MOTOROLA

TECHNICAL UPDATE

MC68HC705C8A

This technical update is a companion to the latest version of the MC68HC705C8A Technical Data Book (M68HC705C8A/D).

Technical Update contains updates to documented information appearing in other Motorola technical documents as well as new information not covered elsewhere.

We are confident that your Motorola product will satisfy your design needs. This Technical Update and the accompanying manuals and reference documentation are designed to be helpful, informative, and easy to use.

Should your application generate a question or a problem not covered in the current documentation, please call your local Motorola distributor or sales office. Technical experts at these locations are eager to help you make the best use of your Motorola product. As appropriate, these experts will coordinate with their counterparts in the factory to answer your questions or solve your problems. To obtain the latest document, call your local Motorola sales office.

TABLE OF CONTENTS

Modules

TECHNICAL UPDATE

Modules

Central Processor Unit (CPU - HC05CPU)

HC05CPU

Revision History

Date	Revision	Description
5/15/96	0.00	Includes trackers HC05CPU.001, HC705C8.002R2, HC705C8.017, HC705C8.018R2, HC705C8.019 and HC05P4.002.

Correction to SUB in Applications Guide

Reference Documents: M68HC05 Applications Guide MC68HC05AG/AD Rev. 2, page A-62; M6805 HMOS/M146805 CMOS Family User's Manual, page 213

Tracker Number: HC05CPU.001 Revision: 1.00

Replace the C bit description with:

The C bit (carry flag) in the condition code register gets set if the absolute value of the contents of memory is larger than the absolute value of the accumulator, cleared otherwise.

External Interrupt Timing

Reference Documents: MC68HC705C8A/D, page 4-2; MC68HC705C8/D Rev. 1, page 3-5; MC68HC05B6/D, Rev. 3, page 11-11, note 4; MC68HC705C8/D, Rev. 1, page 3-5; MC68HC05C9/D, page 13-7, note 3; MC68HC05C12/D, page 13-9, note 4; MC68HC05D9/D, Rev. 1, page 10-4, note 1; MC68HC05J3/D, page 9-6, note 3; and MC68HC05X16/D, page 12-6, note 4

Tracker Number: HC705C8.002 Revision: 2.00

This time (t_{ILIL}) is obtained by adding 19 instruction cycles to the total number of cycles needed to complete the service routine. The return to interrupt (RTI) is included in the 19 cycles.

I Bit in CCR During Wait Mode

Reference Document: M68HC05 Applications Guide, Rev. 2, page 3-93; MC68HC705C8A/D, page 4-2

Tracker Number: HC705C8.019 Revision: 1.00

The wait mode flow chart does not show that the I bit gets cleared upon entering wait mode. The I bit is cleared when wait is entered. An external IRQ or any of the internal interrupts (timer, SCI, SPI) can release the processor from wait mode.

This error is present in the original applications guide as well as the revision.

Stop Mode Application Example

Reference Document: MC68HC05P4/D, page 3-24

Tracker Number: HC05P4.002 Revision: 1.00

```
*******************************************************************************
*******************************************************************************
*                                                                             *
*                       STOP program example for HC05P4                       *
*                                                                             *
*******************************************************************************
*                                                                             *
* Program Name: STOPP4.ASM
* Date: 12/16/91                                                              *
* Written By: Robert Chretien & David Yoder                                   *
*             Motorola CMCU Applications                                      *
* Assembled Under: P&E Microcomputer Systems, Inc.  IASM05                    *
*                                                                             *
*. This code is written for and tested on the MC68HC705P9.  In order to use *
*  this with other HC05 MCU’s, reset vectors and memory map equates may have *
*  to be changed.  See the Technical Databook for the appropriate part for *
*  this memory map information.                                              *
```

```
 Program Description:                                                   *
*             This program shows how to use the MC68HC705P9 STOP          *
*             instructionIt is meant to be used in a stand a lone mode,   *
*             or with an appropriate evaluation/emulation system.         *
*                                                                         *
*             Upon executing the program, PA0 will toggle. When PA1       *
*             is pulled high, the MCU will enter STOP mode and PA0        *
*             will cease to toggle.                                       *
*             An external reset or an event on IRQ will cause the MCU     *
*             to exit from stop mode.                                     *
*                                                                         *
*                                                                         *
*               ________                                                  *
*               | START |                                                 *
*               ---------                                                 *
*                   |                                                     *
*          /--------|                                                     *
*          |        |                                                     *
*          |    __________                                               *
*          |    | Toggle |                                               *
*          |    |  PA0   |                                               *
*          ^    ----------                                               *
*          |        |                                                     *
*          |       / \                                                   *
*          |     /     \                                                 *
*          | n  /  PA1  \                                                *
*          |---<  High ? >                                               *
*          |    \       /                                                *
*          |     \     /                                                 *
*          |       \ /                                                   *
*          ^        | y                                                  *
*          |    ---------                                                *
*          |    | STOP  |                                                *
*          |    ---------                                                *
*          |        |                                                    *
*          \--------/                                                    *
*                                                                         *
***************************************************************************
*                                                                         *
* Revision History                                                        *
* Rev 1.0: Original program.                                              *
* Rev 2.0: Discaimer added                                                *
*                                                                         *
***************************************************************************
***************************************************************************
*
*
*
****************************************
* MCU Equates                          *
****************************************
PortA   equ   $0000
PortB   equ   $0001
DDRA    equ   $0004
DDRB    equ   $0005

****************************************
* Interrupt vectors                    *
****************************************
        org   $1FF8
TIMER   fdb   TRAP
IRQ     fdb   IRQISR
SWI     fdb   TRAP
RESET   fdb   START
```

```
****************************************
* Start                                *
* Main Loop of code                    *
****************************************
        ORG     $0180           ; Begin code in EPROM
Start   LDA     #$01
        STA     DDRA            ; Set port A0 to output, leave
                                ;  others as inputs
Toggle  LDA     PORTA           ; Toggle port A0. This will toggle
        EOR     #%00000001      ;  while the code is running.
        STA     PORTA           ;  This will stop toggling when STOP
                                ;  mode is entered. When STOP mode
                                ;  is exited with IRQ or RESET, this
                                ;  will resume toggling.


        LDA     PORTA           ; See if PA1 has been pulled high
        AND     #$02            ; If not, branch to TOGGLE to toggle
        BEQ     TOGGLE          ;  PA0 again.
                                ; If so, enter STOP mode.

        STOP                    ; Enter STOP mode.
                                ;  This will:
                                ;    Clear interrupt flag in status
                                ;     register - no need to do CLI for
                                ;     IRQ to exit from STOP
                                ;    Disable the oscilltor - you will
                                ;     see OSC2 stop toggling when STOP
                                ;     mode is successfully entered.

        BRA     TOGGLE          ; Stay in main loop toggling


****************************************
* IRQISR                               *
****************************************
* Service routine for                  *
* external interrupts                  *
*                                      *
* Does nothing, only returns to        *
* main routine.                        *
****************************************
IRQISR:
        NOP
        RTI

****************************************
* TRAP                                 *
****************************************
* Routine for unused interrupts        *
*                                      *
* Traps in a branch to self            *
****************************************
TRAP    BRA     TRAP            ; Trap interrupts


        END
```

