



## *Mask Set Errata 1*

# **M68HC912D60 Microcontroller Unit**

## **INTRODUCTION**

---

---

This mask-set errata provides information pertaining to the following 68HC912D60 MCU mask set devices:

- 1F68K

Some items do not report bugs but only contain customer information.

## **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.*

Specifications and information herein are subject to change without notice.



## **INT: WAIT CANNOT BE EXITED IF XIRQ/IRQ LEVEL DEASSERTION OCCURS WITHIN PARTICULAR WINDOW OF TIME 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.

## **MCU: STOP IDD GREATER THAN THE EXPECTED SPECIFICATION AR489**

---

The present STOP  $I_{DD}$  test limit is set to a higher limit on the current mask set which is greater than the expected specification limit of:

10 $\mu$ A	-40C to 85°C
25 $\mu$ A	-40C to 105°C
50 $\mu$ A	-40C to 125°C

### **Work-around**

None.

## **BKP: ADDRESS AND DATA REGISTERS RESET ONLY ON POR AR463**

---

Breakpoint Address and Data registers are properly reset only upon Power on Reset.

### **Work-around**

To insure BRKAH, BRKAL, BRKDH and BRKDL registers have the correct default values, always clear each immediately after reset.

## **INTERRUPTS: ARE NOT MASKED DURING 1ST CYCLE OF ANY BACKGROUND INSTRUCTION**

**AR441**

There is a cycle at the beginning of executing a BDM instruction that is susceptible to being interrupted directly after the background has executed. Since the interrupt source is masked, the interrupt vector that is requested is \$FFF6. The BDM firmware at \$FFF6 points to the routine at \$FF24. The interrupt was stacked when it was taken, but the BDM routines do not un-stack (no RTI). When returning from BDM, the stack pointer is pointing to the wrong place.

Avoid using TRACE during cycles that allow interrupts to become unmasked (i.e. TRACE of a CLI if interrupts are pending). Caution should also be taken when using BGND and TAG following an instruction that clears the X or I mask bits.

**Work-around**      None.

## **MSCAN: ADDITIONAL TX BIT WHEN GOING BUS-OFF**

**AR435**

When a CAN transmission takes place and an error occurs, one extra dominant bit will be sent when going off bus. The problem will be corrected in a future revision.

**Work-around**      None

## **MSCAN: RECOVERY FROM SLEEP MODE**

**AR450**

When entering sleep mode, RXF is not changed, however the foreground/background pointer is reset, resulting in buffer Rx1 in the foreground readable by the CPU. If there was one message in Rx0 (and none in Rx1) the pointer will now point to the wrong buffer with undefined contents.

**Work-around**      The receive buffers must be cleared by the CPU before entering sleep mode.

## **MSCAN: TERRIF IS SET AT END OF BUS-OFF**

**AR451**

When the msCAN is in bus-off state and has counted 128 sequences of 11 recessive bits, the TERRIF flag is set.

**Work-around**      Always clear TERRIF together with BOFFIF.

## MSCAN: INCORRECT RX MESSAGE IN OVERRUN OR 'ALMOST' OVERRUN

### AR434

---

---

It is possible that under a particular set of circumstances, the msCAN receive buffer will contain a corrupted message. Under normal operation, it is expected that the required circumstances for the error will occur infrequently.

The msCAN background receive buffer will contain a corrupt message after either of the following sequence of events have occurred:

#### A. Overrun Condition

##### A.1

Initial state: Both receive buffers (foreground and background) are filled by accepted messages. They have not yet been released by the software.

##### A.2

Another message is received, triggering the overrun condition (OVRIF = 1).

##### A.3

The software releases one (or both) of the receive buffers by clearing RXF once (or twice) during the reception of a message (either the message which triggers the overrun condition or a subsequent message).

#### B. "Almost" Overrun Condition

##### B.1

Initial state: Both receive buffers (foreground and background) are filled by accepted messages. They have not yet been released by the software.

##### B.2

The msCAN begins to transmit a message M.

##### B.3

Software releases one (or both) of the receive buffers by clearing RXF once (or twice) during the transmission of the arbitration field of message M.

##### B.4

The msCAN loses arbitration on the CAN bus while transmitting M.

If either of the above sequences occur, the next bits seen on the bus after clearing RxF will be interpreted as the beginning of the ID field and will be transferred into the background receive buffer. The background buffer will then contain a shifted image of the true message. This message will be brought into the foreground buffer when the foreground buffer is next released ONLY if the shifted identifier happens to pass the acceptance filter configuration programmed into the msCAN.

The erratum is caused by a functional error in the msCAN design implementation.

## Work-around

To avoid the error conditions, the msCAN receive driver software must process the incoming stream of messages fast enough to prevent both receive buffers becoming filled with accepted messages. This may be achieved by optimizing the interrupt service routines with higher priority than the msCAN receive interrupt. Alternatively, the msCAN receive and transmit driver software may be modified to detect the circumstances under which the error occurs.

