Engineering Bulletin

EB398/D
Rev. 0, 8/2002

Techniques to Protect MCU Applications Against Malfunction Due to Code Run-Away

By: Peter Topping
Motorola, East Kilbride

Introduction

The MC68HC(9)08 range of MCUs include features to help prevent code run-away and to protect against application malfunction when it does occur. Code run-away can be caused by faulty code, operating the MCU outside its specification or by a major EMI or electrical noise event. By definition, it is not well defined what will happen during code run-away, but it is caused by the out-of-specification operating environment effectively corrupting the program counter resulting in the MCU behaving unpredictably.

The precautions described below are recommended in MCU applications where this could be detrimental. Even when the recommended precautions are taken, there is still a small probability of code run-away under exceptional conditions. For this reason, techniques are included which prevent any damage to the MCU or application hardware in this eventuality. These precautions are particularly relevant in applications that include any kind of on-chip or external non-volatile memory (Flash, EEPROM or backed-up RAM) or any possibility of external hardware being put into an undesirable configuration or even damaged.

During code runaway the MCU is by definition behaving unpredictably so even the I/O ports cannot be relied on to continue outputting acceptable states. There is thus the potential to put the ports, and hence any external hardware, into an unanticipated configuration.

In applications which include non-volatile memory there is a possibility that its contents could be corrupted by uncontrolled behavior of the MCU. This would be particularly serious in the case of Flash or EEPROM memory that contains application code. If this is corrupted, the application may be rendered non-functional with no possibility of recovery short of reprogramming. This would require the equipment to be repaired by in-circuit reprogramming or perhaps even necessitate the replacement of the complete PCB containing the MCU.

In simple ROM-based applications there may be no non-volatile memory and no concern about unpredictable I/O configurations. In this case there is no pressing need to take precautions against code run-away although it could be used to prevent, for example, the disorganized flashing of LEDs or other displays as the supply falls.

Even if there are no foreseen possibilities for the application to malfunction in a detrimental way, it is good practice and highly recommended that all reasonable precautions are taken to prevent code run-away. These recommendations are not new but they are sometimes forgotten and are thus not always implemented. The specific devices discussed are MC68HC(9)08s but the techniques are applicable to all MCU applications.

1. Prevention of code-runaway

Code run-away is a potential risk during both power-up and power-down. The two cases are handled in different ways.

1a. Power-down protection

The most important feature for the prevention of code run-away is the low voltage inhibit feature (LVI). This has the capability of holding the MCU in reset when the Vdd supply falls below the specified minimum. While this would protect the application during a power or power-supply failure, its most important use is during the transient conditions which occur when the power is intentionally switched off. If some form of LVI is not implemented, code run-away is highly probable at every power-down. There are two alternative methods of implementing an LVI - internal and external.

The on-chip LVI is the most obvious choice as it does not incur any additional cost. In applications using an MCU with an internal LVI (all MC68HC(9)08s) and not employing an external LVI, the internal one should always be enabled. Both of the CONFIG register bits LVIRSTD and LVIPWRD should be zero. This is its default status but in devices with a 3 volt option, care should be taken to use the correct voltage. The default is 3 volts so in 5 volt applications the appropriate bit (LVI5OR3) should be set. Even if the default values in the CONFIG register are acceptable, this register and all other write-once registers should be written to at the beginning of the application code. This provides additional security as any attempt to write to these registers during code run-away will be locked out.

7	6	5	4	3	2	1	0
	LVISTOP	LVIRSTD	LVIPWRD	LVI5OP3			

LVI control bits in the CONFIG register

In devices like the MC68HC(9)08KX8 or the MC68HC(9)08GP32 that are specified to operate down to 2.7 volts, the LVI provides excellent protection in a 5 volt application. It is guaranteed to trip somewhere between 3.9 and 4.5 volts and thus ensures that the reset will be held low well before the supply falls below the minimum Vdd voltage specification. To achieve the maximum benefit, a legal 3 volt bus speed (not more than 4MHz) should be used. This ensures that normal operation is guaranteed all the way down to 2.7 volts.

