MOTOROLA

TECHNICAL UPDATE

MC68HC05P4

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

CPU

HC05CPU

Revision History

Date	Revision	Description
8/7/95	2.00	Includes trackers HC05CPU.001, HC705C8.002R2, HC705C8.017, HC705C8.018R2, HC705C8019 and HC05P4.002.

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; MC68HC05P8/ D, page 4-15

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.

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
```
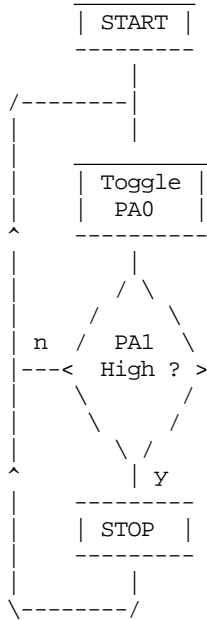
```
****************************************
* Start                                *
* Main Loop of code                    *
****************************************
        ORG     $0180           ; Begin code in EPROM
Start   LDA     #$01
        STA     DDRA            ; Set port A0 to output, leave
                                ;  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
```

Computer Operating Properly (COP)

COP0COP_A

Revision History

Date	Revision	Description
8/4/95	1.00	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:

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

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

Serial Input/Output Port (SIOP)

SIOP_A

Revision History

Date	Revision	Description
8/4/95	1.00	Includes trackers HC705P9.008R2 and HC705P9.015.

SIOP Applications Example

Reference Document: MC68HC705P9/D, page 9-1

Tracker Number: HC705P9.008   **Revision: 2.00**

```
*******************************************************************************
*******************************************************************************
*                                                                             *
*                    SIOP program example for HC705P9                         *
*                                                                             *
*******************************************************************************
*                                                                             *
* Program Name: SIOPP9.ASM                                                    *
* Date: 12/12/91                                                              *
* Written By: Robert Chretien                                                 *
*           Motorola CMCU Applications                                        *
* Assembled Under: P&E Microcomputer Systems, Inc.  IASM05                    *
*                                                                             *
* Program Description:                                                        *
*                                                                             *
*     This was written for the timer module SIOP_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.                                              *
*                                                                             *
*         This program shows how to use the basics of the MC68HC705P9       *
*         SIOP subsystem. It is meant to be used in a stand alone mode,      *
*         or with an appropriate evaluation/emulation system.                *
*                                                                             *
*         This program writes constant data out the SIOP and ignores        *
*         any data that may be transferred in.                               *
*                                                                             *
*                                                                             *
*******************************************************************************
```

```
*                                                                    *
* Revision Descriptions:                                             *
* 1. Rev 0.00: Original program.                                     *
* 2. Rev 1.00: Documentation touchup. David Yoder 6/14/93            *
* 3. Rev 2.00: Reduced cycle count. David Yoder 6/17/93              *
*                                                                    *
*                                                                    *
*********************************************************************
*********************************************************************
*
*
*

SDR        equ    $000C
SCR        equ    $000A
SSR        equ    $000B

           org    $0180
start      lda    #$50
           sta    SCR            ; Enable the SIOP system in master mode.
           lda    SSR            ; Load the status register so that writing
                                 ; to the data register will clear the
                                 ; complete flag (SPIF).
send       lda    #$AA
           sta    SDR            ; Send $AA out of the SIOP. Writing to
                                 ; SDR initiates the transfer and clears
                                 ; the complete flag (SPIF).
                                 ;
                                 ; To view output (SDO) and/or SCK adjust
                                 ; scope for 2usec horizontal resolution.
                                 ;
chkflag    lda    SSR            ; Load status register and test bit 7
           and    #$80           ; (SPIF) to see if transfer is complete.

           beq    chkflag        ; If SIOP transfer is not finished, reload
                                 ; status register.

*flagset   lda    SSR             ; Uncomment these instructions in order to
*          bra    *               ; view the status register after the
                                 ; transfer is complete. Just hit abort and
                                 ; the value of the status register will be
                                 ; left in the accumulator.
                                 ;
           bra    send           ; Branch in order to send another byte.

           org    $1FFE
           fdb    start          ; Reset vector

*                                                                    *
*                                                                    *
*                                                                    *

           end
```

SIOP Pin Functions

Reference Document: MC68HC705P9 Technical Data book Rev. 1, page 9-2

Tracker Number: HC705P9.015   Revision: 1.00

When the serial input/output port (SIOP) is enabled and then disabled, data values in the port B data register and port B data direction register are changed. This occurs because the SIOP uses these two registers. Operation of both the master and slave modes of the SIOP have this effect.

Timer

TIM1IC1OC

Revision History

Date	Revision	Description
8/4/95	1.00	Includes trackers HC05C4.002R2, HC05C4.003R2, and HC705P9.005R3.

Input Capture/Output Compare Code Snippet

Reference Document: Not applicable

Tracker Number: HC05C4.002   Revision: 2.00

```
****************************************************************
*
*       Program Name: ICOCC4.ASM
*       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
*               2.00          Memory map disclaimer added
*
****************************************************************
*
*       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 listing shows the procedure of using the output compare function driven by an interrupt to produce a square wave.

```
****************************************************************
*
* 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
*             2.00    Memory map disclaimer added
```

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

```
******************************************************************
*
* Program Name: P9_INCAP.ASM ( Input Capture Test for the P9EVS )
*
* 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
*       Rev     2.00    Documentation added
*       Rev     3.00    Memory map note added
******************************************************************
*
* 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

Revision History

Date	Revision	Description
8/4/95	1.00	Includes tracker HC05P4mse1.

This errata provides information pertaining to the serial input-output port (SIOP) master mode initial faulty transmission applicable to the following MC68HC05P4 MCU mask set devices:

•D25A

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 (e.g., D25A). Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code (e.g., 0D25A).

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.

Whenever contacting a Motorola representative for assistance, please have the MCU device mask set and date code information available.

Specifications and information herein are subject to change without notice.

SIOP MASTER MODE INITIAL FAULTY TRANSMISSION

A design flaw has been identified that affects only the first transmission after enabling the SIOP in the master mode. Subsequent transmissions from the SIOP are correct.

This error is caused by a delay between the serial port enable (SPE) and the master mode select (MSTR) bits in the SIOP control register. If the SPE and MSTR bits are set with the same write instruction, a race condition occurs in the SIOP logic. This race condition causes the SIOP to be enabled in the slave mode before recognizing the setting of the MSTR bit. In this short period of time, the serial clock (SCK) signal is propagated throughout the SIOP logic. When the MSTR bit is finally recognized, the SCK line is pulled high. If the state of the SCK line was initially low before enabling the SIOP, then the SIOP logic will recognize a low (zero) to high (one) transition on the SCK line and transmit one bit. When the next transmission is made, which is the first true transmission by the user, a data collision will result because a transmission has already been started by the previous SCK transition. The user may also notice that the first transmission has only 7 low to high SCK transitions.

One of two things may be done to avoid this problem. The MSTR bit can be set first, and then the SIOP enabled in a subsequent write operation. An example is shown below:

```
bset  4,$0A

bset  6,$0A
```

The SCK line may also be pulled high when the SIOP is enabled. This may be done by a pull-up device or by making PB7/SCK an output line and setting it high. The pull-up device, however, does not cover the possibility of the pin being an output line driving a low signal immediately before the SIOP is enabled in the master mode. If PB7/SCK line is held high, the SPE and MSTR bits may be set at the same time, as shown below:

```
lda  #$90

sta  $0A
```

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.