---

## ELECTROSTATIC DISCHARGE (ESD) PERFORMANCE

---

The 1F68K 68HC912D60 failed ESD testing to the AEC Human Body Model at voltages above 500V.

---

## CPU: ETBL INSTRUCTION DEPENDENT ON THE INITIAL C-BIT VALUE

---

**AR287**

### ETBL Instruction Dependent on the Initial C-bit Value

The ETBL instruction will provide an incorrect result under either of the following 2 conditions:

1.  $Y2 < Y1$  and C-bit=0 before ETBL instruction executed.
2.  $Y2 > Y1$  and C-bit=1 before ETBL instruction executed.

except for the cases where  $(B \times (Y2 - Y1)) = 0$  (i.e.  $B=0$  or  $Y2=Y1$ )

## Work-around

If a borrow is needed from the equation, then set C-bit=1 before executing the ETBL instruction; if a borrow is not needed than clear C-bit=0 before executing the ETBL instruction. Example:

LDxy (idx) ; initialize index register to point to the start point

LDD (Y2) ; get value of Y2 into ACCD

CPD (Y1) ; D-M,  $Y2-Y1$ , This updates C-bit

LDAB (B) ; accumulator B initialized with ratio

ETBL (idx) ; Perform instruction

---



---

## CPU: REV AND REVW MAY GIVE INCORRECT RESULTS IF INTERRUPTED AR288

---



---

If the REV or REVW instructions are interrupted while processing a rules list, the results from the instructions may be incorrect. The Condition Code Register and Index Register Y (weight pointer for REVW) may be incorrect when the stacking occurs for the interrupt. The REV and REVW instructions produce the wrong result after returning from the interrupt because the V-bit in the CC register and IY registers (REWW) may not hold the same state as prior to the interrupt.

**Work-around**      Disable the interrupts prior to using the REV or REVW instruction (which could increase the interrupt latency). If a non-maskable interrupt has been enabled, there is no workaround.

---



---

## ATD: CONVERSION OF THE (VRH-VRL)/2 INTERNAL REF VOLTAGE RETURNS \$7F NOT \$80 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.

---



---

## ATD: STATUS BIT NOT PROPERLY RESET WHEN STARTING A NEW CONVERSION SEQUENCE AR373

---



---

ATD status bits and conversion counter are not reset properly if a new conversion sequence is started while an active A/D conversion sequence is in the process of completion.

**Work-around**      When starting a new sequence, perform two writes to control registers 4/5 in quick succession. If the first write occurs when the status bit/conversion counter is not reset, the second write will correct ATD operation.

---



---

## FLASH: AT HIGH TEMPERATURE AR469

---



---

Flash cannot be programmed reliably at high temperatures.

**Work-around**      To enhance Flash programming reliability at elevated temperatures, the Flash bulk erase pulse time ( $T_{\text{epulse}}$ ) should be reduced from 90-110ms to 5-10ms.

---



---

## FLASH: REQUIRED 250 NSEC STOP WAKEUP RECOVERY TIME AR458

---



---

Flash requires 250 nsec delay for wakeup from STOP mode. If the operating bus frequency is greater than 4MHz, the Flash can not be used when recovering from STOP mode when the DLY bit is equal to '0'.

**Work-around** In stop mode set the DLY bit equal to '1', or with DLY equal to '0' it is possible to map the EEPROM module over the Flash module in the memory map and place the interrupt vectors in the EEPROM array.

---



---

## FLASH: REQUIRED 250 NSEC WAIT MODE WAKEUP RECOVERY TIME

---



---

Flash requires 250 nsec delay for wakeup from wait mode with FEESWAI=1. If the operating bus frequency is greater than 4MHz, the Flash can not be used when recovering from WAIT mode when the FEESWAI bit is equal to '1'.

**Work-around** If interrupt vectors are located in the Flash array, do not set the FEESWAI bit in Wait mode

---



---

## FLASH: DC CURRENT ON VFP PIN CAUSES THE PART TO LATCH UP

---



---

Latch up is produced when DC current of 100 mA or more is applied to Vfp pin (below  $V_{SS}$ ). All pins should tolerate 200 mA as a minimum.

**Work-around** Avoid applying a DC current of 100 mA or more (below  $V_{SS}$ ).

---



---

## PWM: USING CLOCK S0 OR S1 AT HIGH TEMPERATURE & VDD CAUSES INCORRECT WAVEFORMS AR241

---



---

Race conditions may occur in the logics generating CLOCK S0 and S1. Using these clocks can cause incorrect waveforms. The race condition gets worse at high temperature and high Vdd.

**Work-around** Use only CLOCK A and CLOCK B for driving the PWM channels by setting control bits PCLK3, 2, 1, 0 to '0'.

The draw back is the loss of the extra clock resolutions that are generated by the CLOCK S0 and S1.

