MOTOROLA

TECHNICAL UPDATE

MC68HC05J3
MC68HC705J3

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

Part Specific

TECHNICAL UPDATE

Modules

Computer Operating Properly (COP)

COP0COPRT2

Revision History

Date	Revision	Description
7/16/95	1.00	Includes tracker HC705K1.005R2.

COP Timeout Test

Reference Document: Not applicable

Tracker Number: HC705K1.005 Revision: 2.00

This program tests the timeout on the COP module COP0COPRT2 and can be used on any MCU in the HC05 Family that has the COP0COPRT2 module. The MC68HC705K1 was used to verify operation. Memory and reset vectors may need to be changed to work properly with a particular MCU.

```
*****************************************************************
*
* Program Name: 7K1_COP.ASM ( COP Test on the HC705K1 )
* Revision: 1.00
* Date: May 25, 1993
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
*       ********************************
*       *       Revision History       *
*       ********************************
*
*       Rev     1.00    03/26/93        M.R. Glenewinkel
*                       Initial Release
*
```

```
*****************************************************************
*
* Program Description:
*
*       This program is a simple routine that tests the COP
*       timeout on the HC705K1 MCU. The HC705K1 was programmed
*       with the M68HC705KICS board. The part was then tested
*       for COP resets on a protoboard. If the COP is working
*       correctly, PortA will toggle on approximately
*       1/2 sec intervals.
*
*****************************************************************

PORTA   equ     $00
DDRA    equ     $04
MOR     equ     $17

        ORG     MOR
        DB      $01             ;enable COP

        ORG     $200

START   lda     #$FF            ;make port A all output
        sta     DDRA            ;

        com     $E0             ;complement RAM mem $E0
        lda     $E0             ;ACCA <- ($E0)
        sta     PORTA           ;port A <- (ACCA)


DONE    NOP                     ;branch into an infinite loop
        BRA     DONE            ;waiting for a COP timeout

        ORG     $03FE           ;define reset vector
        DW      START
```

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 (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 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.
7/19/95	1.1	HC705P9.005R2 revised to HC705P9.005R3.

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 Advance Information Data Sheet, 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 MC68HC705C8 on the HC05EVM board and will work on an MC68HC05C4.

```
*************************************************************
*
* 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: 3.00

This input capture test was written for the timer module TIM1IC1OC_A and was tested on the MC68HC705P9. To use this test properly with other MCUs in the HC05 Family, reset vectors and memory map equates may have to be changed.

For memory map information on specific parts, refer to the applicable technical data book for the part.

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

Part Specific

MC68HC705J3

HC705J3

Revision History

Date	Revision	Description
7/16/95	1.00	Includes tracker HC705J3.002, HC705J3.004, and HC705J3.005.

Programming Voltage

Reference Document: MC68HC05J3/D, page 1.

Tracker Number: HC705J3.002 Revision: 1.00

The programming voltage (V_{PP}) for the MC68HC705J3 is 16.5 volts.

Bootloader Code

Reference Document: Not applicable

Tracker Number: HC705J3.004 Revision: 1.00

This bootloader listing applies to these two mask sets for the MC68HC705J3:

- 0E23B
- 1E23B

```
*********************************************************
*                                                       *
*             68HC705J3 EPROM BOOTLOADER PROGRAM        *
*             ==================================        *
*                                                       *
*              Modified from J2 Loader 7-APR-92         *
*                      jmk - Mot munich                 *
*              This version:- 5/30/90 - REV 2           *
*              Timing delays based on 2 Mhz bus         *
*              REV 1 was never used                     *
*              REV 3 - fixed drain stress, made prog    *
*                      time 4 ms.  NCN  3/26/91         *
*********************************************************
```

