

---

# Mask Set Errata for Mask 0L47P

---

## Introduction

This mask set errata applies to the mask 0L47P for these products:

- MC68HC908AP64
- MC68HC908AP32
- MC68HC908AP16
- MC68HC908AP8

---

## MCU Device Mask Set Identification

The mask set is identified by a 5-character code consisting of a version number, a letter, two numerical digits, and a letter, for example 0L47P. All standard devices are marked with a mask set number and a date code.

## SPI Slave Mode Operation Issue

SE122-SPI-slave

### Description

This errata describes a voltage and noise sensitivity issue with the SPI while in slave mode. The SPI, while in slave mode, can experience transmission errors resulting in simultaneous receive and transmit errors. This is attributed to noise corrupting the slave clock input to the SPI module.

### Workaround

You must use fault-tolerant software and/or hardware handshaking protocol that can detect errant data and request re-transmission of the data from the SPI master. The SPI master in-turn must be able to detect anomalous data from the slave SPI and request re-transmission of the data from the SPI slave.

**NOTE:**

*Use of the SPI in slave mode at a nominal voltage of 3.3 V virtually eliminates the issue. All designs should incorporate decoupling capacitors very close to the MCU to avoid noise injection from the application.*

## Brief Pause on Standard Input/Output Pins during Power-On-Reset

SE95-I/O\_POR

### Description

During a power-on-reset, a brief pulse may appear on the following I/O pins: PTA0-PTA7, PTB4-PTB7, PTC0-PTC5, and PTD0-PTD7. The brief pulse, which is due to the turn-on delay of the internal regulator, can drive active-high output circuits (for example, relays or NPN transistors) momentarily. The turn-on delay is greater for slow rising VDD supplies. Input circuits are not affected.

### Workaround

Add an external RC filter to the output port pin, where necessary, for control sensitive applications.

In applications where these brief pulses could cause undesirable effects, add an external RC filter between the MCU output pin and the circuit that it drives.

## ROM-Resident Routines

SE72-ROM

Do not use the following ROM-resident routines:

- ERARNGE
- MON\_ERARNGE
- EE\_WRITE
- EE\_READ

A coding error has been found in the these routines which may cause accidental erasure of the reset vector at \$FFFF.

**Workaround for EE\_WRITE and EE\_READ**

The workaround is to have routines stored in the user FLASH area. An implementation is provided in the Motorola application note: *Using FLASH as EEPROM on the MC68HC908GP32*, which can be downloaded from [http://e-www.motorola.com/files/microcontrollers/doc/app\\_note/AN2183.pdf](http://e-www.motorola.com/files/microcontrollers/doc/app_note/AN2183.pdf).

**Workaround for ERARNGE and MON\_ERARNGE**

The following routine can be used as a workaround for ERARNGE and MON\_ERARNGE. The routine is intended to be stored in an area of FLASH, loaded and executed in RAM when called.

```
#####
*# Includes
#####
        include          "ap64R2.asm"                ; register and bit equates for AP64

#####
*# Variables
#####
.first SECTION short

;V_ADDRH_IN      ds.b      2
V_ADDRH_IN       ds.b      1
V_ADDRL_IN       ds.b      1

V_FLASH_ADDR     ds.b      2
V_TEST_COUNT     ds.b      2

RAMPTR           equ       *
V_RAMPTR_H       ds.b      1
V_RAMPTR_L       ds.b      1

Q_RAM_Blk_Erase equ       *

;CONFIG1         equ       $001F
;PORTA           equ       $0
;DDRA            equ       $04
```

## ROM-Resident Routines SE72-ROM

```

FLCR                equ        $FE08
b_ERASE             equ        1
b_HVEN              equ        3
FLBPR               equ        $FE09

MORVALUE            equ        %11111111
PRGRNGE             equ        $FC34
LDRNGE              equ        $FC00
ERARNGE             equ        $FCE4
WRITE_EE            equ        $FF36
READ_EE             equ        $FD5B

;RAMPTR             equ        $100
BUSSPEED            equ        $0A                ; using 10MHz oscillator CLK input
DATASIZE             equ        $FF
ERASEBLOCK           equ        $1000
;ENDBLOCK           equ        $FC00

;ENDBLOCK           equ        $E000
ENDBLOCK             equ        $C000
BLOCKSIZE            equ        $200

TESTCOUNT          equ        $20

FLASHSTART           equ        $860
RAM_BEGIN            equ        $60
RAM_END              equ        $85F

.data SECTION
#####
*# Functions
#####
        xdef     main
        xdef     DUMMY_ISR
        xdef     IRQ_ISR

        org     FLASHSTART

main:

        sei
        sta     $FFFF                ; reset COP

        mov     #$2b, CONFIG1
        mov     #$01, CONFIG2

        ldhx   #($85F+1)              ; reset stack ptr to end of RAM
        txs
        lda     #$FE
        sta     FLBPR                  ; protect vector table only

        lda     #%00000001
        sta     DDRA                    ; PTA0 as o/p

;        jsr    PLL_on2M5                ; add back PLL init if not using clk I/P