As shown in table 1, the 5 volt specification of the MC68HC(9)08's LVI guarantees to hold the reset line low below its trip point (between 4.5 and 3.9 volts). In a 5 volt only MCU like the MC68HC(9)08AZ60, normal operation is guaranteed down to 4.5 volts. Although correct functional performance is expected down to the LVI trip point, it is not actually guaranteed, and in safety critical applications an external LVI should be employed. This can be chosen to trip before the supply voltage falls below 4.5 volts, thus giving a 100% assurance that the chip will not be asked to execute code out of its specified voltage range. Suitable external LVI devices include the MC33064 and the MC34064 and an appropriate circuit is shown in figure 1.

Table 1. LVI trip voltages

V_{DD}	LVI Trip point (falling supply)		
	Minimum	Typical	Maximum
3 volts	2.45	2.60	2.70
5 volts	3.90	4.25	4.50

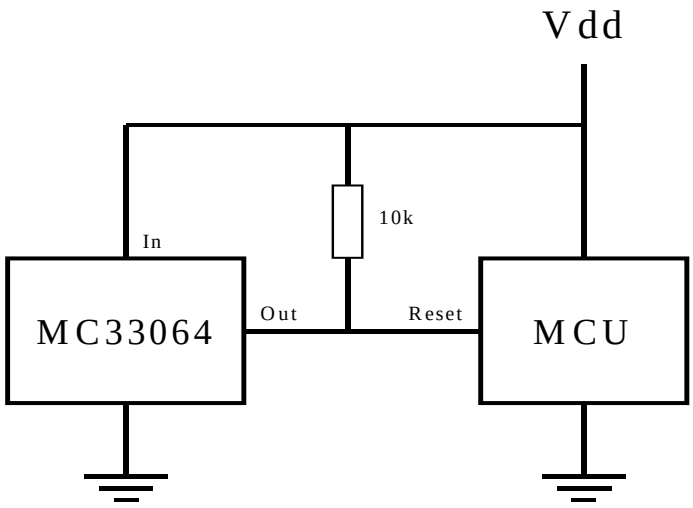

Figure 1. External LVI

Some applications may require the saving of status information into internal or external non-volatile memory when the power falls. The LVI can be used for this purpose but care should be taken to avoid operation out of specification. If the LVI is enabled without enabling its reset function, then polling the LVIOUT bit in the LVI status register can be used to initiate the required information transfer. This should, however, only be done if the supply voltage is still legal. It is thus only appropriate to do this if the 5 volt LVI trip point is used in an application which is also specified to run down to 3 volts. This can for instance be achieved in a 908GP32 or a 908KX8 as long as the bus speed used is 4MHz or less.

If this method is adopted, it is up to the designer to ensure that the reservoir capacitor is large enough to allow sufficient time to execute the required code well before the voltage reaches 3 volts[1]. Clearly this procedure does not provide total LVI power-down protection and it is not generally recommended.

There are methods of initiating this type of process which do not use the LVI and thus leave it available to be fully enabled (with its reset) to provide full power-down protection. They do, however, require an MCU pin and some external hardware. This can, however, be limited to a single 2-pin voltage reference or 3-pin under-voltage sensing (LVI) chip. Possibilities include an LVI chip (e.g. MC34064) feeding a polled input pin and an A/D pin monitoring a band-gap reference (e.g. LM385-1.2). In this latter case (see figure 2), the external reference would remain valid as the supply falls whereas the A/D reference would fall with Vdd. The effect would thus be that the A/D measurement of the reference would rise as the supply voltage falls and the measured value can be used to initiate the appropriate routines at the desired voltages.

1. If code is intentionally run at a Vdd below the specification, then it should be run from RAM rather than from Flash. This is because RAM will read correctly down to lower voltages than is the case for Flash. Correct operation is, however, not guaranteed in either case if the Vdd is below the specified minimum.