Serial Peripheral Interface (SPI - SPI_A)

@SPI_A

Revision History

Date	Revision	Description
5/15/96	0.00	Includes trackers HC05C8.005R2, HC05C8.006R2, and HC705C8.006R2.

SPI Code Snippet (master)

Reference Documents: HC05C4; HC05C4A; HC05C8; HC05C12; HC705C4A; HC705C8; HC705C8A

Tracker Number: HC05C8.005 Revision 2.00

The following code will work on all MCUs that have the SPI_A module. The code was tested on an HC705C8. Some equates and vectors may have to be changed in order to properly work on a specific part.

```
*******************************************************************
*
* Program Name: C8SPIM.ASM
* Revision:     2.0
* Target MCU:   MC68HC05C8, MC68HC705C8
* Date:         8/18/93
* Written by:   David Yoder, Motorola CSIC Applications
* Assembly:     P&E Microcomputer Systems IASM05 3.0m
*
*******************************************************************
* Rev history:
* Rev 2.0       8/18/93         No longer outputs through DACIA of
*                               EVM. Only outputs PA2 if error occurs.
*
* Rev 1.0       8/13/93         original
*******************************************************************
*
* This code shows a basic SPI transfer protocol between one master
* and one slave. A string is continuously transmitted to the slave.
* The companion slave program reverses the case of all alpha
* characters before sending the message back. The message received
* by the master is again case switched and then compared to the
* original message. If a difference is noted, PA2 is driven low. PA2
* idles high.
*
* In order for the handshaking to operate, the MCU's should be
* connected as shown below.
```

```
*
* Master                  Slave
* ------                  -----
* PD2/MISO -------------- PD2/MISO
* PD3/MOSI -------------- PD3/MOSI
* PD4/SCK  -------------- PD4/SCK
* PD5/SS -----\
* PA1 --------/
* PA0 ------------------- PD5/SS
* PA2 ------------------O Error indicator
*
* PA0 controls SS_ (slave select) on the slave. For the mode used
* (cpol=1, cpha=0), the slave SS_ must be brought high between each
* transfer. If it is not, a write collision will result when the
* slave SPDR is written.
*
* PA1 controls SS_ on the master.
*  MC68HCx05C8: The master SS_ must be pulled high while the SPI is
*   enabled in master mode. If it is not, a mode fault will result.
*  MC68HCx05C9: The master PD5/SS pin may be set as output in DDRD bit5.
*   If this is done, master SS_ need not be pulled high. This was not
*   done to insure compatibility with the MC68HCx05C8, which has an
*   input-only Port D.
*
* PA2 pulses low to indicate transmission errors.
*
* PortD DDR
*  MC68HCx05C8: Does not have a data direction register. No need to
*   write to address $07.
*  MC68HCx05C9: Has data direction register. It must be set up
*   appropriately for the SPI to operate.
**********************************************************************

************** Equates *******************
cr      equ     $0d     ;Carriage Return character
lf      equ     $0a     ;Line Feed character

************ MCU Equates *****************
porta   equ     $00
ddra    equ     $04
portd   equ     $03
ddrd    equ     $07

ROM0    equ     $20     ;Start of ROM0
RAM     equ     $50     ;Start of main RAM

************ SPI Equates *****************
spcr    equ     $0a     ;SPI control register
spsr    equ     $0b     ;SPI status register
spdr    equ     $0c     ;SPI data register

************* SPI Bit Equates ************

********** SPCR ***********
spie    equ     7       ;SPI interrupt enable bit
```

```
spe     equ     6       ;SPI enable
mstr    equ     4       ;SPI master enable
cpol    equ     3       ;SPI clock polarity
cpha    equ     2       ;SPI clock phase
spr1    equ     1       ;SPI rate
spr0    equ     0       ;SPI rate

********** SPSR ************
spif    equ     7       ;SPI interrupt flag
wcol    equ     6       ;SPI write collision
modf    equ     4       ;SPI mode fault

********** PortA ***********
sss     equ     0       ;port a0 is tied to ss on slave spi
mss     equ     1       ;port a1 is tied to ss on the master
error   equ     2       ;port a2 pulses low when an SPI error
                        ; is caught

***************DACIA Equates*******************

IER     equ     $20     ;interrupt enable register(write)
ISR     equ     $20     ;interrupt status register(read)
TDR     equ     $23     ;transmit data register(write)
RDR     equ     $23     ;receive data register(read)

*************DACIA bit Equates*******************

DTDRE   equ     6       ;transmit data reg. empty

***************End of Equates******************

************** Variables **********************
        org     RAM     ;start of main RAM
temp    rmb     1       ;temporary variable

************** Reset Vectors *****************
        org     $1ff4           ;vectors
SPI     fdb     trap
SCI     fdb     trap
TIMER   fdb     trap
IRQ     fdb     trap
SWI     fdb     trap
RESET   fdb     start


        org     $0200           ;start of program area

****************Program Beginning**************

start:  bsr     spistrsetup     ;initialise system
        bsr     checksetup
start10 ldx     #msg            ;point to string
start20 bsr     spistr          ;xmit one char of string
        beq     start10         ;start over if end of string
        bsr     casesw          ;reverse case of rec'd char
```

```
        bsr     check           ;same as xmit'd char?
        bra     start20         ;xmit next char

**********************************************
* Subroutine: spistrsetup
* Inputs:
*       none
* Outputs:
*       none
* Alters Regs:
*       A
*       CCR
*
**********************************************
spistrsetup:
        lda     #$18            ;sck=1,mosi=1 this does nothing on C8
        sta     ddrd            ;but MUST be done on C9

        lda     #{1<sss + 1<mss}
                                ;left shift 1's into these bit positions
