MOTOROLA

TECHNICAL UPDATE

MC68HC705P6

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

Computer Operating Properly (COP)....................3

- **COP Timeout Period.................... 3**

CPU....................4

- **Correction to SUB in Applications Guide.................... 4**
- **External Interrupt Timing.................... 4**
- **I Bit in CCR During Stop Mode.................... 5**
- **BSET and BCLR are Read-Modify-Write Instructions.................... 5**
- **I Bit in CCR During Wait Mode.................... 6**

Timer....................7

- **Input Capture/Output Compare Code Snippet.................... 7**
- **Interrupt Driven Output Compare Code.................... 9**
- **Input Capture Test.................... 11**

Analog to Digital....................13

- **A/D Example Code.................... 13**
- **A/D Converter Source Impedance.................... 15**
- **A/D Accuracy Specification.................... 16**

Part Specific

Bootloader Code....................17

Mask Set Erratas....................24

- **705P6MSE1(R1).................... 24**
- **705P6MSE2(R2).................... 26**
- **705P6MSE3.................... 28**

TECHNICAL UPDATE

Modules

Computer Operating Properly (COP)

COP0COP_A

Revision History

Date	Revision	Description
7/7/95	1.00	Includes tracker HC705P6.012

COP Timeout Period

Reference Documents: HC05C4AGRS/D Rev. 1.1, page 21; HC05C5GRS/D Rev. 1.2, page 24; 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, page 5-2; HC05P1AGRS/D Rev. 1.02, page 17; MC68HC05P4/D, page 4-2; HC05P5GRS/D Rev. 1.3, page 35; 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.

CPU

HC05CPU

Revision History

Date	Revision	Description
5/3/95	1.00	Includes trackers HC05CPU.001, HC705C8.002R2, HC705C8.017, HC705C8.018R2, and HC705C8019.

Correction to SUB in Applications Guide

Reference Documents: M68HC05 Applications Guide MC68HC05AG/AD, page A-62; M68HC05 Applications Guide MC68HC05AG/AD Rev. 1, page A-62

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: 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 (TILIL) 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 Stop Mode

Reference Document: M68HC05 Applications Guide, page 3-93

Tracker Number: HC705C8.017 Revision: 1.00

The stop mode flow chart shows that the I bit is set when stop mode is entered. However, this is not true. The I bit actually is cleared when stop mode is entered so that an external IRQ may release the processor from stop mode.

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

BSET and BCLR are Read-Modify-Write Instructions

Reference Documents: MC68HC705C8/D Rev. 1, page 7-6; MC68HC05J1/D Rev. 1, page 5-7; MC68HC05J3/D, page 8-4; MC68HC705J2/D, page 4-16; HC05J3/705J3 Technical Data - MC68HC05J3/D, page 8-6; MC68HC05K1/D, page 10-10; MC68HC705K1/D, page 11-10

Tracker Number: HC705C8.018 Revision: 2.00

In many data books, the read-modify-write instruction table located in the instruction set and addressing mode section does not list the BSET and BCLR instructions. These data books list BSET and BCLR as bit-manipulation instructions only.

While this is correct, it is not complete. These operations use a read-modify-write method to accomplish their task and, therefore, should be included in the table of read-modify-write instructions.

NOTE: These instructions do not use the same addressing modes as the other read-modify-write instructions. Only direct addressing is valid for BSET and BCLR.

Because BSET and BCLR are read-modify-write instructions, they may not be used with write-only registers. These registers will read back undefined data. Therefore, a read-modify-write operation will read undefined data, modify it as appropriate, and then write it back to the register. Because the original data is undefined, the data written back will be undefined also.

I Bit in CCR During Wait Mode

Reference Document: M68HC05 Applications Guide, page 3-93

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.

Timer

TIM1IC1OC_A

Revision History

Date	Revision	Description
7/7/95	1.00	Includes trackers HC05C4.002R2, HC05C4.003R2, and HC705P9.005R2.

Input Capture/Output Compare Code Snippet

Reference Document: Not applicable

Tracker Number: HC05C4.002 Revision: 2.00

```
************************************************************
*
*       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 (ADI-991-R2), page 4-7

Tracker Number: HC05C4.003   Revision: 2.00

The following code uses the output compare function driven by an interrupt to produce a square wave. The code was tested with an HC705C8 on the HC05EVM board and 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: Not applicable

Tracker Number: HC705P9.005   Revision: 2.00

This program tests the input capture module TIM1IC1OC_A on the HC705P9 on the HC05P9EVS.

