Mask Set Errata 3
MC68HC912B32 Microcontroller Unit

INTRODUCTION

This errata provides information applicable to the following MCU mask set devices:

- 0J64Y

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a four-character code consisting of a letter, two numerical digits, and a letter, for example F74B. Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code, for example 0F74B.

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, PC, ZC or XC prefix. An SC, PC or ZC 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.

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

MSE Published Date: March 1, 2001

ERRATA SUMMARY

Errata Number	Module affected	Brief description	Workaround available?	First Issued
AR569	ESD	Device marginal after ESD zapping	Yes	Rev 1
AR311	ATD	Internal reference conversion result undetermined	Yes	Rev 1
AR493	BDLC	Reception of invalid symbol clears BDR	Yes	Rev 1
AR519	BDLC	CRC error in received message does not prevent IFR transmission	Yes	Rev 1
AR520	BDLC	Transmission of IFR byte following EOD of current message	Yes	Rev 1
AR504	WCR	No crystal start-up delay after clock monitor reset	Yes	Rev 2
AR564	CGM	STOP cannot be exited when DLY=1	Yes	Rev 1
AR443	CGM	On reset out of STOP, MCU may restart before crystal is stable	Yes	Rev 1
AR528	INT	With edge-sensitive IRQ, pin must be released before exiting STOP	Yes	Rev 1
AR600	INT	Device can get stuck in WAIT depending on IRQ/XIRQ deassertion	Yes	Rev 1
AR527	INT	SWI can be fetched if interrupt source removed without setting I bit	Yes	Rev 1
AR510	IOB	Current pin leakage in expanded mode can cause inaccurate A/D conversions	Yes	Rev 1

ESD: AR569

Device is marginal after 150V machine model and 1.5KV human body model ESD zapping.

Work-around Extra care must be taken when handling this device.

ATD: AR311

The (VRH–VRL)/2 internal reference conversion result may be $7F, $80 or $81.

Work-around If the (VRH–VRL)/2 internal reference is used (perhaps for system diagnostics), expected pass result may be $7F, $80 or $81.

BDLC: AR493

To transmit a message using the BDLC, the user writes the first byte of the message to be transmitted into the BDLC data register (BDR). This will initiate the transmission process at the beginning of the next idle bus state.An invalid symbol being received by the BDLC clears any byte that had been previously written to the BDR. This will inhibit the transmission process until the user writes another byte to the BDR. The following scenario describes an event sequence that would prevent the user from knowing that an invalid symbol was received and that the BDR had been cleared.

Work-around

A two level strategy has been developed that positively signals the need to restart the transmission of a message. The first level looks for the special case of reception of an illegal symbol with a byte pending transmission in the BDLC data register, as described above. The second level uses a transmit watchdog timer to spot any case of a transmission not occurring within a maximum amount of time.1a) If an illegal symbol interrupt occurs with a byte pending transmission in the BDR, reload the BDR with the first byte of that message to restart transmission.

BDLC: AR519

CRC error in received message does not prevent IFR transmission.

Work-around

In response to the CRC Error interrupt ($18 in BSVR), immediately write an $FF byte into the BDR, and then clear the lower four bits in BCR2 (TSIFR, TMIFR0, TMIFR1, TEOD). This will cancel the IFR transmission.

BDLC: AR520

When programmed to transmit a single byte Type 2 IFR (byte loaded in the BDR and TSIFR bit set in BCR2), the BDLC will attempt to transmit a single IFR byte following the EOD of the message currently being received. If the byte to be transmitted loses arbitration, the BDLC will continue to retry transmission until it is successful or an error occurs or the TEOD bit is set.

Work-around

(Short form) When the first RXIFR interrupt occurs the requester message has been received without errors (invalid symbol or CRC). When the BDLC receives back correctly the IFR byte is was trying to transmit, then response by this node is complete and the requester message received can be passed to the application layer. Ignore any invalid symbol error that occurs after the first RXIFR interrupt, since transmission of IFRs are not retried.

CGM: CRYSTAL START-UP WITHOUT DELAY AT CLOCK MONITOR RESET AR504

When a clock monitor reset occurs, the crystal may start-up with no delay period. As a result, the MCU attempts to fetch reset vectors before the crystal has fully stabilized.