                                ;port a0 controls ss on the slave spi
                                ;port a1 controls ss on the master
        sta     ddra

        bset    sss,porta       ;deselect slave
                                ; the slave ss must go low during the
                                ; transfer and high between transfers
                                ; for the mode cpha=1,cpol=0. If the
                                ; spdr of the slave is written while
                                ; ss of the slave is low, a write
                                ; collision will occur.
        bset    mss,porta       ;deselect master
                                ; the master ss must be held high during
                                ; all time that the master spe and mstr
                                ; bits are set. A mode fault will result
                                ; if it is not held high during this
                                ; time.

        lda     #{1<spe + 1<mstr + 1<cpol + 1<spr0}
                                ;left shift 1's into these bit positions
                                ;set the master up as follows
                                ; do not enable spi interrupts
                                ; enable spi
                                ; enable master mode
                                ; cpol=1
                                ; cpha=0
                                ; spr=01 : sck=eck/4
        sta     spcr

        rts                     ;return from subroutine

**********************************************
* Subroutine: checksetup
* Inputs:
*       none
```

```
* Outputs:
*       none
* Alters Regs:
*       none
*
**********************************************
checksetup:
        bset    error,ddra      ;set error bit as output
        bset    error,porta     ;put error flag in idle state
                                ; pa2 will toggle low if an error is
                                ; detected in the SPI system

        rts                     ;return from subroutine
********************SPI Data Transfer***********
* Subroutine: spistr
* Inputs:
*       X: address of string to xmit
* Outputs:
*       A:      character received by SPI
*       X:      address of next character to xmit
*       CCR:    Z bit set if end of string is reached
* Depends upon:
*       spistrsetup
* Alters Regs:
*       A
*       X
*       CCR
*       Variable TEMP
* Description:
* Transmits one character of a string out the SPI
* system. Returns with Z bit set when the "$"
* is reached. Returns with Z bit clear if "$"
* is not reached.
************************************************

spistr:
        lda     ,x              ;get message data
        cmp     #"$"            ;is it the end of the message?
        beq     spistr10        ;done with string
                                ;return with Z bit set
        bclr    sss,porta       ;select slave
        sta     spdr            ;send character
        brclr   spif,spsr,*     ;check spif, wait until set
        bset    sss,porta       ;deselect slave
                                ; slave must be deselected so that
                                ; it can write to it's own spdr
                                ; and not cause a write collision
        incx                    ;set up to send next byte

        lda     spdr            ;get the recieved character

        clr     temp            ;we are not done with the
        com     temp            ; string, so insure that Z
                                ; bit is clear for return
```

```
spistr10:
        rts                     ;return to calling routine

*******************************************************
* Subroutine: casesw
* Intputs:
*       ASCII character in acc
* Outputs:
*       ASCII character in acc
* Alters Regs:
*       A
*       CCR
*
* Routine changes upper case to lower case and
* lower case to upper case. Leaves non-alpha
* characters unchanged.
********************************************************

casesw  cmp     #$41            ;below alphas?
        bmi     casesw20
        cmp     #$5b            ;above caps?
        bpl     casesw10
        add     #$20            ;must be cap, change to low
        bra     casesw20
casesw10:
        cmp     #$61            ;between alphas?
        bmi     casesw20
        cmp     #$7b            ;above alphas?
        bpl     casesw20
        sub     #$20            ;must be lowercase, change to cap

casesw20:
        rts                     ;return

*******************************************************
* Subroutine: check
* Intputs:
*       A: value to check
*       X: pointer to next character to xmit
*          offset from received char by -2
* Outputs:
*       A: unaffected
*       X: unaffected
* Depends upon:
*       checksetup
* Alters Regs:
*       CCR
*
* Routine compares the value in accumulator to value
* pointed to by (X-2).
*
* If the value do not match, PA0 is pulsed low
*
* If X=msg+1, the routine returns immediately.
*
```

```
* This is usefull for comparing
* received to transmitted data with the SPI.
*******************************************************

check:
        cpx     #msg+1          ;was this the 1st xfer?
        beq     check20         ;if so, don't bother
        decx                    ;X=X-2
        decx
        cmp     ,x              ;rec char = xmit char?
        beq     check10
        bclr    error,porta     ;if not, pulse error
        bset    error,porta
check10:
        incx                    ;X=X+2
        incx
check20:
        rts                     ;done

****************Trap************************

trap    bra     trap            ;trap for unused vectors

**************Message data*******************
        org     ROM0            ;store message in Page0 ROM
msg:    db      "The Quick Brown Fox jumped over the Lazy Dog",cr,lf,"$"
```

