Inc.)
*           Version: 3.02, 3.04
*
* Documentation File Name:           Revision:
*
* Program Description:
*           This FrameWork contains part specific information such as
*           register equates, memory map specifics, and reset/interrupt
*           vector addresses. This file is meant to be used in conjunction*
*           with the Austin CSIC applications group's software library.
*
*           For bit equates:
*           Use bit names without a dot in BSET..BRCLR
*           Use bit names followed by a dot in expressions such as
*           #ELAT.+EPGM. to form a bit mask
*
*****
*
* Update History:
* Rev:      Author:      Date:      Description of Change:
* ----      -
* ESS 1.0   Yoder       12/21/94   Original Release
*
*****
*
* Motorola reserves the right to make changes without further notice to any
* product herein to improve reliability, function, or design. Motorola does
* not assume any liability arising out of the application or use of any
* product, circuit, or software described herein; neither does it convey any
* license under its patent rights nor the rights of others. Motorola
* products are not designed, intended, or authorized for use as components
* in systems intended for surgical implant into the body, or other
* applications intended to support life, or for any other application in

```



\* which the failure of the Motorola product could create a situation where \*  
\* personal injury or death may occur. Should Buyer purchase or use Motorola \*  
\* products for any such intended or unauthorized application, Buyer shall \*  
\* indemnify and hold Motorola and its officers, employees, subsidiaries, \*  
\* affiliates, and distributors harmless against all claims, costs, damages, \*  
\* and expenses, and reasonable attorney fees arising out of, directly or \*  
\* indirectly, any claim of personal injury or death associated with such \*  
\* unintended or unauthorized use, even if such claim alleges that Motorola \*  
\* was negligent regarding the design or manufacture of the part. Motorola \*  
\* and the Motorola Logo are registered trademarks of Motorola Inc. \*

\*\*\*\*\*  
\*\*\*\*\*  
#Page

\*\*\*\*\*  
\*\*\*\*\*  
\*  
\* Register Equates \*  
\*  
\*\*\*\*\*

PORTB	equ	\$01	; Port B byte address
PORTC	equ	\$02	; Port C byte address
PORTD	equ	\$03	; Port D byte address
DDRB	equ	\$05	; Port B Data Direction Register
DDRC	equ	\$06	; Port C Data Direction Register
DDRD	equ	\$06	; Port D Data Direction Register
TSCR	equ	\$08	; Timer Status and Control Register
TCR	equ	\$09	; Timer-Counter Register
*TCR	equ	\$0A	; Timer Control Register
*TSR	equ	\$0B	; Timer Status Register
*ICR_MSB	equ	\$0C	; Input Capture Register - MSB
*ICR_LSB	equ	\$0D	; Input Capture Register - LSB
*OCR_MSB	equ	\$0E	; Output Compare Register - MSB
*OCR_LSB	equ	\$0F	; Output Compare Register - LSB
*Timer_MSB	equ	\$10	; Timer Count Register - MSB
*Timer_LSB	equ	\$11	; Timer Count Register - LSB
*AltTimer_MSB	equ	\$12	; Alt. Timer Count Register - MSB
*AltTimer_LSB	equ	\$13	; Alt. Timer Count Register - LSB
BAUD	equ	\$0D	; SCI BAUD Rate Register
SCCR1	equ	\$0E	; SCI Control Register 1
SCCR2	equ	\$0F	; SCI Control Register 2
SCSR	equ	\$10	; SCI Status Register
SCDAT	equ	\$11	; SCI Data Register
CNFGFR	equ	\$19	; Configuration Register
COPR	equ	\$FFFF	; COP Reset Register

\$PAGE