```
*****************************************************************
*
* 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
```

Analog to Digital

ATD4X8NVRL_A

Revision History

Date	Revision	Description
7/7/95	1.00	Includes trackers HC05P6.005, HC705P9.002, and HC705P9.010.

A/D Example Code

Reference Document: MC68HC05P6/D, page 10-1

Tracker Number: HC05P6.005 Revision: 1.00

The following code shows a simple routine that will read AN0 of the A/D converter. Port C bit 6 is the A/D converter input AN0. The code was tested on an M68HC05P9EVS evaluation system. It is assumed the reader is familiar with the IASM05 assembler and the EVM05 debugger from P&E Microcomputer Systems. These P&E software programs are used to assemble, download, and run the code. If the reader is unfamiliar with emulating the HC05P6 on the P9EVS, consult the P9EVS user's manual, M68HC05P9EVS/D1, for that system.

This code was tested on the HC05P6 but can be used on other parts that have the ATD4X8NVRL_A module. Due to differences in memory maps, equates and orgs for RAM, EPROM, and RESET vectors may need to be changed. Consult the applicable MCU technical data book for specific part information.

Assemble the following lines of code.

```
****    Equates
ADSCR   EQU     $1E                     ;a/d status and ctrl reg
ADDR    EQU     $1D                     ;a/d data reg


        ORG     $100                    ;start of program

START   LDA     #$20
        STA     ADSCR                   ;turn on the a/d converter

        LDA     #33T                    ;execute loop for 100 usecs
WAIT    DECA                            ; so a/d can warm up
        BNE     WAIT

        LDA     #$20
        STA     ADSCR                   ;start a conversion
CHCK    LDA     ADSCR                   ;check if conversion
        AND     #$80                    ; complete flag (COCO)
        BNE     CHCK                    ; is set

        LDA     ADDR                    ;when conversion done
                                        ; put answer in ACCA