SPI Code Snippet (slave)

Reference Documents: HC05C4; HC05C4A; HC05C8; HC05C12; HC705C4A; HC705C8; HC705C8A

Tracker Number: HC05C8.006 Revision 2.00

The following code will work on all MCUs that have the SPI_A module. The code was tested on an HC705C8. Some equates and vectors may have to be changed in order to properly work on a specific part.

```
*******************************************************************
*
* Program Name: C8SPIS.ASM
* Revision:     1.1
* Target MCU:   MC68HC05C8, MC68HC705C8
* Date:         8/18/93
* Written by:   David Yoder, Motorola CSIC Applications
* Assembly:     P&E Microcomputer Systems IASM05 3.0m
*
*******************************************************************
* Rev History:
* 1.1           8/18/93         changed label names for consistancy
* 1.0           8/12/93         original for MC68HCx05C9 memory map
```

```
*********************************************************************
*
* This code shows a basic SPI transfer protocol between one master
* and one slave. This slave module receives characters, changes
* the case of all alpha characters, and transmits the character
* back. Non-alpha characters are transmitted unchanged.
*
* In order for the handshaking to operate, the master should use the
* code snippet (C9SPIM.ASM), and the MCU's should be connected as
* shown below.
*
* Master                  Slave
* ------                  -----
* PD2/MISO -------------- PD2/MISO
* PD3/MOSI -------------- PD3/MOSI
* PD4/SCK  -------------- PD4/SCK
* PD5/SS -----\
* PA1 --------/
* PA0 ------------------- PD5/SS
* PA2 ------------------O Error indicator
*
* PA0 in the master code snippet controls SS_ (slave select) on the
* slave. For the mode used (cpol=1, cpha=0), the slave SS_ must be
* brought high between each transfer. If it is not, a write collision
* will result when the slave SPDR is written.
*
* PA1 controls SS_ on the master. The master code snippet is written
* such that the master controls it's own slave select line.
*  MC68HCx05C8: The master SS_ must be pulled high while the SPI is
*   enabled in master mode. If it is not, a mode fault will result.
*  MC68HCx05C9: The master PD5/SS pin may be set as output in DDRD bit5.
*   If this is done, master SS_ need not be pulled high. This was not
*   done to insure compatibility with the MC68HCx05C8, which has an
*   input-only Port D.
*
* PA2 of the master code snippet pulses low to indicate transmission
* errors.
*
* PortD DDR
*  MC68HCx05C8: Does not have a data direction register. No need to
*   write to address $07.
*  MC68HCx05C9: Has data direction register. It must be set up
*   appropriately for the SPI to operate.
*********************************************************************

************** Equates *******************
cr      equ     $0d             ;carriage return character
lf      equ     $0a             ;line feed character
dc1     equ     $11
************ SPI Equates *****************

portd   equ     $03             ;port d
ddrd    equ     $07             ;data direction register for port d
spcr    equ     $0a             ;SPI control register
spsr    equ     $0b             ;SPI status register
```

```
spdr    equ     $0c             ;SPI data register

************* SPI Bit Equates ************

******* SPCR ***********
spie    equ     7               ;SPI interrupt enable bit
spe     equ     6               ;SPI enable bit
mstr    equ     4               ;SPI master mode bit
cpol    equ     3               ;SPI clock polarity bit
cpha    equ     2               ;SPI clock phase bit
spr1    equ     1               ;SPI rate bit 1
spr0    equ     0               ;SPI rate bit 0

******* SPSR ************
spif    equ     7               ;SPI interrupt flag bit
wcol    equ     6               ;SPI write collision bit
modf    equ     4               ;SPI mode fault bit

***************End of Equates******************

        org     $1ff4              ;reset vectors

*************** Vectors ************************

SPI     fdb     echosw
SCI     fdb     trap
TIMER   fdb     trap
IRQ     fdb     trap
SWI     fdb     trap
reset   fdb     start

        org     $0200           ;start of program area

****************Program Beginning**************

start:
        bsr     setup
        cli                     ;enable system wide interrupts

start10:
        nop                     ;wait for interrupts
        bra     start10

*****************Init SPI*********************
* Subroutine: setup
* Inputs:
*       none
* Outputs:
*       none
*
* Initializes SPI system
**********************************************
setup:  lda     #$04            ;set up PD2/MOSI as output
        sta     ddrd            ;others as input
                                ;MUST be done on C9,
```

```
                                ;has no effect on C8

        lda     #{1<spie + 1<spe + 1<cpol + 1<spr0}
                                ;shift 1's into appropriate
                                ; bit postions
        sta     spcr            ;setup SPI as follows:
                                ; enable SPI interrupts
                                ; enable SPI system
                                ; do not enable master mode
                                ; cpol=1 : in this mode, ss must
                                ; cpha=0 : go high between xfers
                                ; spr=01 : sck=eck/4
        rts


********************SPI Data Transfer ISR*******
* ISR: echosw
* Depends upon:
*       setup
*       casesw
*
* Slave SPI ISR.
* Receives character from SPI system. Assumes the character
* to be ASCII. Switches the case of all alpha characters.
* Does not affect non-alpha characters. Transmits the
* resulting character back to master.
*****************************************************
echosw:
        brclr   spif,spsr,*     ;make sure transmission is complete

        lda     spdr            ;get data received from SPI

        bsr     casesw          ;reverse the case of alphas

echosw10:
        sta     spdr            ;send data
                                ; with cpha=1, ss must go low
                                ; before this write is made.
                                ; If not, a write collision
                                ; will occur. In this example,
                                ; the master controls the slave
                                ; ss line.

        brset   wcol,spsr,echosw10
                                ;if a write collision occurred,
                                ;try again

        rti                     ;return to main loop

*****************************************************
* Subroutine: casesw
* Intputs:
*       ASCII character in acc
* Outputs:
*       ASCII character in acc
```

```
* Alters Regs:
*       A
*       CCR
*
* Routine changes upper case to lower case and
* lower case to upper case. Leaves non-alpha
* characters unchanged.
*******************************************************

casesw  cmp     #$41            ;below alphas?
        bmi     casesw20
        cmp     #$5b            ;above caps?
        bpl     casesw10
        add     #$20            ;must be cap, change to low
        bra     casesw20
casesw10:
        cmp     #$61            ;between alphas?
        bmi     casesw20
        cmp     #$7b            ;above alphas?
        bpl     casesw20
        sub     #$20            ;must be lowercase, change to cap

casesw20:
        rts                     ;return

******************Trap************************
trap    bra     *               ;trap for unused vectors
```