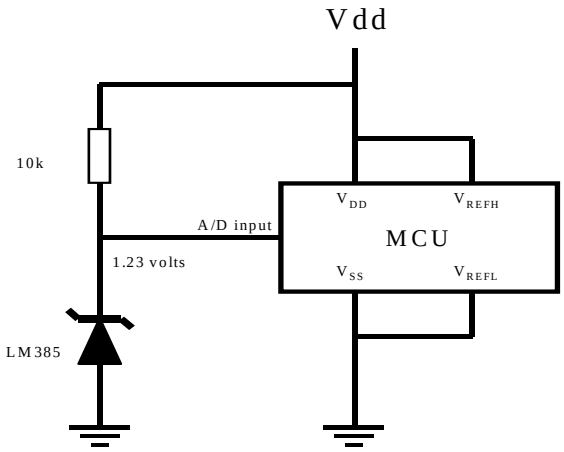

Figure 2. External Voltage Reference

In some battery applications there may be two or even three thresholds at which it is necessary to take different measures (saving information, switching off power-hungry hardware etc.). The A/D method facilitates this with the use of only a single MCU pin.

1b. Power-up protection

The internal LVI can control the reset pin during power-down but care also needs to be taken during power-up. The internal power on reset (POR) circuitry holds the reset pin low prior to oscillator startup (typically a few milliseconds after Vdd has reached 2 volts) and then keeps it low for 4096 clock cycles. If this is not long enough to ensure that the supply voltage is within specification before reset goes high, then measures should be taken to hold the reset pin low for longer. An external LVI like the MC34064 (see figure 1) will hold reset low until the supply is above its trip point. In the absence of an external LVI, the most common method is the addition of a capacitor, as well as a pull-up resistor, to the reset pin as shown in figure 3. This delays its rise for sufficient time for the supply to meet the specified minimum. It is up to the designer to use his knowledge of the system power supply to determine an appropriate time constant. Values of 10kΩ – 100kΩ and 0.1µF – 1.0µF are typical. When using this reset pin circuitry to provide power-up protection, the internal LVI should always be enabled to provide power-down protection.

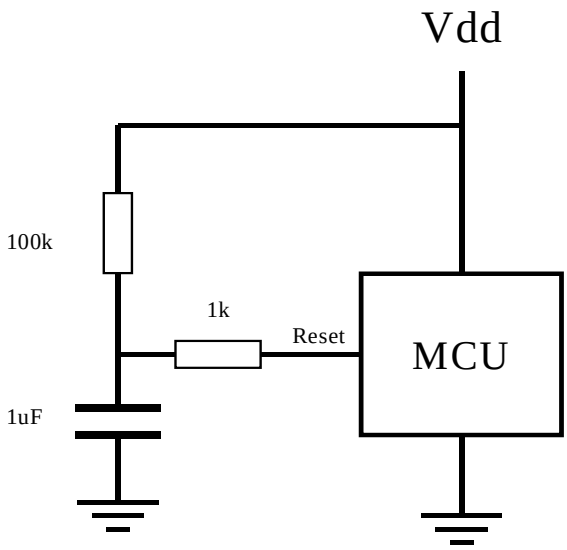

Figure 3. Simple external pull-up and capacitor

The resistor in series with the reset pin in figure 3 is optional. It is not strictly necessary as an internally generated low on the reset pin is buffered before it appears on the pin. It is, however, recommended that the behavior of the reset pin during power-up and power-down is checked with an oscilloscope. The inclusion of this resistor allows the signal to be observed more clearly as any internally generated low condition is not compromised by the presence of the external capacitor.

1c. Overall control of the reset pin

Figure 4 shows typical behavior of the supply voltage and the reset pin during power-down and power-up. At power-down the internal LVI pulls the reset pin low as the supply voltage goes out of specification and holds it low until power is re-established. If this feature is enabled and functioning properly there is no specification for the fall time of the supply voltage. This allows the implementation of designs that may have a very long decay time due to the use of a large power supply reservoir capacitor or low system current drain during power-down.