This workaround only provides clock options of: E clock, E/2, E/4, E/8, E/16, E/32, E/64, E/128.

## **PWM: 16-BIT PWM OUTPUT WORKS ONLY IF ENABLE BOTH PWM BITS**

**AR243**

16-bit concatenated PWM output works only if both PWM bits are enabled. For example in order to use a 16-bit PWM output on PP0(2), both the PWEN0(2) and PWEN1(3) should be set. As PWEN1(3) is set, the GP I/O capability is lost on PP1(3).

### **Work-around**

To use 16-bit concatenated PWM, set both PWENx bits of the desired pair to '1'. There is no workaround for the loss of GP I/O on the low order channel.

## **PWM: WAVEFORM POLARITY 'FLIP' IN CENTER MODE**

**AR315**

Output waveform polarity may 'flip' when writing PWDTY register to \$00 (or \$0000 in 16-bit configuration) in CENTER mode. Problem occurs when writing PWDTY register with associated channel enabled or disabled.

### **Work-around**

Program PWDTY=\$FF (or \$FFFF) instead of PWDTY=\$00 (or \$0000) to achieve 'zero-duty'.

## **PWM: PIN SYNCHRONIZATION AFTER PWENX ASSERTION IS NOT ACCORDING TO SPECIFICATION**

**AR331**

The PWM channel output becomes active on the PWM pin immediately after assertion of the associated PWENx bit. As a result, the start of the first PWM period may be truncated by up to one channel source clock period.

This activity is not according to the HC12 PWM documentation which states 'There is an edge-synchronizing gate circuit to guarantee that the clock will only be enabled or disabled at an edge.'

## **PWM: BOUNDARY CASES IN LEFT-ALIGNED MODE IS NOT ACCORDING TO SPECIFICATION**

**AR332**

When PWDTY = \$FF and PWPERx = \$00, the pin value is opposite the value in the POLx bit.

This activity is not according to the HC12 PWM documentation which states 'If PWPERx = \$00 & POLx = 0, then the output is always low. If PWPERx = \$00 & POLx = 1, then the output is always high.'

### **Work-around**

When programming PWPERx = \$00 to get a boundary case, program PWDTYx

## PWM: PWM REGISTER WRITE FOLLOWED BY IMMEDIATE PWCNT READ CAUSE FALSE RESET AR358

---

A write to any PWM register followed immediately by a read of a PWM counter register (PWCNTx) may cause a false counter reset.

**NOTE:** *If the PWM register written is a PWM counter register the counter may reset after both the write and read accesses.*

**Work-around**      Add NOP instruction between the write and the counter-read operations.

## PWM: BOUNDARY CASES IN CENTER MODE ARE NOT ACCORDING TO SPECIFICATION AR359

---

PWM boundary cases in Center Mode are not according to specification

1. When PWDTYx is greater than PWPERx, the pin is stuck in the opposite value of the specification (i.e. the state opposite the PPOLx value).
2. When PWDTYx = PWPERx, the pin will change value when the counter reaches the new duty value.
3. When the PWM channel is enabled, and the period and duty registers are changed such that PWPERx = \$00 and the new PWDTYx value is between zero and the old PWPERx value, the pin will change to the opposite value of the specification.

**Work-around**

1. If PWDTYx value is greater than the PWPERx value, change the PPOL bit.
2. No workaround available.
3. Update the PWDTYx and PWPERx registers when the associated channel is disabled; or for the boundary case to be correct when PWPERx = \$00, PWDTYx should also be equal to \$00.

---



---

## **PWM: WRITE TO PWM COUNTER MAY RESULT IN OPPOSITE POLARITY ON PIN** **AR360**

---



---

Write to PWM counter under the following conditions may result in opposite polarity on pin:

1. Value of counter is greater than the value of duty register.
2. The PWM channel is enabled.

**Work-around**      Only write PWM counter when associated channel is disabled (PWENx=0)

---



---

## **PWM: PWM PIN MAY HAVE INCORRECT VALUE WHEN ENABLED/DISABLED** **AR363**

---



---

In center or left aligned modes, when operating in 8-bit or 16-bit configurations, the PWM pins associated with channels 1 and 3 may have an incorrect value when enabled or after disable/enable sequence. If DTY register is written when the channel is enabled, the new duty value will not transferred from buffer to register when the channel is disabled.

**Work-around**      While channels 1 or 3 are disabled, write to the associated PWPER and PWDY register.

---



---

## **WCR: RTI INTERRUPT GETS WRONG VECTOR** **AR323**

---



---

When an I-type interrupt occurs at the same time as an RTI interrupt, the program address at \$FFC0 will be fetched.

**Work-around**      Point the \$FFC0 vector to a return from interrupt (RTI) instruction. This will get the program returned to start processing the proper interrupt as quickly as possible.