LOOP    BRA     LOOP                    ;loop forever

        ORG     $1FFE                   ;define reset vector
        DW      $0100
```

The connections needed for this routine are:

Port C7, V_{RH} -----------> --- +5 Volts

|

/

\

Port C6, AN0 --------------> / 10-K Potential

\

/

|

V_{SS} --------------------> --- GND

The V_{RH} is connected to +5 volts. The HC05P6 does not have a low voltage reference for its A/D. The V_{RL} is connected to the chip's V_{SS}. Connect the GND pin of the voltage divider potential to V_{SS}. The potential will vary the voltage between V_{RH} and V_{SS}. This voltage is also fed into the converter channel AN0.

To predict what the A/D converter will read, use this relationship:

$$\frac{\text{A/D reading}}{255} = \frac{\text{pot voltage on AN0}}{\text{5 volts}}$$

Run the code by using these steps:

1. Download the code into the EVS with the EVM05 software.
2. Set the program counter to $100.
3. 'PC 100'
4. Type 'GO'
5. The routine will run and eventually hit the infinite loop. Press the ABORT button on the EVS to get out of the loop.
6. The A/D reading is in accumulator A, which is shown on the EVM05 screen.

Various voltage inputs with their appropriate A/D readings:

1. 0.0 V --> $00
2. 1.0 V --> $33
3. 2.5 V --> $80
4. 4.0 V --> $CC
5. 5.0 V --> $FF

The reading may be off by a couple of least significant bits (LSB) due to system noise.

A/D Converter Source Impedance

Reference Documents: MC68HC705P9 Rev. 1, page 3-8; MC68HC05P6, page 13-7; MC68HC705P6 Rev. 1, page 13-7; MC68HC05P9, page 10-7; MC68HC705P9 Rev. 1, page 13-8

Tracker Number: HC705P9.002 Revision: 1.00

Note 3 in the data books states:

3. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.

The A/D converter on the HC705P9 is a successive approximation converter. Thirty-two cycles are necessary for the A/D to execute one conversion. The first 12 cycles sample the voltage on the A/D input pin by charging an internal capacitor. During the last 20 cycles, a comparator successively compares the output of an internal D/A converter to the sampled analog input. Control logic changes the D/A converter value bit by bit until the D/A value and the sampled input match.

Note 3 refers to the first 12 cycles of the conversion process. If the source impedance is larger than 10 kΩ, more time will be needed for the internal RC charging of the capacitor to reach the voltage on the A/D input. The higher source impedance will affect A/D accuracy.

A/D Accuracy Specification

Reference Documents: MC68HC705P9, page 13-8; MC68HC05P6, page 13-7; MC68HC705P6 Rev. 1, page 13-7; MC68HC05P9, page 10-7; MC68HC705P9 Rev. 1, page 13-8

Tracker Number: HC705P9.010 Revision: 1.00

The A/D converter's accuracy specification has caused some confusion because of differing ways it can be interpreted.

The specification currently states:

Characteristic	Min	Max	Unit
Absolute Accuracy (4.0 > V > V_{DD}) (Note 2)	—	+/— 1-1/2	LSB

The correct reading is "plus or minus 1.5 LSBs."

However, some customers incorrectly read this as "plus or minus 1.0 LSB down to 0.5 LSBs" or as "plus or minus 1.5 LSBs."

Bootloader Code

HC705P6.004

Revision History

Date	Revision	Description
7/7/95	1.00	Includes tracker HC705P6.004.

Reference Document: Not applicable

Tracker Number: HC705P6.004 **Revision: 1.00**

The HC705P6 bootloader code is listed below for the current mask set 0E98K.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;             68HC705P6 EPROM BOOTLOADER PROGRAM                   ;
;             ==================================                   ;
;
;  Programmer:  Paul Brownlee, Western MCU Design Center
;  Date:        January 13, 1992
;  Description: This Bootloader code was adapted from Rev 3
;       (p9boot.3) of the bootloader code for the 68HC705P9. Major
;       differences are:
;              - Memory Map differences between devices
;              - MOR is statically programmed on the P6 (i.e.,
;                there is a MPGM bit in the Programming Register)
;       p6boot4 : done to fix a problem that caused adx $1000 to be
;               programmed from $0000 in the external memory due to
;               the external counter being imcremented one loop late
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;
        REGION0 "IO"
;       ORG     0
; I/O DEFINITIONS
;