SPI Test Program

Reference Documents: HC05C4; HC05C4A; HC05C8; HC05C12; HC705C4A; HC705C8; HC705C8A

Tracker Number: HC705C8.004 Revision 2.00

The following code will work on all MCUs that have the SPI_A module. The code was tested on an HC705C8. Some equates and vectors may have to be changed in order to properly work on a specific part.

```
*****************************************************************
*
* Program Name: 7C8_SPI.ASM ( SPI Test on the 705C8 )
* Revision: 1.00
* Date: June 7, 1993
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
```

```
*       ********************************
*       *       Revision History        *
*       ********************************
*
*       Rev     1.00    06/07/93        M.R. Glenewinkel
*                       Initial Release
*
*****************************************************************
*
* Program Description:
*
*       Use the HC705C8 resident MCU on the HC05EVM to
*         run this test.
*       Jumper pin #34 on header J19 on the EVM to 5v through
*         A 10kOhm resistor. This ties the SS pin of the SPI
*         high insuring against the possibility of a mode
*         fault error.
*       Download the program.
*       Make sure the PC is at $800.
*       Type GO.
*       Look at pin #32 of header J19. This is the MOSI pin
*         of the SPI. You should see '$55' come out of this
*         pin. The MOSI pin's steady state level is a logic '1'.
*         The bitstream's width is 8usecs. Each bit using
*         1usec of time. The program executes in an
*         infinite loop.
*       ABORT the program to stop operation.
*
*****************************************************************

***     Equates for 705C8
SPCR    EQU     $0A                     ;spi ctrl reg
SPSR    EQU     $0B                     ;spi status reg
SPDR    EQU     $0C                     ;spi data reg


***     Start of program                ***

        ORG     $0800                   ;start of user eprom

START   LDA     #$50
        STA     SPCR                    ;spi enabled, mstr
                                        ;  cpha=cpol=spr1=spr0=0

AGAIN   LDA     #$55
        STA     SPDR                    ;send $55 out on spi

LOOP    LDA     SPSR                    ;load spi status reg
        AND     #$80                    ;check if SPIF bit is set
        BEQ     LOOP                    ;if not, go back
                                        ;  and check again

        BRA     AGAIN
```

Serial Communications Interface (SCI - SCICSER_A)

@SCICSER_A

Revision History

Date	Revision	Description
5/15/96	0.00	Includes tracker HC705C8.005R3.