```
*****
*
*                               Bit Equates                               *
*
* Use bit names without a dot in BSET..BRCLR                               *
* Use bit names followed by a dot in expressions such as #ELAT.+EPGM. to  *
* form a bit mask                                                         *
*****
* Port A Data Register (PORTA -> $00)
PA0    equ    0        ; Port A Bit 0
PA1    equ    1        ; Port A Bit 1
PA2    equ    2        ; Port A Bit 2
PA3    equ    3        ; Port A Bit 3
PA4    equ    4        ; Port A Bit 4
PA5    equ    5        ; Port A Bit 5
PA6    equ    6        ; Port A Bit 6
PA7    equ    7        ; Port A Bit 7
PA0.   equ    $01     ; Port A Bit 0
PA1.   equ    $02     ; Port A Bit 1
PA2.   equ    $04     ; Port A Bit 2
PA3.   equ    $08     ; Port A Bit 3
PA4.   equ    $10     ; Port A Bit 4
PA5.   equ    $20     ; Port A Bit 5
PA6.   equ    $40     ; Port A Bit 6
PA7.   equ    $80     ; Port A Bit 7

* Port B Data Register (PORTB -> $01)
PB0    equ    0        ; Port B bit 0
PB1    equ    1        ; Port B bit 1
PB2    equ    2        ; Port B bit 2
PB3    equ    3        ; Port B bit 3
PB4    equ    4        ; Port B bit 4
PB5    equ    5        ; Port B bit 5
PB0.   equ    $01     ; Port B bit 0
PB1.   equ    $02     ; Port B bit 1
PB2.   equ    $04     ; Port B bit 2
PB3.   equ    $08     ; Port B bit 3
PB4.   equ    $10     ; Port B bit 4
PB5.   equ    $20     ; Port B bit 5
```

```

* Port A Data Direction Register (DDRA -> $04)
DDRA0 equ 0 ; Port A Data Direction bit 0
DDRA1 equ 1 ; Port A Data Direction bit 1
DDRA2 equ 2 ; Port A Data Direction bit 2
DDRA3 equ 3 ; Port A Data Direction bit 3
DDRA4 equ 4 ; Port A Data Direction bit 4
DDRA5 equ 5 ; Port A Data Direction bit 5
DDRA6 equ 6 ; Port A Data Direction bit 6
DDRA7 equ 7 ; Port A Data Direction bit 7
DDRA0. equ $01 ; Port A Data Direction bit 0
DDRA1. equ $02 ; Port A Data Direction bit 1
DDRA2. equ $04 ; Port A Data Direction bit 2
DDRA3. equ $08 ; Port A Data Direction bit 3
DDRA4. equ $10 ; Port A Data Direction bit 4
DDRA5. equ $20 ; Port A Data Direction bit 5
DDRA6. equ $40 ; Port A Data Direction bit 6
DDRA7. equ $80 ; Port A Data Direction bit 7

* Port B Data Direction Register (DDRB -> $05)
DDRB0 equ 0 ; Port B Data Direction bit 0
DDRB1 equ 1 ; Port B Data Direction bit 1
DDRB2 equ 2 ; Port B Data Direction bit 2
DDRB3 equ 3 ; Port B Data Direction bit 3
DDRB4 equ 4 ; Port B Data Direction bit 4
DDRB5 equ 5 ; Port B Data Direction bit 5
DDRB0. equ $01 ; Port B Data Direction bit 0
DDRB1. equ $02 ; Port B Data Direction bit 1
DDRB2. equ $04 ; Port B Data Direction bit 2
DDRB3. equ $08 ; Port B Data Direction bit 3
DDRB4. equ $10 ; Port B Data Direction bit 4
DDRB5. equ $20 ; Port B Data Direction bit 5

* Timer Status and Control Register (TSCR -> $08)
RT0 equ 0 ; Real-Time Interrupt Rate Select bit 0
RT1 equ 1 ; Real-Time Interrupt Rate Select bit 1
RTIFR equ 2 ; Real-Time Interrupt Flag Reset
TOFR equ 3 ; Timer Overflow Flag Reset
RTIE equ 4 ; Real-Time Interrupt Enable
TOIE equ 5 ; Timer Overflow Interrupt Enable
RTIF equ 6 ; Real-Time Interrupt Flag
TOF equ 7 ; Timer Overflow Flag
RT0. equ $01 ; Real-Time Interrupt Rate Select bit 0
RT1. equ $02 ; Real-Time Interrupt Rate Select bit 1
RTIFR. equ $04 ; Real-Time Interrupt Flag Reset
TOFR. equ $08 ; Timer Overflow Flag Reset
RTIE. equ $10 ; Real-Time Interrupt Enable
TOIE. equ $20 ; Timer Overflow Interrupt Enable
RTIF. equ $40 ; Real-Time Interrupt Flag
TOF. equ $80 ; Timer Overflow Flag

```