The LVI can only force a low level on the reset pin as long as there is a sufficient supply voltage for its circuitry to function. Close monitoring of the reset pin may thus show it drifting back up to a few tenths of a volt once the supply has fallen very low. This is normal. It only happens once the supply has fallen so low (less than a volt) that no code execution or Flash corruption is possible.

The MC68HC(9)08 specifications include the requirement that the supply starts from below 100mV (200mV on some devices, see technical data) to guarantee

an internal power on reset (POR). The power-down decay time constant in some applications may be very long when the voltage is close to zero as semiconductor components cease to conduct. This should be considered in the system design so that an unreasonably long time is not required after a power-down for the residual Vdd to fall to this level. This could, for example, take the form of an additional resistive load across the main power-supply capacitor.

The initial ramp rate of the supply should also be fast enough to guarantee a POR. The required minimum ramp rate depends on the Vdd being used an on the particular MCU. As an example, the MC68HC908GP32 5 volt specification gives a figure of 35 volts per second. This figure (and 20 volts per second at 3 volts) is shared by most MC68HC08s but the 908AZ60A and 908AB32 are specified at 20 volts per second at 5 volts.

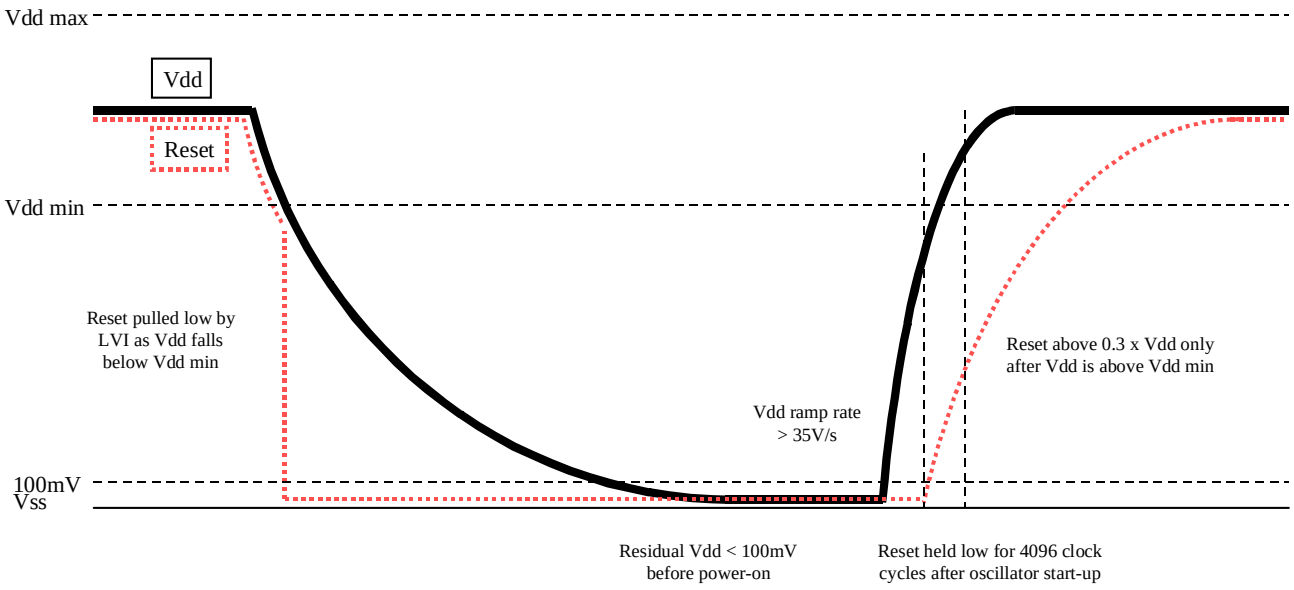

Figure 4. Typical Vdd and reset pin waveforms

Figure 4 shows the behavior of the reset pin assuming an external RC circuit of the type shown in figure 3 and correct operation of the POR circuitry. The POR holds the reset pin low until the oscillator has started and has executed 4096 cycles. At this point the pin is released and it goes high under the control of the external RC circuit. The RC time constant should be chosen using the designer's knowledge of the power supply to allow the reset pin to go above its Vil specification (0.3xVdd) only after Vdd has exceeded its minimum specification.