SCI Test Program

Reference Document: HC05C4, HC05C4A, HC05C8, HC05C12, HC705C4A, HC705C8, HC705C8A

Tracker Number: HC705C8.005 **Revision: 3.00**

An example of running the SCI on an HC705C8 is given below:

The following code will work on all MCUs that have the SCICSER_A module. The code was tested on an HC705C8. Some equates and vectors may have to be changed in order to properly work on other MCU's.

```
*****************************************************************
*
* Program Name: 7C8_SCI.ASM ( SCI Test on the 705C8 )
* Revision: 2.00
* Date: July 5, 1994
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
*       *********************************
*       *       Revision History        *
*       *********************************
*
*       Rev     1.00    06/07/93        M.R. Glenewinkel
*                       Initial Release
*
*       Rev     2.00    07/05/94        M.R. Glenewinkel
*                       Added SCSR dummy read
*
*****************************************************************
*
* Program Description:
*
*       Use the HC705C8 resident MCU on the HC05EVM to
*         run this test.
*       Download the program.
```

```
*       Make sure the PC is at $800.
*       Type GO.
*       Look at pin #30 of header J19. This is the TDO pin
*         of the SCI. You should see '$01' come out of this
*         pin according to Figure 5-1 of the MC68HC705C8
*         Technical Data Manual. Document #MC68HC705C8/D Rev 1.
*         The TDO pin's steady state level is a logic '1'.
*       The total period of the SCI's bit stream for one
*         transmission of an 8 bit character at 9600 baud
*         is 1.042 msecs. Each bit's period is 104.2 usecs.
*       Please see the graph below.
*
*         _     ___                             ___
*          |   |   |                           |   |
*          |___|   |___.___.___.___.___.___.___|   |_
*            S   0   1   2   3   4   5   6   7   S
*            T                                   T
*            A                                   O
*            R                                   P
*            T
*
*
*       ABORT the program to stop operation.
*
*****************************************************************

***     Equates for 705C8
BAUD    EQU     $0D                     ;baud rate reg
SCCR1   EQU     $0E                     ;sci cntl reg 1
SCCR2   EQU     $0F                     ;sci cntl reg 2
SCSR    EQU     $10                     ;sci status reg
SCDAT   EQU     $11                     ;sci data reg


***     Start of program                ***

        ORG     $0800                   ;start of user eprom

        lda     #$00
        sta     SCCR1                   ;8 bit char word len
        lda     #$08
        sta     SCCR2                   ;enable transmitter
        lda     #$30
        sta     BAUD                    ;9600 baud w/4MegHz xtal
        lda     SCSR                    ;dummy read to initialize
                                        ; 1st transmission

OUTPUT  lda     #$01
        sta     SCDAT                   ;load up SCI data reg

LOOP    lda     SCSR                    ;check if transmit data
        and     #$80                    ; reg is empty
        beq     LOOP                    ; if so, go on

        bra     OUTPUT                  ;infinite transmission
```

Timer Interface Module (TIM - TIM1IC1OC_A)

@TIM1IC1OC_A

Revision History

Date	Revision	Description
5/15/96	0.00	Includes trackers HC05C4.002R2, HC05C4.003R3, and HC705P9.005R3.

Input Capture/ Output Compare Code Snippet

Reference Documents: HC05C4; HC05C4A; HC05C8; HC05C9; HC05C9A; HC05C12; HC05D9; HC05D24; HC05D32; HC05J3; HC05P1; HC05P1A; HC05P4; HC05P5; HC05P6; HC05P7; HC05P9; HC05P10; HC05P15; HC05P18; HC705C4A; HC705C8; HC705C8A; HC705C9; HC705D9; HC705D32; HC705J3; HC705P6; HC705P9

Tracker Number: HC05C4.002 **Revision 2.0**

```
Previous Rev: 1.00
Changes: Added memory map disclaimer.

*****************************************************************
*
*       Program Name: ICOCC4.ASM
*       Revision: 1.0
*       Date: 9/6/93
*
*       Written By: Mark Johnson
*                   Motorola CSIC Applications
*
*       Assembled Under: P&E Microcomputer Systems
*                        IASM05 Version 3.02m
*
*                 ********************************
*                 *       Revision History       *
*                 ********************************
*
*       Revision 1.00   9/1/93 Original Release
*
*****************************************************************
*
*       Program Description:
*
*       This was written for the timer module TIM1IC1OC_A and tested
*         on the HC05C4. In order to use this with other HC05 MCU's,
*         reset vectors and memory map equates may have to be changed.
```

```
*         See the Technical Databook for the appropriate part for this
*         memory map information.
*
*       This simple program was written to demonstrate the input
*       capture and output compare functions of the MC68HC(8)05C4
*       timer. The routine generates a level transition on port A
*       which is fed into the input capture pin (TCAP). When
*       the input capture occurs an offset of 50us is added to
*       value in the input capture registers and stored in the
*       output compare registers. The output compare generates
*       a level transition on the TCMP pin and then the entire
*       process is repeated.
*
*
*       The program was run on the M68HC05EVM using the
*       following setup conditions:
*
*       1) HC705C8 Resident Processor
*       2) Fop = 2MHz
*       3) Pin 11 (PA0) on target header J19 jumpered to pin
*          37 (TCAP).
*       4) The user should see a level transition on the
*          TCMP pin approximately* 50us after the level
*          transition on port A.
*
*   *NOTE: The level transition on the TCMP pin will occur at
*          50us + 1 count of the free-running counter = 52us.
*          This is the result of an internal synchronization
*          delay which occurs during an input capture.
*          ( 1 count = 4 internal bus cycles)
*****************************************************************
*
*       Register Equates
*
porta           equ     $00             ;port A data register
ddra            equ     $04             ;port A data dir. reg.
tcr             equ     $12             ;timer control register
tsr             equ     $13             ;timer status register
inpcaph         equ     $14             ;input capture (MSB)
inpcapl         equ     $15             ;input capture (LSB)
outcomph        equ     $16             ;output compare (MSB)
outcompl        equ     $17             ;output compare (LSB)
*
*      RAM Variables
*
                org     $50             ;RAM address space
templ           rmb     1               ;storage for O/C low byte
*
*      Beginning of main routine
*
                org     $200            ;EPROM/ROM address space
start           lda     #$ff
                sta     ddra            ;all port A pins are outputs
                clra
                sta     porta           ;output a low on port A
                lda     #3
                sta     tcr             ;IEDG = positive edge
                                        ;OLVL = high output
loop            lda     tsr             ;read timer status register
                lda     outcompl        ;clear OCF
```

```
                com     porta           ;toggle port A
                lda     #!25            ;I/C low byte offset
                add     inpcapl         ;add I/C low byte value
                sta     templ           ;save new value in temp storage
                lda     inpcaph         ;get high byte of I/C reg.
                adc     #0              ;add carry from last addition
                sta     outcomph        ;store value to O/C high byte
                lda     templ           ;get low byte offset
                sta     outcompl        ;store value in O/C low byte
                lda     inpcapl         ;enable input captures
                brclr   6,tsr,*         ;wait for output compare
                lda     tcr             ;get Timer Control Register
                eor     #3              ;toggle IEDG and OLVL
                sta     tcr             ;store new IEDG and OLVL values
                bra     loop            ;repeat process indefinitely
*
*       Reset vector setup
*
                org     $1ffe
                fdb     start
```

