Order this document
by AN997/D

AN997

CONFIG Register Issues Concerning the M68HC11 Family

Introduction

Some customers and field representatives have expressed concerns about the reliability of the CONFIG (configuration) register in the M68HC11 Family of microcontrollers (MCUs). These fears are based on problems with early mask sets of the part; however, current production units and proper application of the parts have overcome these earlier problems.

The earlier fault mechanisms, the circuit changes to correct them, and application guidelines to prevent undesirable results are presented here.

Discussion of Concerns

First, there is a lack of understanding about the CONFIG register and mechanisms. Because the M68HC11 is the first MCU Family to offer semipermanent configuration options, customers are unfamiliar with the supporting circuit and system mechanisms. There has been very little printed material explaining these mechanisms and showing the preferred ways of using them. As a result, many customers have applied the M68HC11 in ways that contribute to problems or that make recovery from problems difficult.

Useful printed material now exists, which will help new customers avoid problems and older customers understand and correct problems. Refer to Section 3 of the *MC68HC11A8 HCMOS Single-Chip Microcomputer Technical Data*, Freescale document order number MC68HC11A8/D.

Second, customers can get parts with the wrong value in CONFIG. Freescale and distributors do not always ship products exactly as the customer requires. The problem is not always caused byFreescale or the distributor. Sometimes CONFIG gets corrupted by a customer's incoming testing or by the first power turn on, which can happen if reset is not properly controlled during power transitions. Even if all parts were shipped and handled correctly, CONFIG can still get corrupted if the manufactured product has a fault in its properly designed power reset circuitry. Application techniques allowing such an end assembly to correct itself automatically will be discussed in **Application Guidelines**.

Third, reset must be controlled any time V_{DD} is below minimum operating level. The low-voltage inhibit (LVI) circuit is required primarily for protection of EEPROM (electrically erasable programmable read-only memory) contents, but because CONFIG is also based on EEPROM, the protection is required even if the EEPROM array is not being used. Reset must be controlled because the CPU (computer processor unit) is not able to fetch and execute instructions properly when V_{DD} is below minimum operating level.

Presently, there are several economical ways to solve this problem. Two good external components for LVI reset are theFreescale MC34064 and the Seiko S-8054HN (or other S-805 series devices).

- The Seiko part, which is extremely low power (2 µA), TO-92 package, limited temperature range (–20° to +70°C), is available in various trip-point voltage ranges. The S-8054HN has a trip point usually most suited to a 1-MHz rather than a 2.1-MHz bus frequency.
- The Freescale MC34064 is available in TO-92 or SO-8, draws about 300 µA, works over –40° to +85°C, has a very well-controlled trip point, and is very inexpensive. Many other LVI components are available, often incorporated into a voltage regulator.

Early Fault Mechanisms

All known cases of CONFIG erasure or corruption are well understood. In all cases, the CPU starts to execute nonsense code as V_{DD} drops to an unacceptably low level. The most typical problem is due to power turn off, but the same problem can happen even if V_{DD} drops for only a very short period. Power turn on is less often a problem because most MCU users are accustomed to controlling reset for predictable startup. Because the M68HC11 is a fully static design, no RC oscillator startup delay is actually required; this requirement (which was 100 ms) was dropped with the latest documentation revision. If an application has some special requirement for accurate timing within the first several milliseconds after power-up, an RC delay on power-up may still be required. Since the RC delay is not required, the reset circuit can be as simple as the MC34064 LVI and a pullup register.

When V_{DD} drops, the CPU can misbehave in a number of ways, almost all of which altered EEPROM or CONFIG on early mask sets of the M68HC11 (A38P, A49N, and before). However, the current parts are not nearly as sensitive to accidental EEPROM errors.

First, when V_{DD} drops, the branch instructions have a tendency to stop branching sooner than other CPU functions fail to work because the logic circuits used to resolve branch conditions are more complex than instructions such as load, store, etc. As a result, programs tend to be executed linearly, ignoring branches. The instructions intended to do EEPROM programming are soon encountered erroneously, and some change (write or erase) takes place. The likelihood of this occurrence depends on lot-to-lot processing differences, which affect the V_{DD} level where branch instructions stop working. On much older A38P material, this condition was a primary cause of EEPOM corruption, but it is a much less common cause on more recent parts. An example of how the drop out of branch instructions can be used to your advantage is shown in the BUFFALO monitor version 2.5.