```

* Timer Control Register (TCR -> $12)
*ICIE.      equ      %10000000      ;Input Capture Interrupt Enable
*OCIE.      equ      %01000000      ;Output Compare Interrupt Enable
*TOIE.      equ      %00100000      ;Timer Overflow Interrupt Enable
*IEDG.      equ      %00000010      ;Input Capture Input Edge Select
*OLVL.      equ      %00000001      ;Output Compare Output Level

*ICIE       equ      7              ;Input Capture Interrupt Enable
*OCIE       equ      6              ;Output Compare Interrupt Enable
*TOIE       equ      5              ;Timer Overflow Interrupt Enable
*IEDG       equ      1              ;Input Capture Input Edge Select
*OLVL       equ      0              ;Output Compare Output Level

* Timer Status Register (TSR -> $13)
*ICF.       equ      %10000000      ;Input Capture Flag
*OCF.       equ      %01000000      ;Output Compare Flag
*TOF.       equ      %00100000      ;Timer Overflow Flag

*ICF        equ      7              ;Input Capture Flag
*OCF        equ      6              ;Output Compare Flag
*TOF        equ      5              ;Timer Overflow Flag

* SCI Baud Rate Register (SCBR -> $0D)
SCP1.       equ      %00100000      ;SCI Baud Rate Prescaler Bit 1
SCP0.       equ      %00010000      ;SCI Baud Rate Prescaler Bit 0
SCR2.       equ      %00000100      ;SCI Baud Rate Select Bit 2
SCR1.       equ      %00000010      ;SCI Baud Rate Select Bit 1
SCR0.       equ      %00000001      ;SCI Baud Rate Select Bit 0

SCP1        equ      5              ;SCI Baud Rate Prescaler Bit 1
SCP0        equ      4              ;SCI Baud Rate Prescaler Bit 0
SCR2        equ      2              ;SCI Baud Rate Select Bit 2
SCR1        equ      1              ;SCI Baud Rate Select Bit 1
SCR0        equ      0              ;SCI Baud Rate Select Bit 0

* SCI Control Register 1 (SCCR1 -> $0E)
R8.         equ      %10000000      ;SCI Receive Data Bit 8
T8.         equ      %01000000      ;SCI Transmit Data Bit 8
M.          equ      %00010000      ;SCI Character Word Length
WAKE.       equ      %00001000      ;SCI Wakeup Method Select

R8          equ      7              ;SCI Receive Data Bit 8
T8          equ      6              ;SCI Transmit Data Bit 8
M           equ      4              ;SCI Character Word Length
WAKE        equ      3              ;SCI Wakeup Method Select

```

\* SCI Control Register 2 (SCC2 -> \$0F)

```
TIE.      equ      %10000000    ;SCI Transmit Interrupt Enable
TCIE.     equ      %01000000    ;SCI Transmit Complete Int. Enable
RIE.      equ      %00100000    ;SCI Receive Interrupt Enable
ILIE.     equ      %00010000    ;SCI Idle Line Interrupt Enable
TE.       equ      %00001000    ;SCI Transmitter Enable
RE.       equ      %00000100    ;SCI Receiver Enable
RWU.      equ      %00000010    ;SCI Receiver Wakeup Enable
SBK.      equ      %00000001    ;SCI Send Break
```

```
TIE       equ      7            ;SCI Transmit Interrupt Enable
TCIE      equ      6            ;SCI Transmit Complete Int. Enable
RIE       equ      5            ;SCI Receive Interrupt Enable
ILIE      equ      4            ;SCI Idle Line Interrupt Enable
TE        equ      3            ;SCI Transmitter Enable
RE        equ      2            ;SCI Receiver Enable
RWU       equ      1            ;SCI Receiver Wakeup Enable
SBK       equ      0            ;SCI Send Break
```

\* SCI Status Register (SCSR -> \$10)

