Engineering Bulletin

EB608/D
8/2002

Interrupt Handling Considerations When Modifying EEPROM on HC08 Microcontrollers

By Mark Martinets
Field Applications Engineering
Transportation & Standard Products Group

Introduction

This engineering bulletin discusses the various issues associated with handling interrupts while programming or erasing EEPROM (electrically erasable programmable read-only memory) within different members of Motorola's HC08 Family of microcontrollers. Due to the various memory technologies used within the HC08 Family, it is necessary to discuss the techniques that can be utilized when an interrupt occurs.

Background

Some microcontrollers contain internal EEPROM that can be utilized in various applications as a means to store dynamic application data. The advantage of EEPROM over other types of memory is that small amounts of data can easily be stored for later use similar to RAM, yet it is non-volatile and will retain its contents when power is removed similar to FLASH.

In order to store data to EEPROM, it is usually necessary to use a specific algorithm to program the data to the non-volatile memory. The time required to execute this process can range from only a few milliseconds to ten times that amount. In addition, it is sometimes necessary to erase the location where the data is to be stored before the information can be programmed. This adds additional time to the storage process.

Some microcontrollers operate in "real-time" applications. These are typically systems that respond or react immediately to its inputs or other external stimuli. In a real-time system or application, spending more than a few milliseconds to perform a single function, such as modifying data in non-volatile memory, may be unacceptable if a higher priority function needs to be performed. Hence, it may be necessary to interrupt the data storage process. If it is necessary to

interrupt the process, certain precautions should be taken to ensure the subsequent correct operation of the EEPROM. Specifically, if the application requires data to be read from the EEPROM array that is being programmed or erased, a delay from when the operation is interrupted and halted until the EEPROM array can be accessed is required.

An example of such a situation would be if an application was erasing EEPROM, which currently takes 10 ms on an HC08 microcontroller, and a high priority interrupt occurs 3 ms into the operation. If the service routine requires data from the EEPROM in order to process the interrupt, the application would have to cease the erase operation, then wait the necessary amount of time for the high voltage to be removed from the array before attempting to read data from the array.

Within the HC08 Family of microcontrollers, there is a sub-family of devices that possess internal EEPROM. The members of the A-Family of devices that have EEPROM contain the identifier AB, AS, or AZ in their nomenclature. There are two different sized technologies used for this sub-family — 0.65 µm and 0.5 µm (see **Table 1**).

Table 1. A-Family Devices Containing EEPROM

0.65 µm Devices	0.5 µm Devices
XC68HC08AZ32 MC68HC(9)08AS32 MC68HC908AS60 MC68HC(9)08AZ60	MC68HC08AB16A MC68HC908AB32 MC68HC908AS32A MC68HC(9)08AZ32A MC68HC908AS60A MC68HC(9)08AZ60A

The 0.5 µm MC68HC(9)08AB/AS/AZ devices utilize a newer EEPROM technology that differs slightly from the older 0.65 µm devices and are typically identified by an “A” suffix at the end of the part number. The programming algorithm for the older devices is compatible with the newer devices; however, an accurate and specific programmable timebase is now needed for proper program and erasure.

Device Operation

The EEPROM array(s) can be programmed or erased without an external voltage supply because the microcontroller possesses an internal charge pump. The charge pump provides the necessary high voltage required to perform these functions.

0.65 Micron Devices

When programming or erasing the 0.65 µm devices, it is necessary for the user to design the application to implement the necessary timing delays in the respective algorithms. The delays are needed to allow the high voltage from the internal charge pump to be applied to the EEPROM array for the prescribed amount of time. If these timing constraints are not met, the array may not be programmed or erased correctly, or worse, rendered inoperable for future programming or erasing.

There is a control bit located in one of the EEPROM configuration registers that manages the operation of the internal charge pump. The EEPROM program/erase enable (EEPGM) bit is a read/write bit that enables the internal charge pump, which in turn applies the programming/erasing voltage to the EEPROM array. Hence, a portion of the typical flow of a programming/erasing algorithm implemented by the user should involve setting the EEPGM bit, waiting for a prescribed delay time, and then clearing the EEPGM bit.

If the EEPGM bit is cleared before the prescribed amount of delay time has passed, the programming/erasing voltage is immediately removed from the EEPROM array. Subsequently, the success of the programming/erasing operation cannot be guaranteed.