```
*
* I/O DEFINITIONS
*
PORTA   EQU     $00     PORT A DATA
PORTB   EQU     $01     PORT B DATA
DDRA    EQU     $04     PORT A DDR
DDRB    EQU     $05     PORT B DDR
*
* EPROM CONTROL REGISTER
*
PROG    EQU     $1C     EPROM CONTROL
EPTST   EQU     $40     Eprom Test
EPTS1   EQU     $10     Test select bit 1
EPTS0   EQU     $08     Test select bit 0
ELAT    EQU     $04     Eprom latch bit
ELATB   EQU       2
EPGM    EQU     $01     PROG  BIT0; - Vpp CONTROL BIT
EPGMB   EQU       0
*
* MEMORY MAP DEFINITIONS
*
RAM     EQU     $80      BEGINNING OF RAM
SWIV    EQU     RAM+4    Software interrupt
IRQV    EQU     RAM+7    IRQ & KEYB Vector
T8INT   EQU     RAM+10   Core timer
T16CAP  EQU     RAM+13   t16 input capture
T16CMP  EQU     RAM+16   t16 output compare
T16OVL  EQU     RAM+19   t16 overflow
NUSED1  EQU     RAM+22   Not used INT3
MORADR  EQU     $0F00    Address of mask option register
BOOTST  EQU     $0F01    START OF BOOTSTRAP ROM AREA
BOOTV   EQU     $0FE0    START OF BOOTSTRAP VECTOR AREA
VECTOR  EQU     $0FF0    START OF USER VECTOR AREA
*
* RAM VARIABLES
*
        ORG     RAM
*
RAMSUB  RMB     1        LOCATION OF RAM SUBROUTINE
ADDR    RMB     3        EXTENDED ADDRESS FOR RAM SUBROUTINE
TEMP    RMB     1        TEMPORARY RAM LOCATION
SAVA    RMB     1        TEMPORARY LOCATION FOR ACC. A
SAVE    RMB     1        ANOTHER TEMPORARY LOCATION FOR ACC.
*
* PORT A DEFINITIONS
*
DATAIN  EQU     PORTA     ROM DATA INPUT PORT
*
* PORT B DEFINITIONS
*
SYNC    EQU     0         PB0, SYNC INPUT
MODE1   EQU     2         PB2
MODE2   EQU     3         PB3
RST     EQU     4         COUNTER RESET
CLK     EQU     5         COUNTER CLK'
VFYLED  EQU     3         BIT 3 DRIVES 'VERIFY' LED
PRGLED  EQU     2         BIT 2 DRIVES 'PROGRAMMING' LED
        page

* MISCELLANEOUS DEFINITIONS
*
ERASED  EQU     $00       VALUE OF AN ERASED EPROM BYTE
INSTAT  EQU     %00111100 INITIAL PORT B STATUS
TEST    EQU     1         PORTB BIT1; - '1' GO BOOT,'0'GO $81 (RAM)
TSTREG  EQU     $1F       TEST REGISTER
*
```

```
*****************************************************************
*
*    INITIAL REGISTER VALUES
*       FCB     %00000000   PORT A :- ROM DATA INPUT
*       FCB     %  101000   PORT B :- COUNTER IN RESET
*       FCB     %00000000   PORT A DDR :- ALL INPUTS
*       FCB     %  111100   PORT B DDR :- PB0,1 ARE USED AS INPUT
*
*   RAM AREA IS INITIALISED AS FOLLOWS;
*
*   LOCATION:-            INSTRUCTION:-
*
* RAMSUB  $80     $C7     STA   extended
* ADDR    $81     $00
* ADDR+1  $82     $00
*         $83     $81     RTS
*
*
*****************************************************************
        page

         ORG     MORADR
         FCB     0              just for programming

         ORG     BOOTST
*
TABLE   FCB     $C7             'STA EXTENDED' INSTRUCTION
        FCB     $00             ADDRESS $0000
        FCB     $00             .
        FCB     $81             'RTS' INSTRUCTION
TABEND  EQU     *
*
START   EQU     *
*
* CHECK PORT B, BIT 1 TO SEE IF USER WISHES TO JUMP TO
* RAM OR JUMP INTO THE BOOTLOADER PROGRAM.
*
*
        BRSET   TEST,PORTB,BOOT
        JMP     RAM+1           GO TO RAM PROGRAM AT $0081
*
* SET UP PORTS, RAM SUBROUTINE, AND RAM VARIABLES
*
*
BOOT    EQU     *
*
*  INITIALISE RAM SUBROUTINE AND VARIABLES
*
        LDX     #TABEND-TABLE-1
MOVE    LDA     TABLE,X         GET A BYTE FROM THE TABLE
        STA     RAM,X           MOVE IT INTO RAM
        DECX                    POINT TO THE NEXT BYTE TO BE MOVED
        BPL     MOVE            KEEP MOVING UNTIL ALL ARE IN PLACE
*
        BRSET   MODE1,PORTB,MAYPRG
        BRSET   MODE2,PORTB,VERIFY
*
* DUMP EPROM CONTENTS
*
DUMPE   BSR     INIT
        BSR     INC700          BUMP COUNTER TO START OF EPROM
        COM     DDRA            MAKE PORT A ALL OUTPUT
        DEC     RAMSUB          PUT `LDA EXTENDED' IN RAMSUB ($C6)
DLOOP   JSR     RAMSUB          GET DATA STARTING AT $0700
        STA     PORTA           PUT DATA ON PORT A
        JSR     NXTADR          BUMP ADDRESS
        BNE     DLOOP           EXIT IF END ADDRESS
```

