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

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

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

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.

**Workarounds for EE_WRITE and EE_READ**


**Workarounds for ERARNGE and MON_ERARNGE**

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

```assembly
*###################################
*# 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 addr 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_Blk_Erase ; execute block erase in RAM
  rts
```

*-------------------------------------------------------------
BlkErase2RAM:
```
; clrh
;  ldhx # Blk_Erase_Len ; get blk erase routine length

BE2RAM1:
```
  lda (Block_Erase-1),X ; load from FLASH
  sta (Q_RAM_Blk_Erase-1),x ; copy to RAM
  dbnzx BE2RAM1 ; NB: Assume "Blk_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
```
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:  ; [4] cycles
  lda  #2  ; [2]
dbnza  $  ; [3]
  rts  ; [4]

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

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

Mask Set Errata for Mask 0L47P, Rev. 0
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

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

Phase Lock Loop Stability

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

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 (\( \text{AUTO} = 0 \) and \( \text{ACQ} = 0 \)). 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 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

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](image-url)
External Bypass Capacitor Connection on \( V_{\text{REG}} \) Pin

Do not use a larger value than the recommended 100nF bypass capacitor connection between the \( V_{\text{REG}} \) and \( V_{\text{SS}} \) pins. Large capacitor values can cause the \( V_{\text{REG}} \) voltage to fall too slowly during a power-down. The \( V_{\text{REG}} \) 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

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

<table>
<thead>
<tr>
<th>CONFIG1 Register ($001F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4</td>
</tr>
<tr>
<td>LVIPWRD</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

X = don’t care  
LVIPWRD = 1 is \( V_{\text{DD}} \) LVI circuit disabled.  
LVIREGD = 1 is \( V_{\text{REG}} \) 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 \( I_{\text{DD}} \) to a minimum.