0.5 Micron Devices

The internal control logic associated with the EEPROM array(s) found in the 0.5 µm devices has been enhanced in order to automate some of the programming/erasing functionality. The EEPROM array found on the 0.5 µm devices requires a much shorter program/erase delay time. In order to allow backwards compatibility with the 0.65 µm device program/erase algorithms, provide alternative algorithms that are more robust, and provide a more automated means of programming/erasing a state machine was implemented into the control logic.

The state machine program/erase cycle is initiated when EELAT and EEPGM bits are sequentially set. When the cycle begins, the state machine enables the internal charge pump, hence applying the programming/erasing voltage to the EEPROM array. After the automatic EEPROM program/erase time has expired, the state machine removes the high voltage from the EEPROM array by disabling the internal charge pump. If the AUTO bit is set in the previous step, then the EEPGM bit is also cleared, indicating to the user that the programming/erasing cycle is complete. Since the timing for programming/

erasing the EEPROM array is critical, the state machine must, in some way, understand the internal bus clock speed.

The EEPROM state machine requires a 35 µs timebase for correct program/erase of EEPROM content. This timebase is derived from dividing the CGMXCLK or bus clock using a timebase divider circuit controlled by the EEPROM timebase divider register.

When programming or erasing the 0.5 µm devices, the user may implement the timing delays as in the 0.65 µm device program/erase algorithms, or the AUTO mode can be enabled and utilized. If the user implements the former method, the normal, continuous flow of the algorithm defined for 0.65 µm devices will operate on a 0.5 µm device in the same manner. Since the timing delay utilized in the standard (manual) algorithm is greater than automatic EEPROM program/erase time delay, the state machine automatically disables the internal charge pump before the user clears the EEPGM bit.

When enabled, AUTO mode will automatically clear the EEPGM bit when the program/erase cycle is complete. As in the standard method, the state machine disables the internal charge pump after the automatic EEPROM program/erase time delay has passed.

If the EEPGM bit is cleared before the automatic EEPROM program/erase time delay has passed, the internal charge pump is NOT disabled and the programming/erasing voltage is NOT immediately removed from the EEPROM array. Consequently, accesses (read) of the array are still prohibited internally. If there is an attempt to read any part of the array, the resulting data read is indeterminate. Also, any modification of any EEPROM related registers could result in incorrect operation, including complete erasure of the EEPROM array. To prevent array erasure and any other improper operation from occurring, EEPROM related registers should not be modified until the remainder of the automatic EEPROM program/erase time delay has passed and the state machine disables the internal charge pump.

Handling Interrupts

To ensure proper execution of the programming or erasing algorithms, it is recommended to disable interrupts. The best situation is to disable interrupts so that the respective algorithm can finish executing without disruption. This removes the possibility of memory cells not being completely erased or programmed, or worse, inadvertent erasure of the EEPROM array.

However, as previously mentioned, there are ‘real-time’ applications that require the immediate handling of the highest priority conditions, which are typically presented in the form of interrupts. If it is necessary to have interrupts enabled while executing program or erase algorithms, then extra care should be taken in how they are handled.

0.65 Micron Devices

When an interrupt occurs in the 0.65 µm devices, there are two possible ways the service routine can handle the interrupt if it occurs during an EEPROM program/erase sequence. The flow that the routine takes is determined by whether the routine needs to read data from the EEPROM array or not (see **Flow Diagrams**). If it is not necessary to acquire data from the array, then the interrupt service routine can execute without halting the program/erase sequence. However, care must be taken to continue to monitor the program/erase delay time in order to correctly resume operation of the program/erase sequence upon return from the interrupt. Once the application returns from the interrupt service routine, it can complete the program/erase sequence in a normal manner. If the required minimum amount of delay time has already expired upon return, then the application should immediately execute the final steps of the sequence and terminate the algorithm.

If the user chooses not to monitor the delay time while in the interrupt service routine or if it is necessary for the service routine to access the EEPROM array, then the first action that the service routine should take is to clear the EEPGM bit. This will immediately disable the internal charge pump and in turn, remove the high voltage from the EEPROM array. It should be assumed that whichever function was being executed, programming or erasing, was not completed correctly and should be repeated when possible. If the array was being programmed, then the memory locations should be erased first, then reprogram the data. This will prevent the possibility of over-programming the memory cells.

Immediately following the clearing of the EEPGM bit, the application should wait for the defined amount of time that will allow the high voltage on the EEPROM array to fall (t_{EEFPV}). Since the user's application should not be accessing the array during the EEPROM programming voltage discharge period, other operations can be performed while allowing the high voltage to fall. Care should be taken to ensure enough time is allowed for the voltage to be removed.