---



---

## **WCR: WRITE TO RTIFLG REGISTER TO CLEAR RTIF BIT DOES NOT ALWAYS CLEAR THE BIT** **AR369**

---



---

Clearing the RTI flag while the MCLK is running significantly slower than the PCLK may not clear the bit in the current timer cycle.

**Work-around**      It takes at least 11 cycles to get into an interrupt service routine. If the effective MCLK value is greater than divide-by-8 off of PCLK, then the user must account for the number of cycles it takes to toggle the MCLK signal before clearing the RTIF flag and exiting the interrupt service routine.

---



---

**WCR: CLOCK MONITOR DISABLED DURING STOP**
**AR375**

The clock monitor feature is disabled during STOP due to the addition of the limp-home mode feature. The clock monitor will operate properly in other operating modes.

---



---

**WCR: CME CONTROL BIT DOES NOT REFLECT STATE OF CLOCK MONITOR AFTER FAILURE**
**AR382**

If a clock monitor is enabled with the RSTE bit set to 1 and a clock failure occurs, the state of the CME control bit, after the reset, does not reflect the actual state of the clock monitor. The clock monitor is enabled for 4096 cycles after the reset. If another clock failure occurs within 4096 cycles, another clock monitor action will be taken.

**Work-around**

Connecting the Vddpll to Vdd enables the PLL and the ability to use the limp home mode of the device. Set the RSTE bit to 0 to enable limp home mode action when a clock monitor failure occurs.

---



---

**WCR: WRITE TO CR[2:0] IN COPCTL CAN CAUSE COP TIMER TO FREEZE**
**AR420**

When changing the value of the CR[2:0] bits in the COPCTL register, the COP timer may freeze. This problem can only occur when decreasing the value of the bits, i.e. going to a faster time-out rate.

**Work-around**

Immediately after changing the value of the CR[2:0] bits, reset the COP timer by writing \$55 followed by \$AA to the COPRST register. This will ensure that the COP is working properly.

---



---

**WCR: CLOCK MONITOR RESET WITH RSTE (NOLHM) SET EXITS IN LIMP HOME MODE**
**AR437**

When a clock monitor reset occurs with the RSTE (NOLHM) bit set, the part exits reset running in limp home mode. The RSTE (NOLHM) bit is bit 0 of the PLLCR register located at \$003C. A loss of the reference clock causes a clock monitor reset.

**Work-around**

None

---



---

## **MEBI: DURING THE DELAY COMING OUT OF STOP, E CLOCK IS OUTPUT**

### **AR251**

---



---

When the part is in the delay phase of coming out of STOP, the E clock will be running external.

Since the oscillator is just starting up, the E clock is likely to be not 'clean' during beginning of this time.

---



---

## **MEBI: PIPE SIGNALS TO PINS PE[6:5] NOT CORRECT DURING BDMFRZ**

### **AR276**

---



---

The multiplexed IPIPE[1:0] signals which are available on PE[6:5] pins, do not convey the correct information for reconstruction of the internal instruction queue during a BDMFRZ.

BDMFRZ occurs when a BDM access is requested and either no free cycles are found within 128 cycles or the requested access cannot be completed in one cycle (i.e.: mis-aligned word access).

#### **Work-around**

When using a logic analyzer, be aware that a BDMFRZ operation will not be properly identified by the pipe status bits even though the operation will function properly.

---



---

## **MEBI: IN NARROW EXPANDED MODES, ADDRESSES ON PORT B ARE NOT HELD**

### **AR313**

---



---

In narrow expanded modes, addresses on Port B are not held past the multiplexed address hold time. Port B switches to high impedance input during the data cycle and is not held throughout the cycle as on the HC11.

#### **Work-around**

A 16 bit address latch is required in all expanded modes.

---



---

## **CGM: MORE FLEXIBLE SLOW MODE DIVIDER REGISTER**

### **AR334**

---



---

The slow divide counter ratios are currently: 2, 8, 12, 16, ..., 252, 256.

On future revisions of this module the divide ratios will be changed to 2, 4, 8, 12, 16, ..., 252 (in increments of 4).

---



---

**CGM: ABNORMAL CURRENT CONSUMPTION OF PLL**
**AR366**

When VDD is applied on VDDPLL, in wait or stop modes, the VDDPLL pin can drain up to 1mA @ 5v.

**Work-around**

Ground VDDPLL during wait or stop modes for lower current consumption.

---



---

**CGM: OSCILLATOR NOT FUNCTIONING PROPERLY WITH LOW FREQUENCY CLOCK SOURCE (<1MHZ)**
**AR373**

When low frequency crystals or resonators (<1MHz) are used as the clock source, the E clock can have the same frequency as the crystal or resonator.

**Work-around**

Do not use low frequency clock sources.

---



---

**CGM: JITTERY BUS CLOCK WHEN PLL ENGAGED**
**AR452**

PLL synthesized bus frequency may not be accurate enough for some applications due to excessive jitter.