PORTA   EQU     0x00       ; PORT A DATA
PORTB   EQU     0x01       ; PORT B DATA
PORTC   EQU     0x02       ; PORT C DATA
DDRA    EQU     0x04       ; PORT A DDR
DDRB    EQU     0x05       ; PORT B DDR
DDRC    EQU     0x06       ; PORT C DDR
EPROG   EQU     0x1C       ; EPROM PROGRAMMING REGISTER
;
; PORT A DEFINITIONS
;
DATAIN  EQU     PORTA      ; EPROM DATA INPUT PORT
DATAOUT EQU     PORTA      ; HOST DATA OUTPUT PORT
;
; PORT B DEFINITIONS
;
A12     EQU     5          ; ADDRESS BIT 12 FOR EPROM
VFYLED  EQU     6          ; BIT 6 DRIVES 'VERIFY' LED
PRGLED  EQU     7          ; BIT 7 DRIVES 'PROGRAMMING' LED
```

```
; PORT C DEFINITIONS
;
RST     EQU     1          ; COUNTER RESET
CLK     EQU     2          ; COUNTER CLK'
MODE1   EQU     3          ; PC3
MODE2   EQU     4          ; PC4
SYNC    EQU     5          ; PC5, SYNC INPUT
TEST    EQU     6          ; PORTC BIT6; - '1' GO BOOT,'0'GO 0x80 (RAM)

; MISCELLANEOUS DEFINITIONS
;
EPGM    EQU     0          ; EPROG  BIT0; - Vpp CONTROL BIT TO ARRAY
MPGM    EQU     1          ; EPROG  BIT1; - Vpp CNTL BIT TO MOR
ELAT    EQU     2          ; EPROG  BIT2; - EPROM ADDRESS LATCH BIT
ERASED  EQU     0x00       ; VALUE OF AN ERASED EPROM BYTE
INSTATC EQU     00000110B  ; INITIAL PORT C STATUS
INSTATB EQU     11000000B  ; INITIAL PORT B STATUS
;
        REGION0 "RAM"
;       ORG     0x50
; RAM VARIABLES
RAMSUB  DS     1           ; LOCATION OF RAM SUBROUTINE
ADDR    DS     3           ; EXTENDED ADDRESS FOR RAM SUBROUTINE
TEMP    DS     1           ; TEMPORARY RAM LOCATION
;

;*****************************************************************
;
;
;    INITIAL REGISTER VALUES
;       FCB     %XXXXXXXX   PORT A :- ROM DATA INPUT
;       FCB     %110        PORT B :- LED'S OFF, A12=0
;       FCB     %00000110   PORT C :- COUNTER IN RESET, CLK HIGH
;       FCB     %00000000   PORT A DDR :- ALL INPUTS
;       FCB     %111        PORT B DDR :- ALL OUTPUTS
;       FCB     %00000110   PORT C DDR :- PC7-PC3,PC0 INPUTS, REST
;                                                  OUTPUTS
;
;
;   RAM AREA IS INITIALIZED AS FOLLOWS;
;
;   LOCATION:-            INSTRUCTION:-
;
; RAMSUB  0x50     0xC7     STA    EPROM0
; ADDR    0x51     0x00
; ADDR+1  0x52     0x20
;         0x53     0x81     RTS
;
;
;*****************************************************************

        REGION  "BOOTST"
;       ORG     0x1F01
;
TABLE   DB     0xC7        ; 'STA EXTENDED' INSTRUCTION
        DB     0x00        ; ADDRESS $0020
        DB     0x20
        DB     0x81        ; 'RTS' INSTRUCTION
;
START   EQU     $
;
; CHECK PORT C, BIT 6 TO SEE IF USER WISHES TO JUMP TO
; RAM OR JUMP INTO THE BOOTLOADER PROGRAM.
;
;
        BRSET   TEST,PORTC,BOOT
        JMP     RAMSUB     ; GO TO RAM PROGRAM AT $0050
```

```
;
;
;
; SET UP PORTS, RAM SUBROUTINE, AND RAM VARIABLES
;
;
BOOT    EQU     $
;
;  INITIALIZE RAM SUBROUTINE AND VARIABLES
;
        LDX     #3
MOVE    LDA     TABLE,X     ; GET A BYTE FROM THE TABLE
        STA     RAMSUB,X    ; MOVE IT INTO RAM
        DECX                ; POINT TO THE NEXT BYTE TO BE MOVED
        BPL     MOVE        ; KEEP MOVING UNTIL ALL ARE IN PLACE
;
        BSR     INIT        ; INITIALIZE PORTS
;
        BRSET   MODE1,PORTC,PRGVERF  ; IF PC3=1, DO PROGRAMMING
        BRSET   MODE2,PORTC,VERIFY   ; IF PC4=1, DO VERIFY
;
; DUMP EPROM CONTENTS
;
DUMPE   COM     DDRA        ; MAKE PORT A ALL OUTPUT
        DEC     RAMSUB      ; PUT "LDA EXTENDED" IN RAMSUB ($C6)
        BSR     INC20       ; BUMP COUNTER TO ADDRESS $0020
DLOOP   JSR     RAMSUB      ; GET DATA FROM INTERNAL EPROM
        STA     DATAOUT     ; WRITE IT TO PORT A
        BSR     NXTADR      ; BUMP ADDRESS
        BNE     DLOOP       ; EXIT IF END ADDRESS
;
        WAIT


;*****************************************************************
;
; THE BOOTLOADER PROGRAM HAS 3 MODES OF OPERATION:
;
;   I. PROGRAM/VERIFY - PERFORMS A NORMAL PROGRAM CYCLE FOLLOWED ;                          BY
BY A VERIFY CYCLE WHICH HANGS IF THE EPROM ; IS NOT CORRECTLY PROGRAMMED.
;
;
;  II. VERIFY - PERFORMS ONLY A VERIFY CYCLE WHICH HANGS IF THE ;                     EPROM IS
NOT CORRECTLY PROGRAMMED.
;
;
; III. DUMP EPROM - DUMPS THE EPROM CONTENTS OF
;                                THE 705P9 TO PORT A
;
; WHEN COMING OUT OF RESET INTO THE BOOTLOADER PROGRAM (ASSUMING
; THAT PORT C PIN 6 ALLOWS YOU TO ENTER THE BOOTLOADER) THE STATE