Work-around

When clock monitor failure has occurred, hold the reset signal until the crystal has reached stable oscillation.

In some applications the clock monitor reset is used to detect an accidental crystal clock loss. An alternative solution is to engage the PLL and enable the limp-home mode. As the limp-home mode is enabled, the clock monitor will not reset the MCU (NOLHM =0 in PLLCR). In case of clock loss, the limp-home mode will be engaged.

If the crystal clock is restored, the PLL will be automatically re-engaged. As the transition from PLL to limp-home and vice versa is smooth, the CPU will continue to run the code even if the crystal oscillation amplitude is not fully stabilized.

The LHIF flag in PLLFLG is set when MCU enters or exits the limp-home condition.

CGM: AR564

STOP mode cannot be exited using interrupts when DLY=1 depending on where the Real-Time-Interrupt (RTI) counter is when the STOP instruction is executed. The RTI counter is free-running during normal operation and is only reset at the beginning of Reset, during Power-on-Reset, and after entry into STOP. The free-running counter will generate a one cycle pulse every 4096 cycles. If that pulse occurs at the exact same time that the stop signal from the CPU is asserted then the OSC is stopped but the internal stop signal will remain low. In this state the OSC is shut off until RESET.

Work-around

1. If you are not using the Real Time Interrupt function you can wait for a RTI flag before entering into STOP to guarantee the counter is in a safe state. When executing the following code all interrupt sources except for those used to exit STOP mode must be masked to prevent a loss of synchronization.
   A loss of synchronization can occur if an interrupt is processed between the setting of the RTIF and the execution of the STOP instruction. Also, you must enable the RTI counter in the initialization code, set to the fastest RTI time-out period, and the RTIE bit should NOT be set.

```
          BRCLR   RTIFLG,#RTIF,RTIFClr   ; RTIF flag is already clear
          LDAB    #RTIF                  ; if it's set, clear the flag.
          STAB    RTIFLG
RTIFClr:  BRCLR   RTIFLG,#RTIF,*         ; wait until the RTIFLG is set.
          NOP
          NOP
          NOP
          STOP                           ; enter stop mode
```

CGM: AR443

When coming out of STOP by using reset, the crystal start-up delay time-out does not occur. This means the MCU may be attempting to run before the crystal has become stable.

Work-around When coming out of STOP with reset hold the reset signal until the crystal has reached stable oscillation.

INT: AR600

The device can get trapped in WAIT mode if, on exiting the WAIT instruction, the deassertion timing of the XIRQ or level-sensitive IRQ occurs within a particular timeframe. Only reset will allow recovery. Noise/bounce on the pins could also cause this problem.

Work-around

1. Use edge-triggered IRQ (IRQE=1) instead of XIRQ or level-triggered IRQ.
2. Use RTI, timer interrupts, KWU or other interrupts (except level-sensitive IRQ or XIRQ) to exit WAIT. If using RTI, it must be enabled in WAIT (RSWAI=0) and the COP must be disabled (CME=0).
3. Assert XIRQ or level-sensitive IRQ until the interrupt subroutine is entered.
4. Add de-bouncing logic to prevent inadvertent highs when exiting WAIT.

INT: AR528

When using an edge sensitive IRQ signal to trigger an interrupt service routine and the IRQ is not released during servicing, the part will not enter stop mode if the following executed instruction is STOP.

Work-around When IRQ is set to be edge sensitive, release pin before executing STOP instruction.

INT: AR527

An SWI can be fetched if the source of an interrupt is taken away by disabling the interrupt without setting the I mask bit in CCR.

Work-around Before disabling an interrupt, set the I mask bit in CCR.

IOB: AR510

Expanded mode operation causes an increase in bus driver shoot through current pin leakage which can cause inaccurate A/D conversions.

Work-around: Enabling reduced drive on ports A, B, and E while the HC12 is in expanded mode reduces the amount of VDD/VSS noise caused by bus driver shoot through current. Therefore, the A/D accuracy meets specified limits.

Additional mask set errata can be found on the World Wide Web at http://www.mcu.motsps.com/documentation/index.html