Second, when V_{DD} drops, an opcode is incorrectly read from memory (either internal or external) resulting in an illegal opcode interrupt. In many user systems, this vector is not initialized; thus, another illegal opcode is usually encountered before an RTI (return from interrupt), which in turn creates an infinite series of illegal opcode requests. The stacking operations cause the registers to be repeatedly written to all memory locations in descending order. At 2 MHz, the entire memory could be filled in less than 60 ms by this mechanism. In A38P, A49N, or older mask sets, having EELAT and EEPGM both set to 1 would cause programming or erasure. Since stacking takes nine bytes, all registers will eventually be written to the PPROG address, and it is likely that at least one of the registers will have its two least significant bits set (corresponding to EELAT and EEPGM). This mechanism will have a tendency to byte or row erase $B600 since CONFIG is not byte or row erasable. Thus, in the case of stack runaway, $B600 would be the last EEPROM address written to until the next wraparound. If the BYTE and ROW bits happen to be 0s and ERASE is 1, a bulk erase of everything will occur, including CONFIG, since in this case CONFIG would be the last EE location written until the next wraparound of the stack. Depending on the other bits written to PPROG, several results may occur including programming or erasure of EE locations or CONFIG. This failure mechanism was common on older parts (A38P, A49N, and before).

Since the runaway problem concerns CPU operation at low levels of V_{DD}, many other sequences are possible. It should be clear that the CPU should not be allowed to try to execute instructions when V_{DD} is too low to support proper operation. Fortunately, because of the standby RAM feature of the M68HC11, the reset logic was specifically designed to operate properly to extremely low levels of V_{DD}.

Questions of Interest

Question At what level of V_{DD} does the CPU misinterpret commands?

Answer This level depends on processing, operating temperature, and operating frequency. At 2.1 MHz and 125°C, the CPU could fail if V_{DD} drops below 4.5 V. At 1 MHz and room temperature, the CPU should operate correctly with V_{DD} as low as 2 V (an indication, not a guarantee).

Question At what level of V_{DD} is the internal programming supply incapable of changing EE?

Answer EEPROM changes have been caused in the MC68HC05A6, which is similar to the M68HC11, at V_{DD} levels as low as 2.0 V. Most Freescale EEPROMs stop being able to program at 2.5 V to 3 V. The lower the V_{DD} and the slower the E, the longer it takes to program or erase because the charge pump efficiency falls off.

Digital logic devices and conventional opamps are not generally good enough for the LVI function since their operation is not guaranteed to a low enough supply voltage.

Question Why not just incorporate an LVI circuit on the M68HC11 chip?

Answer Experiments are being conducted that might eventually allow this, but the right combination of process capabilities does not presently exist. The M68HC11 uses a digital HCMOS process that is optimized and characterized for digital logic. The LVI circuits of choice use a linear CMOS process, which is not compatible with the current M68HC11 process.

Even if the process existed, the problem still is not solved because not all applications need the same trip-point voltage range. High performance applications need maximum bus speed, which, in turn, dictates a high trip point (~4.5 V). Battery-operated applications require very low power supply current and wider operating voltage range (~3.0 to 5.5 V to accommodate battery wear out). These applications sacrifice speed (32-kHz to 1-MHz E frequency). A trip point of about 3.0 V would be required for these applications.

Many system-related details must be resolved to allow and control this flexibility.

Question What about the long-term retention of the CONFIG EEPROM byte?

Answer The key to answering this question lies in understanding the wear out characteristics of the EEPROM. When the EEPROM is new (that is, it has been written and erased only tens of times), it has ideal oxides, programs quickly, and retains data very well.

As the cell is erased and reprogrammed thousands of times, the oxide degrades due to a charge-trapping mechanism. This trapped charge interferes with the operation of the EEPROM cell by increasing the programming time and, in more unusual cases, by reducing the retention time for programmed bits. The specified programming time (10 ms nominal) is set to a value that will assure proper programming after thousands of write-erase cycles; a new bit typically programs 2 or 3 ms. Theoretically, a bit that is programmed longer will retain its value longer. The benefit is limited because the rate of charge transfer is exponential; consequently, there is no significant improvement once the bit is programmed hard. Based on extensive reliability data, no parts have shown failure to retain data in the early stages of wear out (tens or hundreds of cycles).

The CONFIG byte should see only a few write-erase cycles in the life of a product; hence, it will have much shorter programming time and much longer retention than a byte programmed and erased thousands of times. The specifications are a statement of EEPROM capabilities after it has been exposed to thousands of write-erase cycles. Furthermore, it is significantly better than the specifications when it is new.

For bytes that have only been programmed and erased tens of times, the standard 10 ms programming time is more than enough to program a byte hard. There would be no special advantage to programming CONFIG longer than this time, although it would also cause no harm.

Circuit Design Changes

The original specification called out a programming procedure requiring several writes of specific data in specified order to enable the program/erase voltage. Unfortunately, the actual circuit design did not enforce this sequence; therefore, it was very easy to enable the programming voltage source to the array (until B96D and B46E mask sets). When the field problems associated with this error surfaced, Freescale made drastic changes to the programming logic to fully enforce the sequence needed to enable the programming supply. These changes actually went beyond what was originally specified, and they first appeared in production on the B96D mask set.