; OF PORT C PINS 3 AND 4 DETERMINES WHICH MODE OF OPERATION THE
; PROGRAM WILL ENTER.
;
```

```
; THE GATE AND DRAIN STRESS TESTS CAN ALSO BE INVOKED THROUGH THE ; BOOTLOADER.
;
;     |--------------------------|-------------------------------|
;     |         PORT C           |                               |
;     |-------|--------|---------|             MODE              |
;     | PIN 5 | PIN 3  |  PIN 4  |                               |
;     |-------+--------+---------+-------------------------------|
;     |  SYNC |   1    |    1    |   PROGRAM/VERIFY   4 ms PULSE |
;     |-------+--------+---------+-------------------------------|
;     |  SYNC |   0    |    1    |   VERIFY                      |
;     |-------+--------+---------+-------------------------------|
;     |  SYNC |   0    |    0    |   DUMP EPROM                  |
;     |=======+========+=========+===============================|
;
;*****************************************************************


;*****************************************************************
; INITIALIZE PORTS B AND C TO THEIR INITIAL CONDITIONS.
;*****************************************************************

INIT    LDA     #INSTATC     ; GET PORTC INITIAL VALUE
        STA     PORTC        ; SET UP PORTC
        STA     DDRC         ; SET UP PORTC DATA DIRECTION
        LDA     #INSTATB
        STA     PORTB
        LDA     #11100000B
        STA     DDRB         ; SET UP PORT B DATA DIRECTION
        RTS
;
; PROGRAM THE EPROM WITH THE CONTENTS OF THE EXTERNAL EPROM
;
PRGVERF EQU     $
        BCLR    PRGLED,PORTB ; LIGHT 'PROGRAMMING' LED
        BSR     INC20        ; BUMP COUNTER TO ADDRESS $0020
;
; PROGRAM AN EPROM ADDRESS WITH DATA RECEIVED FROM PORTB.
; THE ADDRESS TO BE PROGRAMMED SHOULD BE PLACED IN LOCATION
; 'ADDR' & 'ADDR+1'.
;
PRGLOP  LDA     TEMP        ; GET DATA BYTE
        BEQ     SKIP        ; RETURN IF EQUAL TO ERASED STATE
                            ; ($00)
;
;
ZAP     LDX     ADDR        ; CHECK HI ADX BYTE FOR 1F
        CPX     #0x1F
        BNE     ARRAY       ; IF NOT 1F, PROG ARRAY BYTE
        LDX     ADDR+1      ; CHECK LO ADX BYTE FOR 00
        BNE     ARRAY       ; IF NOT ADX 1F00, GO PROG ARRAY
                            ; BYTE
        JSR     RAMSUB      ; ELSE, PROGRAM STATIC MOR BYTE
        BSET    MPGM,EPROG  ; APPLY Vpp TO MOR
        BRA     DELAY

ARRAY   BSET    ELAT,EPROG  ; GET HERE EVERY ADX <> MOR ($1F00)
        JSR     RAMSUB      ; WRITE ONE BYTE OF DATA
        BSET    EPGM,EPROG  ; APPLY Vpp TO CIRCUIT
DELAY   LDA     #8          ; PROGRAMMING PULSE = 4 MS @ 4MHz
                            ; OSC.

;*****************************************************************
; DELAY N mS SUBROUTINE. ON ENTRY, ACCUMULATOR SHOULD CONTAIN
; 2xDELAY WANTED IN MILLISECONDS.
; ( ASSUMES 4MHz OSCILLATOR FREQUENCY ).
;
;
```

```
DELNMS  LDX     #0xA6       ; 0.5 MS INNER LOOP
MS1     DECX
        BNE     MS1
        DECA                ; DECREMENT OUTER LOOP
        BNE     DELNMS
        CLR     EPROG       ; REMOVE Vpp FROM CIRCUIT (AND ELAT
                                        ; IF ARRAY)

SKIP    BSR     NXTADR      ; POINT TO NEXT ADDRESS
        BNE     PRGLOP      ; KEEP PROGRAMMING UNTIL DONE
        STA     ADDR+1      ; RESET LOW ORDER  ADDR TO $20
        CLR     ADDR        ; RESET HIGH ORDER ADDR TO $00
;
;
; VERIFY THE EPROM CONTENTS AGAINST EXTERNAL MEMORY.
;        ( ASSUMES 'RAMSUB' CONTAINS $C7 )
;
VERIFY  BSR     INIT        ; INITIALIZE PORTS B AND C
        INC     RAMSUB      ; CHANGE 'STA' TO 'EOR' EXTENDED
                            ; ($C8).
;
        BSR     INC20       ; INCREMENT THE COUNTER BY 20
CHECK   LDA     TEMP        ; GET DATA
        JSR     RAMSUB      ; COMPARE TO AN EPROM BYTE
        BNE     $           ; HANG IF THEY DON'T MATCH
        BSR     NXTADR      ; POINT TO NEXT ADDRESS TO BE
                            ; COMPARED
        BNE     CHECK       ; KEEP CHECKING BYTES UNTIL EPROM
                            ; END
;
DONE    BCLR    VFYLED,PORTB; INDICATE EPROM VERIFIED AS CORRECT
        WAIT                ; HANG
;
;*****************************************************************
;
;                     S U B R O U T I N E S
;
;*****************************************************************