**Work-around**

Do not use PLL for applications where stability and/or accuracy is required.

---



---

**CGM: STOP DOES NOT TAKE TIME-OUT WHEN RESET CAUSES THE RELEASE**
**AR431**

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.

## **CGM: CPU STOPS GOING INTO WAIT BEFORE STRETCHED CYCLE IS FINISHED** **AR475**

---

When the device exits WAIT mode, it does not return to the correct location within the routine if the stack is positioned in external memory and if the stretch bits have been enabled to lengthen the clock.

**Work-around** To overcome this problem, locate the stack in internal RAM resources and/or clear the stretch bits to prevent clock stretching.

## **CGM: CPU CLOCKS RUNNING OFF DUE TO UNSTABLE CLOCK WITH DLY=1** **AR353**

---

When the clock source (i.e. crystal) has been stopped for longer than 300 $\mu$ s there is high probability that an unstable clock will disrupt the internal CPU tclocks. With a free running clock source (i.e. oscillator pack), asserting the wake-up signal (i.e. XIRQ, IRQ) just prior to the falling edge of an external clock source can cause the part to not exit STOP mode.

**Work-around** Crystal: no workaround  
 Oscillator pack: Synchronize wake-up signal to the rising edge of external clock input.

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

---

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.

## CGM: OSCILLATOR START-UP WITH 8MHZ AND 16MHZ CRYSTALS AND RESONATORS AR536

---

It is possible that oscillator start up will fail with high frequency crystals and resonators under specific environmental conditions where moisture forms on the pins of the MCU, creating a leakage path to Vss or Vdd.

Typical conditions for failure are the following: 8Mhz or higher frequency crystal or resonator and leakage on the EXTAL pin to Vss corresponding to an impedance of 8M ohms.

No failures have been observed with moisture affecting oscillator start-up with resonators or crystals at or below 4Mhz frequency.

Once started, the oscillation is not affected by moisture. The issue is only related to oscillator start-up.

### Work-around

Conformal coating (water repellent or sealant) is required across the oscillator components and oscillator pins of the MCU where there is a possibility of the leakage paths described above.

The most critical area is around the EXTAL pin of the MCU where it is more likely to have a small spacing between EXTAL exposed tracks and low voltage sources. In particular the EXTAL and RESET pins are next to each other. At Power-on, the RESET pin is held to Vss in most systems and may be the most likely leakage path if condensation forms on the MCU pins.