Interrupt Driven Output Compare Code

Reference Document: MC68HC05C4/D, P.4-7; MC68HC705C4A/D; MC68HC705C4A/D

Tracker Number: HC05C4.003 Revision 2.00

Previous Rev: 1.00
Changes: Added memory map disclaimer.

The following code listing shows the procedure of using the output compare function driven by an interrupt to produce a square wave. The code was tested with an HC705C8 on the HC05EVM board. The code will work on an HC05C4.

```
*****************************************************************
*
* Program Name: 7C8_OCI.ASM ( Square wave generation on OC )
* Revision: 1.00
* Date: September 29, 1993
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
*       ********************************
*       *       Revision History       *
*       ********************************
*
*       Rev     1.00    09/29/93        M.R. Glenewinkel
*                       Initial Release
```

```
*
*****************************************************************
*
* Program Description:
*
*       This was written for the timer module TIM1IC1OC_A and tested
*         on the HC05C4. In order to use this with other HC05 MCU's,
*         reset vectors and memory map equates may have to be changed.
*         See the Technical Databook for the appropriate part for this
*         memory map information.
*
*       This program uses the Output Compare function of the
*         timer to generate a square wave. The output compare
*         interrupt is utilized to take care of adding the
*         appropriate value to the 16 bit output compare
*         register to create the square wave. With some
*         modification, this routine can perform pulse width
*         modulation.
*
*       Use the HC705C8 resident MCU on the HC05EVM to
*         run this test.
*       Download the program.
*       Make sure the PC is at $1000. Type GO.
*       OR, hit USER RESET on the EVM.
*       Look at pin #35 of header J19. This is the Timer
*         Compare Output pin (TCMP) of the timer. You should
*         see a 3.906kHz square wave on this pin with a
*         256 usec period.
*       Press ABORT on the EVM to halt program execution.
*
*****************************************************************

***     Equates for 705C8
TCR     equ     $12                     ;timer ctrl reg
TSR     equ     $13                     ;timer status reg
OCH     equ     $16                     ;output compare high reg
OCL     equ     $17                     ;output compare low reg
TCH     equ     $18                     ;timer counter high reg
TCL     equ     $19                     ;timer counter low reg
TEMP    equ     $50                     ;temp loc for OCL

***     Start of program                ***

        org     $1000                   ;start of user code

START   lda     #$41                    ;output compare interrupt
                                        ; enabled, output level 0
        sta     TCR                     ;store to timer ctrl reg
        cli                             ;clear the I bit in CCR

DUMLOOP bra     DUMLOOP                 ;dummy loop waiting for
                                        ; timer interrupt


***     Interrupt Service Routine       ***
OCISR   lda     TSR                     ;read timer status
                                        ; to clear flag

*       Flip the OLVL bit in the TCR reg
```

```
        lda     TCR                     ;load ACCA w/ TCR
        eor     #$01                    ;flip bit 0 of ACCA
        sta     TCR                     ;store ACCA to TCR

*       Add 64 counts to timer counter reg
*       With a 2 MHz internal bus clock, the timer count
*         period is 2 usec. 64 counts of the timer counter
*         will produce a square wave half cycle of 128 usecs.
        lda     #$40                    ;load #$40 into acca
        add     OCL                     ;add OCL to ACCA
        sta     TEMP                    ;store res to temp loc
        lda     #$00                    ;add $00 to out comp hi
        adc     OCH                     ; with carry
        sta     OCH                     ;store res to out comp hi
        lda     TEMP                    ;store temp to out
        sta     OCL                     ; comp low

        rti                             ;return from interrupt

***     Set up vectors
        org     $1FF8                   ;define timer
        dw      OCISR                   ; interrupt vector

        org     $1FFE                   ;define reset vector
        dw      START
```