In the old circuit, a single write to PPROG with the two least significant bits set would initiate a program or erase function on some EEPROM location(s). It was not necessary to run into code that changed the EEPROM on purpose. In any runaway situation, there was a relatively high probability that something in EEPROM or the CONFIG would be corrupted.

In the current circuit, a sequence of at least three writes must be performed in the correct order with the correct data for any change to take place. An additional protection mechanism on the CONFIG location exists in that this register may only be changed while operating in one of the special modes (test or bootstrap). Getting into a special mode requires different levels on hard-wired mode pins. The current sequence required still conforms to the documented programming and erase procedures; therefore, no changes are required to user application software.

The current sequence is as follows:

1. Write to PPROG with the EELAT bit equal to 1 and the EEPGM bit equal to 0. EELAT will not change to a 1 unless the bit corresponding to EEPGM is a 0.
2. Write to an EEPROM location (or CONFIG). If an EE location is not written to before the next step, no EE location will be changed.

3. Write to PPROG with both EELAT and EEPGM equal to 1, which locks out any further changes to the latched address. Writing to an EE address after writing EEPGM to 1 will not change the address of the EE location being altered.
4. Write to PPROG with EEPGM equal 0. (Any write to PPROG with the EELAT or EEPGM bits equal to 0 will also cut off a programming or erase operation.)
5. Write $00 to PPROG.

The only shortcuts allowed to this procedure are:

- Clear PPROG at the end of a program or erase operation instead of the 2-step procedure (steps 4 and 5) or
- Leave EELAT equal to 1 between successive bytes when programming a series of locations (turn off EEPGM between successive bytes).

The probability of meeting the sequence requirements “by accident” has been drastically reduced. Reset must be controlled with an LVI circuit during power transitions to be sure no corruption will occur.

Application Guidelines

Always include a low-voltage detector to control reset. The easiest type is a 3-terminal device like the MC34064 or the Seiko S-8054HN. If an open drain type device is not available, a device with a push-pull output may be used by placing a resistor in series from the output of the LVI to the reset line of the M68HC11. The R looks like a pullup when V_{DD} is good; it looks like a pulldown when V_{DD} is too low.

Another circuit based on a Freescale TL431 is shown in Section 9 of the *MC68HC11A8 HCMOS Single-Chip Microcomputer Technical Data*, Freescale document order number, MC68HC11A8/D. This circuit draws several milliamps but provides a very good reset signal over the full –40°C to +125°C range. A nice aspect of this circuit is that it passively pulls down and only provides a logic 1 to the reset pin when V_{DD} is above the trip point. Other circuits stop working at some level of V_{DD}. For proper operation of the M68HC11, reset must be low until V_{DD} is less than 2.0 V.

The system can be configured so it comes out of reset in special test mode. Some of the system integrity functions are overridden while the MCU is in test (or bootstrap) mode. For example, the 64-cycle limitation on writes to sensitive registers does not apply in test mode, and the resets from the COP watchdog and clock monitor are inhibited.

Secondly, there are some important capabilities that are not available in normal operating modes. The mode itself can be changed while it is in a special mode (SMOD bit in HPRIO register equal 1). Software can begin in a special mode and change to a safer normal mode under program control. The reset and interrupt vectors are fetched from $BFC0 to $BFFF rather than from $FFC0 to $FFFF. In normal expanded operating mode, vectors in external memory are only accessible if the ROMON bit in CONFIG is 0. If the part comes with the wrong value in CONFIG or if CONFIG gets changed, the internal ROM could get turned on, making vectors inaccessible. A similar problem arises if the NOCOP bit gets enabled and software does not have the necessary routine to service it. Two approaches for correcting CONFIG are presented in the following paragraphs.

If the system is an expanded system with external program memory, an expanded test mode approach may be used to map a memory in such a way that reset and interrupt vectors are located in the $BFFx area and to wire the mode selects for special test mode. If 8-K ROM is used for external memory, leave address line A14 out of the decode for that memory such that it appears at both $A000 to $BFFF and $E000 to $FFFF. Most software would be written as if the memory were only in the $E000 to $FFFF area, but the initialization routine would be written as if it were in the $A000 to $BFFF area. In response to a reset, the MCU will execute this code even if CONFIG has the wrong value.

This code would check CONFIG and only correct it if it is wrong. Do not erase and reprogram CONFIG at every power up. If CONFIG is already correct, write to the time-protected registers as needed; then write to the HPRIO register to change to a normal operating mode. If CONFIG is wrong, correct it as needed, then force a reset to cause CONFIG to be

updated with the corrected value. This reset can be forced by software in two ways.