## CGM: CANNOT INTERRUPT OUT OF STOP WITH DLY=1 AR565

---

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.

```

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

2. If you are driving a clock in (not using the OSC) then you could set DLY=0.
3. Pseudo-STOP and DLY=0 could be used. The oscillator will be kept alive during STOP at the expense of power consumption but no recovery delay is needed. Set the PSTP bit and clear DLY bit prior to going to STOP.
4. Limp Home and DLY=0 could be used. The part comes out of STOP in limp home mode while the crystal recovers. If a known frequency is not a requirement, this workaround avoids having the crystal alive in STOP mode. Clear the NOLHM and DLY bits prior to going to STOP.

## **BDM: CLKSW IS CLEARED BY FIRMWARE**

---

This is primarily an issue only for developers of BDM interface equipment (BDM pods). The BDM logic was changed to allow switching between XTAL/2 and a possibly faster bus rate clock. If you set the CLKSW bit (faster bus rate clock) in the BDM STATUS register (with a WRITE\_BD\_BYTE @ FF01 command, and then later send a GO, TRACE1, or TAG\_GO command (or encounter a \$00 opcode with ENBDM=0), the CLKSW bit is cleared by BDM firmware such that the BDM speed switches back to the default (XTAL/2) rate.

### **Work-around**

When communicating at the bus rate (CLKSW=1), issue a new WRITE\_BD\_BYTE command (at the XTAL/2 rate) to set the CLKSW bit in BDM\_STATUS after any GO, TRACE1, or TAG\_GO command or if BDM communications fail unexpectedly.

## **BDM: LOSS OF BDM COMMUNICATION WITH PLL SELECTED AS SYSCLK SOURCE AND EXECUTING LONG INSTRUCTIONS (11 CYCLES OR MORE: IDIV, FDIV, EMACS, EDIVS, IDIVS)**

---

It is possible to lose BDM communication when executing long instructions, 11 cycles or more, if the PLL is being used as the SYSCLK source.

### **Work-around**

Do not use the PLL as SYSCLK source when using the BDM interface to debug code that uses instructions taking  $\geq 11$  cycles, or insert a breakpoint before such an instruction and single step over it before continuing code execution.

## **BDM: FAILURE TO RELEASE ADDRESS BUS AFTER MEMORY ACCESS**

---

When the BDM module is using XTAL/2 as its reference clock and the PLL is providing the clock for the CPU buses, a BDM logic circuit can fail to release control of the address bus after completing a memory access. When the next BDM serial command is completed, the BDM gives up control of the address bus, but by this time the CPU is already lost. This often results in the CPU eventually reaching a \$00 opcode and getting into active BDM mode.

### **Work-around**

The error can be avoided if the BDM operates from the same clock source as the bus. Everything works after reset because both the BDM and the bus use XTAL/2 as the clock source. If the CLKSW bit is changed to one before engaging the PLL, the system also works (although the host must change communication speed to match the bus frequency changes).

In some systems the PLL is turned on and off and the bus frequency can be changed at random intervals and there is no practical way for a host to track these changes without access to the E-clock frequency. In these systems there is no workaround.

## **KWU: INFORMATION ON 80QFP VERSION**

---

In 80QFP version, the PGUPD and PHUPD are connected internally to VDD and VSS.

The PG4 will have a pull-up. The PH4 will have a pull-down. These devices can be enabled or disabled by the related control bit in the PUCR register.