```

\*----- Clear RAM -----\*

```

        ldhx    #(RAM_END-RAM_BEGIN)
Clear_RAM:
        clr     (RAM_BEGIN-1),X
        dbnzx  Clear_RAM

```

\*--- erase block -----\*

```

erase1:
        ldhx    #ERASEBLOCK                ; load block address to V_FLASH_ADDR
        sthx    V_FLASH_ADDR

        pshh                    ; "h" content will be corrupted in the jump
RAM routine
        jsr     FLASH_ERASE
        pulh

```

\*--- erase have been done -----\*

```

finish_all:
        bset    0,PORTA
        nop
        nop
        bclr   0,PORTA
        bra    finish_all

```

```

;=====
; Block Erase
;
;=====

```

```

FLASH_ERASE:

        bsr     BlkErase2RAM                ; copy block erase routine to RAM
        ldhx    V_FLASH_ADDR                ; load H:X with the block address
        lda     #(1<<b_ERASE)              ; MUST load Acc with b_ERASE
        jsr     Q_RAM_BlkJ_Erase           ; execute block erase in RAM
        rts

```

\*-----\*

```

BlkJ_Erase2RAM:
;       clrh
;       ldh     #BlkJ_Erase_Len            ; get blk erase routine length
;       ldhx    #BlkJ_Erase_Len            ; get blk erase routine length
;                                           ; NB: Assume "BlkJ_Erase_Len" is one byte long

BE2RAM1:
        lda     (Block_Erase-1),x          ; load from FLASH
        sta     (Q_RAM_BlkJ_Erase-1),x    ; copy to RAM
        dbnzx  BE2RAM1                    ; NB: Assume "BlkJ_Erase_Len" is one byte long
;                                           ; need modification if length over 1 byte
        rts

Block_Erase:
        sta     FLCR                        ; set ERASE bit
        sta     ,x                          ; write any data to block

```

## ROM-Resident Routines SE72-ROM

```

        bsr      Dly_5us

        lda      FLCR
        ora      #(1<<b_HVEN)
        sta      FLCR                ; set HVEN bit

* ----- *
        ldx      #20                ; (2)
Blk_Erase_Time:
        bsr      Dly_1ms            ; [14]
        dbnzx   Blk_Erase_Time     ; (3)

        ldhx    #FLCR
        lda      #%00001000        ; clear ERASE bit
        sta      ,x

        bsr      Dly_5us
        clr      ,x                ; clear HVEN bit
        rts

Blk_Erase_Exit:

* ----- *
;For 2.5MHz bus
Dly_5us:
        lda      #2                ; [4] cycles
        dbnza   $                  ; [2]
        rts                        ; [3]
                                     ; [4]

Dly_1ms:
        lda      #255              ; [4] cycles
        mov     PORTB,PORTB        ; [2]
        dbnza   $                  ; dummy for 5 bus clk
        rts                        ; [3]
                                     ; [4]

Dly_5us_Exit:
Blk_Erase_Len equ (Dly_5us_Exit - Block_Erase)

*****
* dummy interrupt service routine
*****
DUMMY_ISR:
        nop
        rti

*****
* IRQ interrupt service routine
*****
IRQ_ISR:
        bset    ACK1,INTSCR1      ; IRQ acknowledge
        rti

*--- block to be erased -----
        org     ERASEBLOCK

        dc.w   $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
        dc.w   $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000

```



```

dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000

```

```

dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
dc.w $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000

```

```

*****
*MOR register
*****
      ORG      MOR
dc.b  MORVALUE

```

## FLASH Memory Erasing and Programming

SE71-FLASH

To maintain data integrity in the FLASH memory, do not access FLASH memory locations that are not intended for program or erase once the high voltage enable bit is set (HVEN = 1). If the COP is enabled, a COP counter reset (write to \$FFFF) should be performed before setting the HVEN bit.

### FLASH Page Erase Operation

*Do not access any other FLASH locations between steps 4 to 8.  
Do not reset the COP counter between steps 4 to 8.*

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.

## Phase Lock Loop Stability SE70-PLL

2. Write any data to any FLASH location within the page address range desired.
3. Wait for a time,  $t_{nvs}$  (5  $\mu$ s).
4. Set the HVEN bit.
5. Wait for a time  $t_{erase}$  (20 ms).
6. Clear the ERASE bit.
7. Wait for a time,  $t_{nvh}$  (5  $\mu$ s).
8. Clear the HVEN bit.
9. After time,  $t_{rcv}$  (1  $\mu$ s), the memory can be accessed in read mode again.

## FLASH Program Operation

*Do not access any FLASH locations that are not intended for programming between steps 4 to 11. Do not reset the COP counter between steps 4 to 11.*

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Write any data to any FLASH location within the address range of the row to be programmed.
3. Wait for a time,  $t_{nvs}$  (5  $\mu$ s).
4. Set the HVEN bit.
5. Wait for a time,  $t_{pgs}$  (10  $\mu$ s).
6. Write data to the FLASH location to be programmed.
7. Wait for time,  $t_{prog}$  (20  $\mu$ s to 40  $\mu$ s).
8. Repeat steps 6 and 7 until all bytes within the row are programmed.
9. Clear the PGM bit.
10. Wait for time,  $t_{nvh}$  (5  $\mu$ s).
11. Clear the HVEN bit.
12. After time,  $t_{rcv}$  (1  $\mu$ s), the memory can be accessed in read mode again.