2. Techniques to minimise the effects of code run-away

Even when the recommended precautions have been taken, there is still a small possibility of code run-away under exceptional conditions. For this reason, the following precautions should be observed to prevent any damage to the MCU or application hardware in this eventuality.

2a. Unused Flash or ROM Addresses

By definition code run-away is the corruption of the MCU's program counter (PC). To guard against the possible detrimental effects of unintentional execution from unused addresses in the program code area of the memory map, they should always be filled with known safe code. This applies whether the code is stored in ROM, Flash or EEPROM. In ROMed applications the only concerns are I/O behavior and the possibility of corruption of data stored in EEPROM. In Flash based applications the additional possibility of code corruption is also present.

All unused addresses should contain a sensible and appropriate instruction. This could be a sequence of NOPs ending in an SWI or, more simply, all SWI instructions. If this is done, any inadvertent execution of code in these addresses will result in a software interrupt. It is up to the programmer to decide exactly what happens in this eventuality.

The SWI vector should be programmed to point to an appropriate error handling routine. This could be an infinite loop that doesn't feed the COP (see below), the execution of a STOP instruction or a jump to the entry point of the application software.

The most appropriate strategy will be different for different applications. A STOP instruction does just what it says: no further instructions will be executed and the I/O configuration freezes at its current status. If an infinite loop is used then the COP will force a reset but the maximum time this will take to occur is dependant on the COP refresh interval and on the exact point in that interval at which control was lost. If the last method is employed (a jump to the entry point of the application), the programmer should remember that there has not been a reset and deal correctly with the stack pointer. It should be initialized appropriately at the start of the code before any interrupts are enabled. In the absence of a reset, write-once registers will still be locked out.

Another possibility is to fill un-required addresses with an illegal op-code (e.g. $32). This would force an illegal op-code reset if it were executed. It is not appropriate, however, to leave spare locations at a default of $00 or $FF as these are legal instructions. The behavior of BRSET0 ($00) depends on portA's data register while an indexed STX ($FF) will store the index register at some unintended address. Neither is advisable.

All unused vectors should also be filled with the address of an error handling routine in case the corresponding interrupt occurs inadvertently.

It is also advisable to use any other features provided to detect undesirable behavior. When a PLL is being use, for instance, the out-of-lock interrupt capability should be enabled and used to determine what should happen if the clock speed is suspect.

2b. Internal and external COPs

If an external watchdog chip is not being used, the on-chip COP should be enabled and regularly written to during normal code execution. This should be done within the main code and not from an interrupt routine. This is recommended as interrupts could continue to function correctly even if the main background task has failed. Correct use of the COP will catch most instances of code run-away not detected by other means. In conjunction with the appropriate filling of unused addresses, the COP can provide a full reset in the event of program counter corruption.

The watchdog timeout period should be chosen to be as small as is possible consistent with the application's system requirements. Clearly lower timeout periods provide greater protection but the degree of protection against Flash corruption is limited. The trend towards faster writing of large blocks of Flash means that modification can happen very quickly (in microseconds) and the COP thus provides only limited protection. It should thus always be used in conjunction with the other protection methods described in this engineering bulletin.

2c. Flash block protect on 908 MCUs

A particular risk associated with code run-away is the possibility of non-volatile memory corruption in applications that include internal or external Flash or EEPROM. In this type of application it is very important that all the precautions described above are taken to prevent code run-away. Even when these precautions have been taken, it is, however, still good practice to enable the on-chip Flash protection features.