---



---

## INT: EDGE SENSITIVE IRQ DOES NOT WORK CORRECTLY DURING STOP 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: DISABLING INTERRUPT WITH I MASK BIT CLEAR CAN CAUSE SWI AR527

---



---

If the source of an interrupt is taken away by disabling the interrupt without setting the I mask bit in the CCR, an SWI interrupt may be fetched instead of the vector for the interrupt source that was disabled.

**Work-around**      Before disabling an interrupt using a local interrupt control bit, set the I mask bit in the CCR.

---



---

## KWU: FILTER MAY NOT PREVENT PULSES SHORTER THAN 2 $\mu$ S FROM WAKING THE PART FROM STOP AR578

---



---

The key wake up filter may not prevent pulses shorter than 2 $\mu$ s from waking the part from STOP.

---



---

## PLL: LIMP HOME MODE AR627

---



---

The device can prematurely indicate that the oscillator has stabilized releasing the part from Limp Home clock mode to the oscillator clock mode with an unstable oscillator. This can cause unpredictable behavior of the MCU. This situation can arise with short external power-on reset periods and / or crystal oscillator circuits that exhibit slow startup characteristics. If the PLL is not being used (Vddpll connected to Vss) Limp Home mode is disabled and this issue does not apply.

All customers should review any applications based on the referenced devices. If the crystal clock is stabilized before the external RESET line is released and the customer is not using stop mode (pseudo-stop is not affected) then there is no problem. If the clock is not stable when external RESET is released then they should contact Motorola for consultation.

Common practice for the start up mode of operation of HC12 microcontrollers is for the external RESET line to be held active until such time as the crystal has stabilized at its operating frequency. On release of the external RESET line and

when the WCR (counter register) reaches a count of 4096 cycles, normal operating mode is entered with the CPU clocked from the crystal frequency (see fig. 1).

The HC12 mode of operation known as Limp-Home Mode (LHM) is enabled when the VDDPLL pin is at VDD and is entered if for any reason the external crystal ceases to oscillate. During this mode the CPU will be clocked from the free-running VCO clock of the PLL (at a nominal frequency of 1MHz). If LHM is enabled during the start-up phase (i.e. VDDPLL=5V, NOLHM bit=0) and the external RESET line is not held active until after the crystal frequency is stable then the device starts up in LHM since no crystal oscillations will be detected. This situation can arise with short reset periods and/or crystals that exhibit slow start-up characteristics.

For the first 4096 cycles i.e. during the internal reset period, Limp Home mode will be de-asserted if oscillator activity is detected by the clock monitor circuit -due to the asserted Reset signal there can be no CPU activity during the Reset phase. Following release of the external or internal POR RESET in LHM (which ever is later) the crystal oscillator is sampled by the clock monitor circuit after another 4096 VCO clock cycles and at intervals of 8192 clock cycles thereafter until the crystal is deemed to be operating. If the crystal oscillator is showing activity at the time it is checked then it will be deemed to be good, even though it may not have fully stabilized, and LHM will be de-asserted. This can cause the device to switch from LH mode to normal mode with the CPU clocked from an unstable signal from the crystal oscillator (see fig. 2) resulting in unpredictable function of the CPU.