Following the delay for the high voltage to fall, the next step is to clear the EELAT bit. This will place the EEPROM control logic into its standard state that will allow the user to read from the array or initiate subsequent program or erase cycles. The remainder of the service routine, which requires access to the EEPROM array, can properly be executed at this point.

0.5 Micron Devices

Interrupt handling on the 0.5 µm devices is similar to that of the 0.65 µm devices; however, there are some significant differences that the user needs to understand. Since the algorithms can be executed in either a standard or AUTO mode, the manner in which the interrupt is handled will be dependent on the operational mode of the algorithm.

The better situation will be if the algorithm is being executed in AUTO mode. This is because an interrupt can more easily be serviced without adversely affecting the flow of the algorithm and can still conform to the prescribed algorithm routine. Since the state machine will automatically clear the EEPGM bit, an interrupt service routine can return program execution to where it left off without having to monitor the delay time while within the service routine. If the EEPGM bit has been cleared because the state machine has finished the program/erase cycle, the application will proceed normally. If not, then the application should continue polling the EEPGM bit where it left off, waiting for the cycle to finish. However, note that accessing the EEPROM array during the service routine would not be possible since the current state of the state machine would not be known, unless the service routine also polls the EEPGM bit and can determine when the algorithm is finished.

If the device is executing the algorithm in standard mode, this will be comparable to the algorithm flow on a 0.65 µm device. Hence the interrupt will be handled similarly. As previously noted, if the service routine needs to read data from the EEPROM array the first thing to do is clear the EEPGM bit. Since the internal state machine controls the charge pump, clearing the bit does not disable the charge pump nor remove the voltage from the array. This action simply allows the state machine to return to its idle state when it finishes its programming/erasing cycle.

Immediately following this, the EELAT bit can be cleared. Again, the state machine is controlling the array, hence this will not have any immediate effect. This bit must also be cleared to allow the state machine to return to its idle state once the internal time delay has expired.

Once these two steps are done, the user must wait for the state machine to disable the charge pump before the array will be ready for normal use again. Since the prescribed delay time for programming/erasing the EEPROM array using the standard method is greater than that of the internal timer, the minimum amount of time to wait is indeterminate. Therefore, it is necessary to wait for a period of time equivalent to the EEPROM programming/erasing maximum time to AUTO bit Set time as defined in the device specification. If this is not done and a subsequent program/erase cycle is attempted, the cycle could possibly be ineffective (no program or erase operation would occur). The cycle would simply act as a delay and return the array to a ready state once it completed. Also, if an attempt to read from the array is made, the data being read will be indeterminate. Hence, the appropriate amount of delay time must pass before correct operation of the array can be insured. Another alternative is to monitor the delay time within the service routine as described in **0.65 Micron Devices**. Care should also be observed in how the program returns from the service routine. Since the program/erase sequence has essentially been completed within the routine, program flow should return to the end of the normal algorithm sequence.

It is recommended that if it is necessary to implement interrupts during programming or erasing procedures on a 0.5 µm device, then the AUTO mode should be used. This removes the responsibility of clearing the EEPGM bit from the application and allows the device to automatically clear it, thereby making it easier to determine after an interrupt whether the EEPROM is ready to be accessed.

As in the standard mode, if there is not a requisite to access the EEPROM array within the interrupt service routine, then when the application returns from the routine, it can continue normal execution of the program/erase sequence. If the array does need to be accessed, then required delay time is easily achieved by continuing to poll the EEPGM bit until it is cleared by the internal state machine. Once done, EELAT must be cleared, then the routine can properly access the array. As mentioned previously, since the program/erase sequence has essentially been completed within the routine, program flow should return to the end of the normal algorithm sequence.

If it is necessary to have quick access to data within the EEPROM array on which a program or erase operation is being performed, there are two possible methods which can be used to avoid having to wait for the state machine to disable the internal charge pump.