In MC68HC908 devices this takes the form of a FLash Block Protection Register (FLBPR) which can be configured to protect a user selected address range within the Flash memory. If appropriate in the application, it is recommended that the complete address range be protected. The full address range occupied by executable code should always be protected. On all the 0.5µm MC68HC908 devices shown in table 2, the default erased value of FLBPR ($FF) disables all protection. Any value other that $FF protects FLBPR itself and the range of addresses determined by the actual value. Once protected, FLBPR can only be modified with the higher V_{TST} voltage applied to the IRQ pin. The only exception to this is the MC68HC908AZ60A which does not require a high voltage on IRQ to facilitate modification of FLBPR.

The mapping of FLBPR into the start address of the protected range varies between MC68HC908 family members according to their particular memory map as shown in table 1. The FLBPR corresponds to the indicated bits of the start address of the protected area. Higher order fixed bits are ones (bit 15 is a zero in the case of Flash block 2 in a 908AZ60A) and lower order bits zeros.

The table is only presented as an indication of the mechanism, the data sheet for any particular device being the definitive source of information. The end address is always $FFFF ($7FFF for Flash 2 on the 908AZ60A). This method results in a resolution of the start addresses of most 908 devices of 64 or 128 bytes. This corresponds to the block (or page) size of each device[1]. For the devices in the table, FLBPR is itself a Flash register but in some 908s (e.g. the 908JL/JK) it is implemented as a read-write register and thus has to be written to by the application software after every reset or power-up.

Table 2. Flash Block Protect Register (FLBPR)

Device	FLBPR address	Start address				Resolution	Full protection
		Bit 15	Bit 14	Bit 13	FLBPR mapping		
908AZ60A - 1 block 2	$FF80 $FF81	1 0	– –	– –	Bits 14-7 Bits 14-7	128 bytes 128 bytes	$00 $00
908AB32 908GP32 908MR16/32	$FF7E	1	–	–	Bits 14-7	128 bytes	$00
908EY16 908GT16	$FF7E	1	1	–	Bits 13-6	64 bytes	$00
908MR8 908GR8 908KX8	$FF7E	1	1	–	Bits 13-6	64 bytes	$80

If suitable precautions are not taken, the risk of Flash corruption in the event of code run-away is higher in small RAM devices like the MC68HC908KX8 which include ROMed Flash erase and write routines (these routines are used for Flash & EEPROM burn-in write/erase cycling and testing). The risk is that a corruption of the program counter could cause unintended execution of this ROMed code and consequent Flash corruption.

1. The Flash block protection register worked differently on older 0.65µm devices like the non-"A" 908AZ60

In larger devices like the MC68HC908GP32 or the MC68HC908AZ60A, there are no Flash routines in ROM. If it is consistent with the system's requirements, the user should avoid including their own Flash modifying routines in the application software. This doesn't necessarily prevent the ability to update code in the field as a small bootloader can be included which can be used to download appropriate Flash routines for execution from RAM when required. Alternatively the 08's monitor mode can be used in conjunction with P&E's PROG08SZ software. Even with these lower risk devices, however, the Flash protection feature should always be enabled.

3. Summary

1. Enable the LVI. In a 5 volt only application, the current specification of the LVI will prevent code runaway below 3.9 volts and should be used to ensure a reset when the supplies falls. In devices like the 908KX8 or the 908GP32 which are specified down to 2.7 volts, the LVI provides even better protection in a 5 volt application.
2. Fill any unused bytes of application code (ROM or Flash) with a sensible instruction (e.g. an SWI). The SWI vector (and all other unused vectors) should be programmed to point to an appropriate error handling routine.
3. Enable the COP and clear its counter at an appropriate frequency, lower timeout periods provide greater protection.
4. In Flash based applications, fully enable the Flash block protection feature.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://www.motorola.com/semiconductors

MOTOROLA