The COP Reset doesn't exhibit this behavior as, although the same reset sequence is followed, the oscillator isn't stopped.

When exiting Stop mode (DLY=1) a similar 4096 cycle delay is executed and therefore this behavior could also show up at this time. In applications where this is likely to be an issue, using pseudo-stop is recommended as an alternative. Current draw will increase <100  $\mu$ A at 4MHz in pseudo stop versus stop mode.

Following a loss of external clock in normal operation, Limp Home mode will be entered successfully but if the oscillator is reconnected for some reason a similar situation may arise.

The Reset condition can be overcome by allowing the crystal oscillator circuit to stabilize before releasing the external RESET line (see fig. 3). Operation is similar to that shown in fig 2.

To determine if crystal is 'stable' at the release of reset can be difficult and the time can vary some from board to board. If the customer has special high impedance probes, it is possible to monitor the amplitude of the voltage from XTAL to ground (<2 pF scope probes are recommended). Please note that any loading on the circuit can affect its operation. (Any resistance to ground or Vdd on the EXTAL pin can greatly attenuate the amplifier gain and cause erroneous operation.)

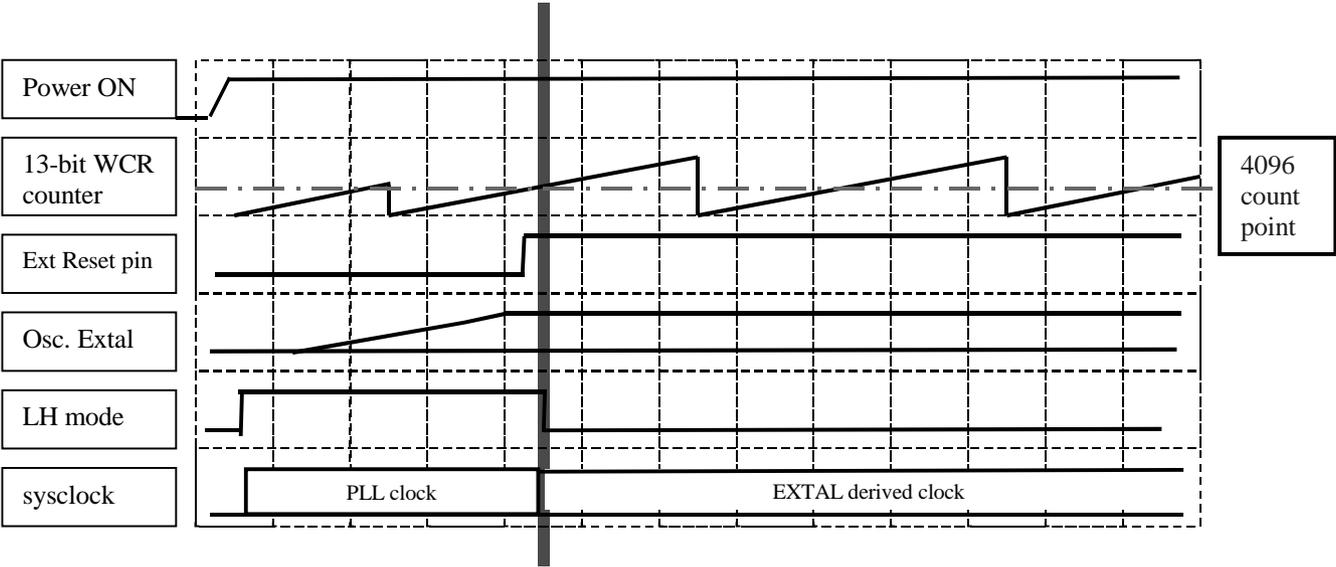
A second way to measure the oscillator startup time is to monitor the XFC pin. This method does not require a high impedance scope probe. The PLL will not lock until

the oscillator clock feeding it is present and stable. Remove the external reset circuit and during power up watch the XFC pin. The voltage should start high (Vdd). After the part releases internal reset it will drop to some stable voltage between Vdd and Vss. If external reset (measured independent from this test) is held till this 'stable voltage' time the oscillator will be stable. Please note the filter components mounted on the XFC pin will affect this ramp (for evaluation purposes, alternative components can be selected to provide a fast lock time). More than one board should be measured because of pcb and crystal variability. It is also recommended that the test be run over the operating temperature of the device.

Lastly, an alternative and simpler approach is to just hold reset low for a substantial time (> 100 milliseconds) after Vdd has reached the operating voltage range.

In some applications it may be possible avoid this issue by delaying the connection of Vddpll to Vdd until the device has exited reset. This will sacrifice the limp home mode safety function upon startup, i.e. the part will no longer be able to start without a functioning crystal. A similar technique (disable PLL under software control) can be used to overcome the limitations of Stop mode.

**Limp-Home mode start-up issue**



**Figure 1. Representation of Normal start-up condition**

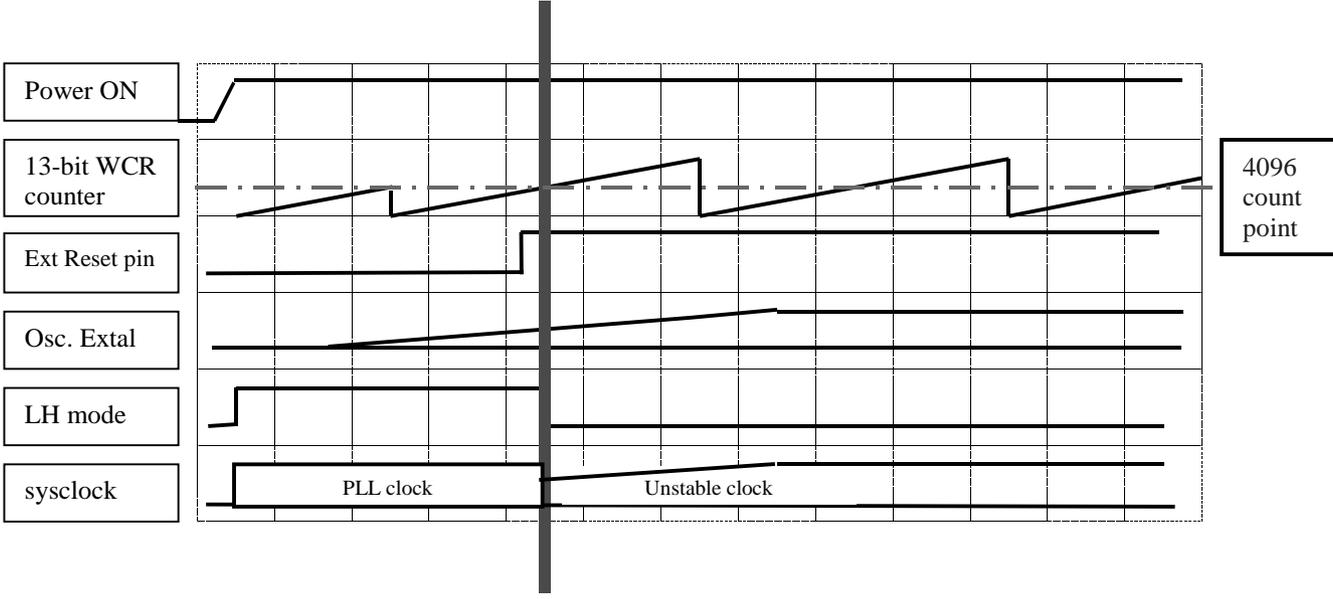


Figure 2. Representation of unreliable start-up mode with slow crystal

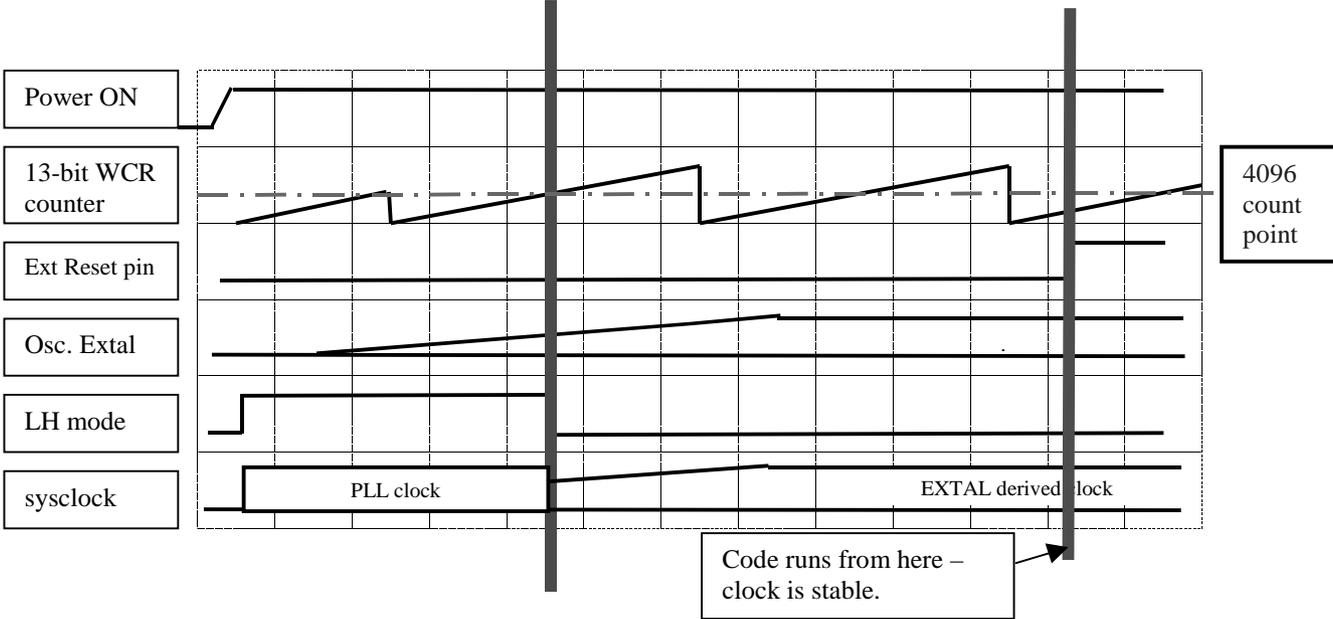


Figure 3. Representation of preferred means of overcoming issue with Figure 2

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

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



**MOTOROLA**

68HC912D60MSE1