;*****************************************************************
; ADVANCE COUNTER
;
ADCNT   BCLR    CLK,PORTC   ; PULSE COUNTER
        BRSET   SYNC,PORTC,$; WAIT FOR SYNC PULSE
        LDX     DATAIN      ; GET DATA
        STX     TEMP        ; SAVE DATA (USED BY PROG/VERF)
        BSET    CLK,PORTC
        RTS

;*****************************************************************
; INCREMENT COUNTER BY $20
;
INC20   LDA     #0x20
        BCLR    RST,PORTC   ; REMOVE RESET FROM COUNTER
BUMP    BSR     ADCNT
        DECA
        BNE     BUMP
        RTS
```

```
;******************************************************************
;
;  NXTADR SUBROUTINE
;
;  COMPUTES NEXT EPROM ADDRESS TO BE PROGRAMMED, VERIFIED, OR
;  DUMPED. INCREMENTS THE COUNTER ACCORDINGLY.
;  UPDATES RAMSUB ALSO. SKIPS THE RAM, BOOTSTRAP AND UNUSED AREAS.
;  RETURNS WITH Z=1 IF THE COMPUTED ADDRESS IS = $2000, MEANING
;  THAT A PASS THROUGH THE MEMORY MAP HAS BEEN COMPLETED.
;  OTHERWISE Z=0.
;  VALID ADDRESSES ARE: $0020-$004F, $0100-$12FF, $1F00 (MOR), AND
;  $1FF0-$1FFF.
;******************************************************************

NXTADR  LDA     ADDR       ; GET MS.BYTE OF LAST ADDRESS USED
        BNE     NOT04F     ; BRANCH IF NOT IN PAGE ZERO EPROM AREA
        LDX     ADDR+1     ; GET LS.BYTE OF LAST ADDRESS USED
        CPX     #0x4F      ; LAST BYTE OF PAGE ZERO EPROM?
        BNE     NOT04F     ; BRANCH IF NOT $004F
;
; MUST HAVE JUST ACCESSED LAST PAGE ZERO EPROM LOCATION IF HERE SO
; FORCE NEXT ADDRESS INCREMENT TO POINT TO MAIN EPROM
; AREA AT $0100.
;
        LDX     #0x100-1   ; POINT TO ADDRESS BELOW THE ONE WANTED
        STX     ADDR+1     ; PLACE IN LOCATION TO BE INCREMENTED
;
        LDA     #0xB0      ; INCREMENT COUNTER TO $00FF
INCLP   BSR     BUMP
;
; 16 BIT ADDRESS INCREMENT
;
NOT04F  BSR     ADCNT      ; INCREMENT COUNTER AND GET DATA
        INC     ADDR+1     ; INCREMENT LS. ADDRESS BYTE
        BNE     CMOR       ; ONLY INCREMENT MS. ADDRESS IF PAGE
                           ; BOUNDARY
        INC     ADDR       ; INCREMENT MS. ADDRESS
;
; LOOK OUT FOR HAVING GONE THROUGH THE ENTIRE MEMORY MAP.
;
CMOR    LDA     ADDR
        CMP     #0x20      ; WAS THAT THE END OF MEMORY ( $1FFF )?
        BEQ     GOBACK1    ; EXIT WITH Z=1 IF THE END WAS REACHED.
;
; CHECK FOR ADDRESS $0FFF BECAUSE A12 MUST GET SET FOR NEXT ACCESS
;
        LDX     ADDR+1     ; CHECK IF LO ADX IS 'FF' BY INC AND ZERO
                           ; CK.
        INX
        BNE     CK13       ; IF NOT, GO TO ADX 1300 CHECK

        CMP     #0x0F      ; IF LO ADX IS FF, CHECK FOR HI ADX=0F
        BLO     GOBACK     ; IF NOT, LEAVE WITHOUT SETTING A12
        BSET    A12,PORTB  ; ELSE SET A12 FOR UPPER 4K ADX RANGE
                           ; ENDIF

CK13    CMP     #0x13      ; IF $10 < ADDR < $13
        BLO     GOBACK     ; THEN CONTINUE USER EPROM SPACE
        BEQ     INC2MOR    ; ELSE CHECK IF WE NEED TO SKIP TO MOR
;
; ADDRESS = $1F01, $1FF1-$1FFF. IF $1F01 THEN SKIP OVER THE
; BOOTSTRAP
; AREA TO ADDRESS $1FF0, ELSE GOBACK
```

```
;
        LDA     ADDR+1     ; CHECK FOR LO BYTE = $01 (HI MUST = $1F)
        DECA               ; IF IT WAS A 1, ACC IS NOW ZERO
        BNE     GOBACK     ; IF ADDR+1<>1 THEN GOBACK

        LDA     #0xEF      ; ELSE, $1F01+$EF=$1FF0
        BSR     BUMP       ; MAKE COUNTER DO IT
        LDA     #0xF0      ; FORCE ADDR+1=F0 (ADDR ALREADY $1F)
        STA     ADDR+1     ; THUS 16-BIT ADX IS 1FF0
        BRA     GOBACK     ; AND RETURN WITH NEW ADX AND DATA

;
; ADDRESS = $1300 WHICH BEGINS UNIMPLEMENTED. THUS, ADVANCE TO MOR
; ($1F00)
;
INC2MOR LDA     #48        ; $1F00-$1300=3072(DECIMAL). THUS, SKIP
        STA     ADDR+1     ; 3072=48*64 ADDRESSES
LA      LDA     #64
        BSR     BUMP       ; ADDR+1 IS JUST USED FOR TEMP STORAGE BUT
        DEC     ADDR+1     ; DOING SO FINISHES WITH 00 IN ADDR+1 AS DESIRED
        BNE     LA
                           ; PREVIOUS LOOP FORCED LS. ADDRESS BYTE TO
                           ; $00
        LDA     #0x1F      ; FORCE MS. ADDRESS BYTE TO $1F
        STA     ADDR
GOBACK  LDA     TEMP       ; GET DATA BYTE
        LDX     #1         ; CLEAR Z BIT
GOBACK1 RTS
;*****************************************************************

      REGION    "BOOTVEC"
;       ORG     0x1EFA
;
        DW     [3]START    ; RESET VECTOR FOR ALL OF THEM

        END
?
```