```
*
        WAIT
*
MAYPRG  BRSET   MODE2,PORTB,PRGVERF
        BRSET   SYNC,PORTB,DTEST
*
*DO THE GATE STRESS TEST
*
GTEST   LDA     #EPTST          EPTST=1, TS1:TS0=0:0
NOCOM   STA     PROG            ENABLE GATE STRESS TEST
        TXA                     WRITE $FF DATA (MAINLY FOR DRAIN STRESS)
        BSR     ZAP
        WAIT
*
* DO THE DRAIN STRESS TEST
*
DTEST   LDA     #EPTST|EPTS0    EPTST=1, TS1:TS0=0:1
        BRA     NOCOM
        page

*********************************************************************
*
* THE BOOTLOADER PROGRAM HAS 3 MODES OF OPERATION:
*
*   I. PROGRAM/VERIFY - PERFORMS 1 NORMAL PROGRAM CYCLES FOLLOWED BY A
*                       VERIFY CYCLE WHICH HANGS IF THE EPROM IS NOT
*                       CORRECTLY PROGRAMED.
*
*  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 705J1 TO PORT A
*
* WHEN COMING OUT OF RESET INTO THE BOOTLOADER PROGRAM (ASSUMING THAT
* PORT B PIN 1 ALLOWS YOU TO ENTER THE BOOTLOADER) THE STATE OF
* PORT B PINS 3 AND 2 DETERMINES WHICH MODE OF OPERATION THE
* PROGRAM WILL ENTER.
*
* THE GATE AND DRAIN STRESS TESTS CAN ALSO BE INVOKED THROUGH THE BOOTLOADER.
*
*    |-------------------------|--------------------|
*    |         PORT B          |                    |
*    |-------|--------|--------|       MODE         |
*    | PIN 0 | PIN 2  |  PIN 3 |                    |
*    |-------+--------+--------+--------------------------------|
*    |  SYNC |   1    |    1   |   PROGRAM/VERIFY 1.5 ms PULSE  |
*    |-------+--------+--------+--------------------------------|
*    |  SYNC |   0    |    1   |   VERIFY                       |
*    |-------+--------+--------+--------------------------------|
*    |  SYNC |   0    |    0   |   DUMP EPROM                   |
*    |=======+========+========+================================|
*    |   0   |   1    |    0   |   GATE STRESS TEST             |
*    |-------+--------+--------+--------------------------------|
*    |   1   |   1    |    0   |   DRAIN STRESS TEST            |
*    |=======|========|========|================================|
*
*********************************************************************
*
```

```
* INCREMENT COUNTER BY $700
*
INC700  LDA     #7
        STA     SAVE            SAVE ACC.
INC100  CLRA
        BCLR    RST,PORTB       REMOVE RESET FROM COUNTER
BUMP    JSR     NXTADR
        LDA     SAVA            RECOVER ACCUMULATOR
INCFF   DECA
        BNE     BUMP
        DEC     SAVE
        BNE     INC100
        RTS
*
* ADVANCE COUNTER
*
ADCNT   BCLR    CLK,PORTB               PULSE COUNTER
        BRSET   SYNC,PORTB,*            WAIT FOR SYNC PULSE
        LDX     DATAIN                  GET DATA
        STX     TEMP                    SAVE DATA
        BSET    CLK,PORTB
        RTS
        page

*********************************************************
* INITIALIZE PORTB AND ITS DDR
*********************************************************

INIT    EQU     *
        LDA     #%00111100
INITV   STA     DDRB                    PB2-5 OUTPUT
INIT1   LDA     #INSTAT                 GET INITIAL STATE FOR PORTB
        STA     PORTB
        RTS
*
* PROGRAM THE EPROM WITH THE CONTENTS OF THE EXTERNAL ROM
*
PRGVERF BSR     INIT
        BCLR    PRGLED,PORTB            LIGHT 'PROGRAMMING' LED
*
        BSR     INC700
*
PRGLOP  BSR     PRGSUB                  PROGRAM ONE EPROM BYTE
        BSR     NXTADR                  POINT TO NEXT ADDRESS
        BNE     PRGLOP                  KEEP PROGRAMMING UNTIL DONE
        CLR     ADDR                    RESET HIGH ORDER ADDR TO $02
        BSR     INIT1
        BRA     VERIFY1
*
************************************************************
*
* VERIFY THE EPROM CONTENTS AGAINST EXTERNAL MEMORY.
*        ( ASSUMES 'RAMSUB' CONTAINS $C7 )
*
VERIFY  LDA     #%00111000      KEEP 'PROG' PIN AS INPUT
        BSR     INITV
VERIFY1 INC     RAMSUB          CHANGE 'STA' TO 'EOR' EXTENDED ($C8).
*
        BSR     INC700
*
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
        page