- One way would use the FCOP or FCM bits in the TEST 1 register.
- Another way, which does not use test features, would be to enable the clock monitor (CME bit = 1); enable reset functions (DISR to 0 in TEST 1 register); enable STOP (S bit in CCR register = 0); and then execute a STOP instruction, which will result in a clock monitor reset. The MCU should now have the corrected value in the CONFIG register.

A completely different approach uses the bootstrap mode to correct an erroneous CONFIG in the event of corruption. In this approach, the finished product is wired to reset in normal modes but can be reset in bootstrap mode under special circumstances. The procedure for reset in special mode can be hidden from customers, if desired. Rather than tying the MODB pin directly to V_{DD}, connect it through a pullup (4.7-k would be good), and provide a test point or other means so this pin can be pulled low to allow reset in special bootstrap mode.

The PD0/RxDS and PD1/TxD pins should be accessible in such a way that the activity on these pins during bootloading is compatible with normal application use. Usually, this procedure requires nothing new, except possibly a test point or wire so an external system could drive the RxD line. The loading process drives transmit data out the TxD pin; however, it is not necessary to monitor this information. Since the port D outputs are configured for open-drain operation during bootloading, a pullup may be needed on the PD1/TxD line.

The expanded test mode approach has the advantage of being completely automatic but is not necessarily the best choice. The bootstrap approach is a minimum requirement for a good new design.

Avoid any design in which CONFIG change involves removing the MCU from the finished product. Refer to Section 3 of the *MC68HC11A8 HCMOS Single-Chip Microcomputer Technical Data*, Freescale document order number MC68HC11A8/D.

General Comments

When resetting in normal single-chip mode, the internal 8-K ROM is enabled (regardless of the state of the ROMON bit in CONFIG), which means that the internal ROM will be accessible even if CONFIG gets corrupted.

If a part with security mode capability gets its NOSEC bit programmed to 0, erase CONFIG by attempting to reset in bootstrap mode. It is not necessary to load anything; just come out of reset in special bootstrap mode. The firmware in the bootloader ROM checks the NOSEC bit; if the bit is 0, it erases the EEPROM and CONFIG register before going to the downloading phase.

A problem existed on older parts involving the IRQ pin and programming EEPROM while certain other unusual conditions existed. The IRQ pin is used by Freescale to provide an external 19-volt programming supply for a bulk programming mode. An MOS switch on this pin, made from a single MOS transistor, worked fine when IRQ was high during EEPROM programming operations. The problem is caused by IRQ configured for edge-triggered operation and by a low-level on IRQ while an EEPROM programming or erase operation is being performed. The difference between the low level at the pin and the charge pump output (~20 volts) on the other side of the MOS switch caused a breakdown of the MOS switch. This breakdown (temporary and non-destructive) caused repeated IRQ requests, which hung the system long enough for a COP reset to occur. The customers were only programming the EEPROM in their factories; hence, they changed to a procedure that masked IRQ and/or kept IRQ high during the programming operation.

In the meantime, Freescale changed the design of the MOS switch to eliminate the breakdown mechanism. Current production parts are not subject to this potential problem.

Some MC68HC11A2 parts (and older MC68HC11A8 parts) were manufactured with EEPROM that was slow to recover from programming operations. This problem could cause the EEPROM to read incorrectly if an attempt was made to read an EEPROM location less than 11 E cycles (at E = 2 MHz) after EEPGM was written to 0 (corresponding to the end of the programming cycle).

Consider the problem described in the previous paragraph. In that case, the EEPROM programming operation got terminated by a COP reset. The COP reset drove the reset pin low for four E cycles and released it. On this rising edge of reset, the CONFIG EE location was transferred to the CONFIG working latches (like a read operation). Because of the slow recovery of the EEPROM, CONFIG was incorrectly initialized.

In test mode, the CONFIG register of the MC68HC11A2 can be written like a regular register. The MCU can be reset in test mode, the CONFIG may be written directly, then the mode controls may be changed to a normal mode. Even if the CONFIG EEPROM byte is completely stuck, normal operation is possible.

You can byte erase the CONFIG byte in the MC68HC11A2.

Conclusion

While the CONFIG register mechanism offers an unprecedented degree of flexibility to the MCU, it also presents some design challenges to the MCU user. Like many features of the MCUs, serious problems can arise if care is not exercised in their use. Many such problems have arisen in the use of the CONFIG register in the M68HC11 because it is a new function, and guidelines for proper application were not readily available. This application note has explained the problems and what should be done to correct and prevent them. Following the guidelines set forth here will assure reliable operation of the CONFIG register mechanism in the M68HC11.