Mask Set Errata 1

705P6MSE1(R1)

68HC705P6 8-Bit Microcontroller Unit

INTRODUCTION

This errata provides information pertaining to the page zero EPROM, mask option register, reliability information, and data corruption at address location $1000 applicable to the following 68HC705P6 MCU mask set device:

- 0E32A

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a four-character code consisting of a letter, two numerical digits, and a letter (for example, E32A). Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code (for example, 0E32A).

MCU DEVICE DATE CODES

Device markings indicate the week of manufacture and the mask set used. The data is coded as four numerical digits where the first two digits indicate the year and the last two digits indicate the work week. The date code "9115" would indicate the 15th week of the year 1991.

MCU DEVICE PART NUMBER PREFIXES

Some MCU samples and devices are marked with an "SC" or "XC" prefix. An "SC" prefix denotes special/custom device. An "XC" prefix denotes device is tested but is not fully characterized or qualified over the full range of normal manufacturing process variations. After full characterization and qualification, devices will be marked with the "MC" prefix.

PAGE ZERO EPROM

Page zero EPROM is not accessible. Do not use EPROM addresses $0020 through $004F.

MASK OPTION REGISTER (MOR)

The mask option register is not functional. Therefore, the following MCU device options are not changeable at this time.

- STOP Enable
- COP Disable
- Interrupt Edge Only
- SIOP MSB
- SIOP PH2/64
- Crystal/ Ceramic Resonator

RELIABILITY DATA

There is no reliability data at this time. Reliability of the MCU device, therefore, cannot be guaranteed. The 68HC705P6 is a prototype device. 0E32A mask set MCU devices have been tested at 25° C only. For 0E32A devices, the COP watchdog timer is not tested at 3 volts.

DATA CORRUPTION AT ADDRESS LOCATION $1000

When programming the 68HC705P6 MCU device with an external 2764 EPROM device, data corruption occurs at address location $1000. The data value at address location $0000 of the EPROM device is programmed at address location $1000 of the MCU device.