Input Capture Test

Reference Document: HC705P9; MC68HC705C4A/D; MC68HC705C8A/D

Tracker Number: HC705P9.005 **Revision 3.00**

```
Previous Rev of DCO: 2.00
Changes from previous DCO: Expanded text for memory map note.

Previous Rev of DCO: 1.00
Changes from previous DCO: Added memory map note to code.


This was written for the timer module TIM1IC1OC_A and tested on the HC705P9. In
order to use this with other HC05 MCU's, reset vectors and memory map equates may
have to be changed.
See the Technical Databook for the appropriate part for this memory map informa-
tion.

*****************************************************************
*
* Program Name: P9_INCAP.ASM ( Input Capture Test for the P9EVS )
* Revision: 1.00
* Date: June 7, 1993
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
```

```
*
*       ********************************
*       *       Revision History       *
*       ********************************
*
*       Rev     1.00    06/07/93        M.R. Glenewinkel
*                       Initial Release
*
*****************************************************************
*
* Program Description:
*
*       This was written for the timer module TIM1IC1OC_A and tested
*         on the HC705P9. In order to use this with other HC05 MCU's,
*         reset vectors and memory map equates may have to be changed.
*         See the Technical Databook for the appropriate part for this
*         memory map information.
*
*       Tests the Input capture pin.
*       Use the HC705P9 resident MCU on the HC05P9EVS to
*         run this test.
*       Jumper pins PA0 and PD7/TCAP on Target Header P4.
*       We will use Port A, bit 0 to toggle the TCAP pin.
*       Download the program.
*       Make sure the PC is at $100.
*       Type GO.
*       ABORT the program and look at locations $80-$83.
*         After the first Input Capture, the Input Capture
*         Registers High and Low are loaded into RAM
*         location $80 and $81, respectively. After the
*         second Input Capture, the Input Capture Registers
*         High and Low are loaded into RAM location $82
*         and $83, respectively.
*       If you trace this program, the Input capture
*         flag will look like its not being set when you
*         view with the emulator software. Remember, the
*         flag gets cleared when a read of ICL and TSR occurs.
*         The emulator software does this automatically when
*         reading those locations to display in the
*         emulator window.
*
*****************************************************************

***     Equates
PORTA   EQU     $00
PORTB   EQU     $01
PORTC   EQU     $02
DDRA    EQU     $04
DDRB    EQU     $05
DDRC    EQU     $06
DDRD    EQU     $07
TCR     EQU     $12
TSR     EQU     $13
ICRH    EQU     $14
ICRL    EQU     $15

TEMP1   EQU     $0080
TEMP2   EQU     $0081
TEMP3   EQU     $0082
TEMP4   EQU     $0083
```

```
***     Start of code

        ORG     $0100                   ;start of program

START   LDA     #$FF
        STA     PORTA                   ;PortA is $FF
        LDA     #$00
        STA     DDRD                    ;PortD is input
        LDA     #$FF
        STA     DDRA                    ;PortA is output
        STA     DDRC

        LDA     #$00
        STA     TCR                     ;set InCap to fall edge
        LDA     TSR                     ;look at tsr
        LDA     ICRL                    ;look at input reg low
                                        ;this clears any flags

        LDA     #$00                    ;falling edge created
        STA     PORTA                   ; on PortD/TCAP

LOOP    LDA     TSR                     ;wait in loop for flag
        AND     #$80                    ;  to be set
        BEQ     LOOP

        LDA     ICRH                    ;write counter values
        STA     TEMP1                   ;in memory
        LDA     ICRL
        STA     TEMP2

        LDA     #$02                    ;set InCap to rising edge
        STA     TCR
        LDA     #$FF                    ;rising edge created
        STA     PORTA                   ; on PortD/TCAP

LOOP2   LDA     TSR                     ;wait in loop for flag
        AND     #$80                    ;  to be set
        BEQ     LOOP2

        LDA     ICRH                    ;write counter values
        STA     TEMP3                   ;in memory
        LDA     ICRL
        STA     TEMP4

LOOP3   NOP
        BRA     LOOP3
```

Computer Operating Properly (COP - COP0COP_A)

@COP0COP_A

Revision History

Date	Revision	Description
29	0.00	Includes tracker HC705P6.012

COP Timeout Period

Reference Document: HC05C4AGRS/D Rev 1.2; HC05C5GRS/D Rev 1.2; HC705C5GRS/D Rev 1.3, page 49; MC68HC705C8AD/D Rev 4.0, page 14 (705C4A); MC68HC705C8AD/D Rev 4.0, page 31 (705C8A); MC68HC705C8AD/D Rev. 4.0, page 51 (HSC705C8A); MC68HC05C12/D; HC05P1AGRS/D Rev. 1.3; MC68HC05P4/D, page 4-2; HC05P5GRS/D Rev 1.3; MC68HC05P7/D, page 4-2; HC05P15GRS/D Rev. 0.0, page 33; HC05P18GRS/D Rev. 0.5, page 12

Tracker Number: HC705P6.012 Revision: 1.00

The timeout period for the COP0COP_A computer operating properly watchdog timer is a direct function of the crystal frequency. The equation is:

$$\text{Timeout Period} = \frac{262{,}144}{\text{Fxtal}}$$

For example, the timeout period for a 4-MHz crystal is 65.536 ms.

Computer Operating Properly (COP - COP55AACOPCR_A)

@COP55AACOPCR_A

Revision History

Date	Revision	Description
5/15/96	0.00	Includes tracker HC705C8.020.

COP Register Correction

Reference Document: HC705C8A; HC705C8, Page 3-2

Tracker Number: HC705C8.020 Revision 1.00

The COP Reset Register detailed in section 3.1.3.1 of the MC68HC705C8/D Rev 1 Technical Data Manual has the wrong address documented. The databook currently states the address at $0010.

The CORRECT address is $001D.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No.2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.