---

## Phase Lock Loop Stability

SE70-PLL

Do not use the PLL under these settings:  $AUTO = 1$ ; or  $AUTO = 0$  and  $\overline{ACQ} = 1$ .

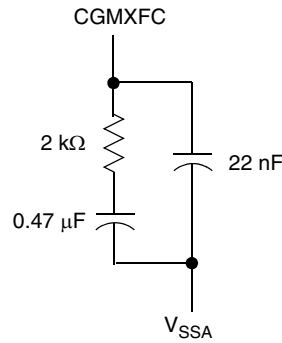
The clock generator module's PLL does not meet stability requirements when running in automatic bandwidth control mode or manual tracking mode. In these modes, the VCO output clock can become erratic, causing unpredictable MCU clocks, and resulting in possible MCU failure.

To overcome this problem, the PLL must be used in manual acquisition mode ( $AUTO = 0$  and  $\overline{ACQ} = 0$ ). As a result, there is no PLL lock detector, nor lock interrupt. The LOCK and PLLF bits are meaningless.

Follow this procedure for using the PLL:

1. Use the following external filter components:





2. Configure the PLL to the desired frequency.
3. Clear AUTO and  $\overline{ACQ}$  bits.
4. Set PLLON bit to turn on the PLL.
5. Wait 50ms for the PLL to lock.
6. Set BCS bit to select the VCO clock as the reference clock for the MCU.

### ADC Accuracy at Zero Volt Input

SE68-ADC

The 10-bit ADC conversion result for 0V input (zero input reading) is \$000, \$001, \$002, or \$003; not the specified \$00 or \$01. With inputs of 5mV and above, the ADC is within the specified accuracy of  $\pm 1.5$  LSB (see [Figure 1](#)).

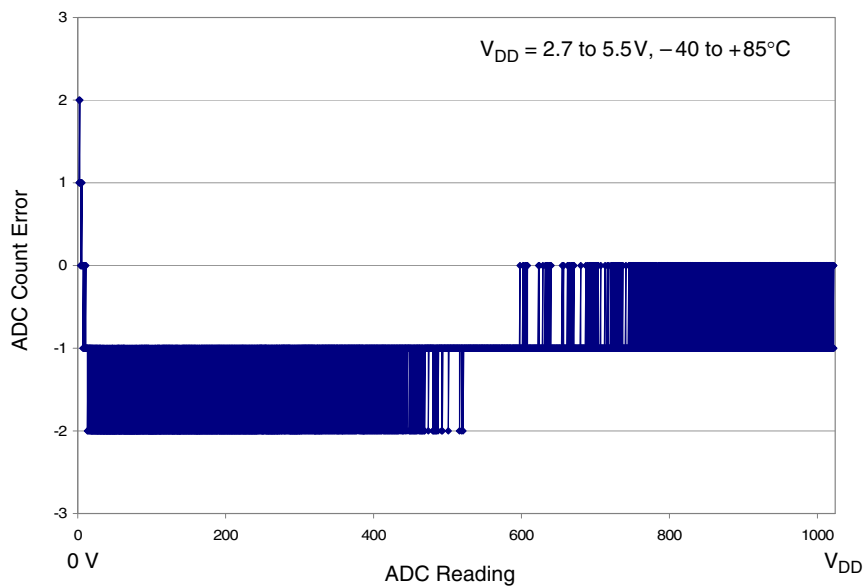


Figure 1. Typical ADC Error Count

## External Bypass Capacitor Connection on V<sub>REG</sub> Pin

SE69-VREG

Do not use a larger value than the recommended 100nF bypass capacitor connection between the V<sub>REG</sub> and V<sub>SS</sub> pins. Large capacitor values can cause the V<sub>REG</sub> voltage to fall too slowly during a power-down. The V<sub>REG</sub> voltage must be allowed to fall below 100mV before the MCU is powered up again. This allows the power-on reset circuit to rearm properly on power-up.

## Low-Voltage Inhibit Reset in Stop Mode

SE67-LVI

An internal reset occurs when entering stop mode if the LVI control bits are configured as shown in the table below:

CONFIG1 Register (\$001F)			
Bit 4	Bit 3	Bit 6	Bit 5
LVIPWRD	LVIREGD	LVISTOP	LVIRSTD
X	X	0	0
1	1	X	

X = don't care

LVIPWRD=1 is V<sub>DD</sub> LVI circuit disabled.

LVIREGD=1 is V<sub>REG</sub> LVI circuit disabled.

LVISTOP=0 is LVI disabled in stop mode.

LVIRSTD=0 is LVI resets enabled.

To enter stop mode with LVI disabled, set LVIRSTD=1 before entering stop mode. This will not cause an internal reset and also reduces the stop I<sub>DD</sub> to a minimum.