When programming the 68HC705P6 MCU, a bootloader circuit is used to copy the contents of a 2764 EPROM to the MCU EPROM. The MCU device bootloader code causes the data corruption problem. When the bootloader circuit 12-bit counter overflows from $FFF to $000, the data is fetched before address line A12 is pulled high. By the time data is fetched for address $1001, address line A12 is high and the remainder of the address locations program correctly.

To circumvent the MCU device data corruption problem, the user is advised to duplicate the data value of the assembled code at address location $1000 and insert the duplicated value into address location $0000 of the external EPROM device.

Mask Set Errata 2
705P6MSE2(R2)

68HC705P6 8-Bit Microcontroller Unit

INTRODUCTION

This errata provides information pertaining to the mask option register, reliability information, and data corruption at address location $1000 applicable to the following 68HC705P6 MCU mask set devices:

- 1E32A
- 2E32A

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a four-character code consisting of a letter, two numerical digits, and a letter (for example, E32A). Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code (for example, 1E32A).

MCU DEVICE DATE CODES

Device markings indicate the week of manufacture and the mask set used. The data is coded as four numerical digits where the first two digits indicate the year and the last two digits indicate the work week. The date code "9115" would indicate the 15th week of the year 1991.

MCU DEVICE PART NUMBER PREFIXES

Some MCU samples and devices are marked with an "SC" or "XC" prefix. An "SC" prefix denotes special/custom device. An "XC" prefix denotes device is tested but is not fully characterized or qualified over the full range of normal manufacturing process variations. After full characterization and qualification, devices will be marked with the "MC" prefix.

MASK OPTION REGISTER (MOR)

Programming time for the MOR byte ($1F00) is longer than that for the other EPROM cells. Therefore, to ensure appropriate programming of the MOR, it is recommended that the MOR byte be programmed at least twice, but no more than three times with V_{PP} = 17.5 volts.

The RC bit of the MOR (bit 6) is hard-wired to the crystal oscillator configuration. No attempts should be made to set this bit to the RC configuration, as this option currently is unavailable.

RELIABILITY DATA

There is no reliability data at this time. Reliability of the MCU device, therefore, cannot be guaranteed. The 68HC705P6 is a prototype device. 1E32A mask set MCU devices have been tested at 25° C only.

DATA CORRUPTION AT ADDRESS LOCATION $1000

When programming the 68HC705P6 MCU device with an external 2764 EPROM device, data corruption occurs at address location $1000. The data value at address location $0000 of the EPROM device is programmed at address location $1000 of the MCU device.

When programming the 68HC705P6 MCU, a bootloader circuit is used to copy the contents of a 2764 EPROM to the MCU EPROM. The MCU device bootloader code causes the data corruption problem. When the bootloader circuit 12-bit counter overflows from $FFF to $000, the data is fetched before address line A12 is pulled high. By the time data is fetched for address $1001, address line A12 is high and the remainder of the address locations program correctly.

To circumvent the MCU device data corruption problem, the user is advised to duplicate the data value of the assembled code at address location $1000 and insert the duplicated value into address location $0000 of the external EPROM device.

Mask Set Errata 3

705P6MSE3

68HC705P6 8-Bit Microcontroller Unit

INTRODUCTION

This errata provides information pertaining to the mask option register applicable to the following 68HC705P6 MCU mask set device:

- 0E98K

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a four-character code consisting of a letter, two numerical digits, and a letter (for example, E98K). Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code (for example, 0E98K).

MCU DEVICE DATE CODES

Device markings indicate the week of manufacture and the mask set used. The data is coded as four numerical digits where the first two digits indicate the year and the last two digits indicate the work week. The date code "9115" would indicate the 15th week of the year 1991.

MCU DEVICE PART NUMBER PREFIXES

Some MCU samples and devices are marked with an "SC" or "XC" prefix. An "SC" prefix denotes special/custom device. An "XC" prefix denotes device is tested but is not fully characterized or qualified over the full range of normal manufacturing process variations. After full characterization and qualification, devices will be marked with the "MC" prefix.

MASK OPTION REGISTER (MOR)

Programming time for the MOR byte ($1F00) is longer than that for the other EPROM cells. Therefore, to ensure appropriate programming of the MOR, it is recommended that the MOR byte be programmed at least twice, but no more than three times with V_{PP} = 17.5 volts.

The RC bit of the MOR (bit 6) is hard-wired to the crystal oscillator configuration. No attempts should be made to set this bit to the RC configuration, as this option is currently unavailable.