```
TDRE.     equ      %10000000    ;Transmit Data Register Empty Flag
TC.        equ      %01000000    ;Transmit Complete Flag
RDRF.     equ      %00100000    ;Receive Data Register Full Flag
IDLE.     equ      %00010000    ;Idle Line Detect Flag
OR.        equ      %00001000    ;Overrun Error Flag
NF.        equ      %00000100    ;Noise Error Flag
FE.        equ      %00000010    ;Framing Error Flag
```

```
TDRE      equ      7            ;SCI Transmit Data Register Empty
TC         equ      6            ;SCI Transmit Complete Flag
RDRF      equ      5            ;SCI Receive Data Register Full
IDLE      equ      4            ;Idle Line Detect Flag
OR         equ      3            ;Overrun Error Flag
NF         equ      2            ;Noise Error Flag
FE         equ      1            ;Framing Error Flag
```

\* Configuration Register (CNFGR -> \$19)

```
STPEN.    equ      %10000000    ;STOP Enable
STREC.    equ      %01000000    ;STOP Recovery
COPEN.    equ      %00100000    ;COP Enable
IRV.      equ      %00010000    ;Internal Read Visibility
LIRV.     equ      %00001000    ;Load Instruction Register Visibility
LVREN.    equ      %00000100    ;Low Voltage Reset Enable
CS1P1.    equ      %00000010    ;Chip Select 1 Parameter bit 1
CS1P0.    equ      %00000001    ;Chip Select 1 Parameter bit 0
```

```

STPEN          equ      7          ;STOP Enable
STREC          equ      6          ;STOP Recovery
COPEN          equ      5          ;COP Enable
IRV            equ      4          ;Internal Read Visibility
LIRV           equ      3          ;Load Instruction Register Visibility
LVREN          equ      2          ;Low Voltage Reset Enable
CS1P1          equ      1          ;Chip Select 1 Parameter bit 1
CS1P0          equ      0          ;Chip Select 1 Parameter bit 0

```

\* COP Register (COPR -> \$FFFF)

```

COPC           equ      0          ; COP Clear
COPC.          equ      $01        ; COP Clear

```

\$PAGE

\*\*\*\*\*

\*

\* Memory Map Equates \*

\*

\*\*\*\*\*


```

RAM            equ      $0040      ; Start of on chip RAM
Vectors        equ      $07F8      ; Start of Reset/Interrupt vectors
Keyboard_INT   equ      $FFF2      ; Keyboard Scan Interrupt Vector
MFT_INT        equ      $FFF4      ; Multifunction Timer Vector
SCI_INT        equ      $FFF6      ; Serial Comm. Interface Vector
TIM_INT        equ      $FFF8      ; 16 bit Timer Interrupt Vector
IRQ_INT        equ      $FFFA      ; External Interrupt Vector
SWI_INT        equ      $FFFC      ; Software Interrupt Vector
RESET          equ      $FFFE      ; Reset Vector

```

```
*****
*
*           RESET and Interrupt Vectors
*
* For any interrupts used, the ORG and FDB statement given below must be
* placed in the routine using the interrupt.
*
*****
*
*           org     Vectors
*
*           org     Timer_INT
*           fdb     Timer_SVR
*
*           org     IRQ_INT
*           fdb     IRQ_SVR
*
*           org     SWI_INT
*           fdb     SWI_SVR
*
*           org     RESET
*           fdb     Start
*
*****
```

## Application Note

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

### How to reach us:

**USA/EUROPE/Locations Not Listed:** Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447 or 602-303-5454

**MFAX:** RMFAX0@email.sps.mot.com – TOUCHTONE 602-244-6609

**INTERNET:** <http://Design-NET.com>

**JAPAN:** Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan.  
03-81-3521-8315

**ASIA/PACIFIC:** Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298



**MOTOROLA**

---