*********************************************************************
*                                                                   *
*                     S U B R O U T I N E S                         *
*                                                                   *
*********************************************************************
*
* PROGRAM AN EPROM ADDRESS WITH DATA RECEIVED FROM PORTB.
* THE ADDRESS TO BE PROGRAMMED SHOULD BE PLACED IN LOCATION
* 'ADDR' & 'ADDR+1'.
*
*
PRGSUB  LDA     TEMP            GET DATA BYTE
        BEQ     SKIP            RETURN IF EQUAL TO ERASED STATE ($00)
*
ZAPSUB  BSR     ZAP
        BCLR    EPGMB,PROG      REMOVE Vpp FROM CIRCUIT
         BCLR    ELATB,PROG      CLEAR THE LAT BIT
SKIP    RTS
*
ZAP     BSET    ELATB,PROG
        JSR     RAMSUB          WRITE ONE BYTE OF DATA
        BSET    EPGMB,PROG      APPLY Vpp TO CIRCUIT
        LDA     #8              4 MS PROGRAMMING PULSE LENGTH

*********************************************************************
*
*
* DELAY N mS SUBROUTINE. ON ENTRY, ACCUMULATOR SHOULD CONTAIN
* 2xTIME DELAY WANTED IN MILLISECONDS.
* ( ASSUMES 4MHz OSCILLATOR FREQUENCY ).
*
*
DELNMS  LDX     #$A6        0.5 MS INNER LOOP
MS1     DECX
        BNE     MS1
        DECA                DECREMENT OUTER LOOP
        BNED    ELNMS
        RTS                 RETURN AFTER WANTED DELAY
        page

*********************************************************************
*
*  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 = $0800, MEANING THAT
*  A PASS THROUGH THE MEMORY MAP HAS BEEN COMPLETED. OTHERWISE Z=0.
*
NXTADR  STA     SAVA
        BSR     ADCNT           INCREMENT COUNTER AND GET DATA
        INC     ADDR+1          INCREMENT LS. ADDRESS BYTE
        BNE     CMOR            CHECK FOR MOR ADDRESS ($F00)
        INC     ADDR            INCREMENT MS. ADDRESS
*
* LOOK OUT FOR HAVING GONE THROUGH THE ENTIRE MEMORY MAP.
*
CMOR    LDA     ADDR            READ MS. ADDRESS
        CMP     #$10            WAS THAT THE END OF MEMORY ( $0FFF )?
        BEQ     GOBACK1         EXIT WITH Z=1 IF THE END WAS REACHED.
*
```

```
* LOOK OUT FOR HAVING ACCESSED THE LAST LOCATION IN THE MAIN BLOCK
*
        CMP     #$0F            WAS THAT THE END OF THE MAIN BLOCK
        BNE     GOBACK          BRANCH IF STILL WITHIN THE MAIN BLOCK
        LDA     ADDR+1          CHECK FOR LS. ADDRESS = $01
        CMP     #1
        BNE     GOBACK
*
* SKIP OVER THE BOOTSTRAP AREAS
*
LA      LDA     #239
LX      JSR     ADCNT
        DECA
        BNE     LX
*
INMAIN  LDA     #$F0            FORCE LS. ADDRESS BYTE TO $F0
        STA     ADDR+1          .
GOBACK  LDA     TEMP            GET DATA BYTE
        LDX     #1              CLEAR Z BIT
GOBACK1 RTS
*****************************************************************
*
          ZMB    BOOTV-1-*      Fill with zeroes

        ORG     BOOTV-1
CHKSUM  FCB     0

        ORG     BOOTV

         FDB     NUSED1
         FDB     T16OVL
         FDB     T16CMP
         FDB     T16CAP
         FDB     T8INT
         FDB     IRQV
         FDB     SWIV
RESET   FDB     START           RESET VECTOR
*
        END
```

COP Documentation Errata

Reference Document: MC68HC05J3/705J3 Technical Data book, MC68HC05J3/D, page 1-3

Tracker Number: HC705J3.005 Revision: 1.00

On page 1-3 of the MC68HC05J3 Technical Data book, incorrect information concerning the COP bit in the MOR register of the MC68HC705J3 is included.

Currently, the incorrect wording reads:

1 (set) — COP watchdog timer is disabled.

0 (clear) — COP watchdog timer is enabled.

Replace it with this correct wording:

1 (set) — COP watchdog timer is enabled.

0 (clear) — COP watchdog timer is disabled.