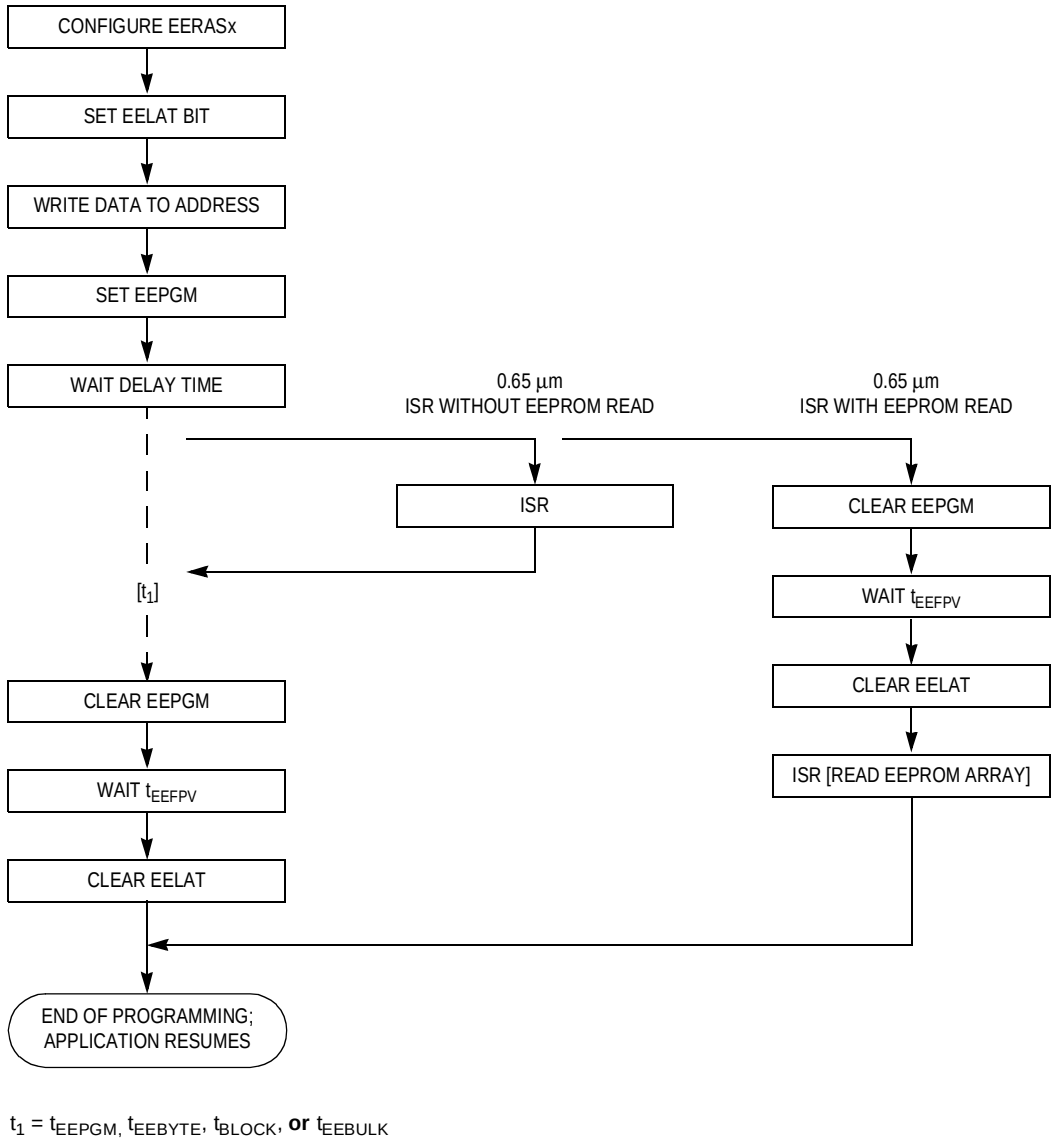
The first method is to buffer the needed data in RAM. The data can be buffered upon initialization of the device so that it is available at any time during operation. The internal charge pump has no affect on accessing RAM, hence the data could immediately be read after an interrupt from a programming/erasing algorithm without having to wait for the charge pump to be disabled and discharged.

The second method is only possible with the larger devices, MC68HC908AS60A and MC68HC908AZ60A. It entails utilizing the other EEPROM array found on the device. Unlike the other 0.5 µm devices, the two above mentioned microcontrollers have two EEPROM arrays instead of one. Each array is completely independent of the other. They each have their own charge pump and separate control registers. This allows access to one array while the other is being programmed or erased. Hence, the second method that will allow quick access to necessary data is to store the data in the opposite array from the one that is being programmed or erased. This could be achieved by designating one array to contain all fixed data — any data that is constant throughout the operation of the application. Then use the second array to contain all dynamic data — any data that is changed or stored during the operation of the application. In this way, when the application is interrupted from a program or erase operation to the array used for storing dynamic data, it will be able to access data from the other array containing operational data without having to wait for the internal charge pump to become disabled.

Conclusion

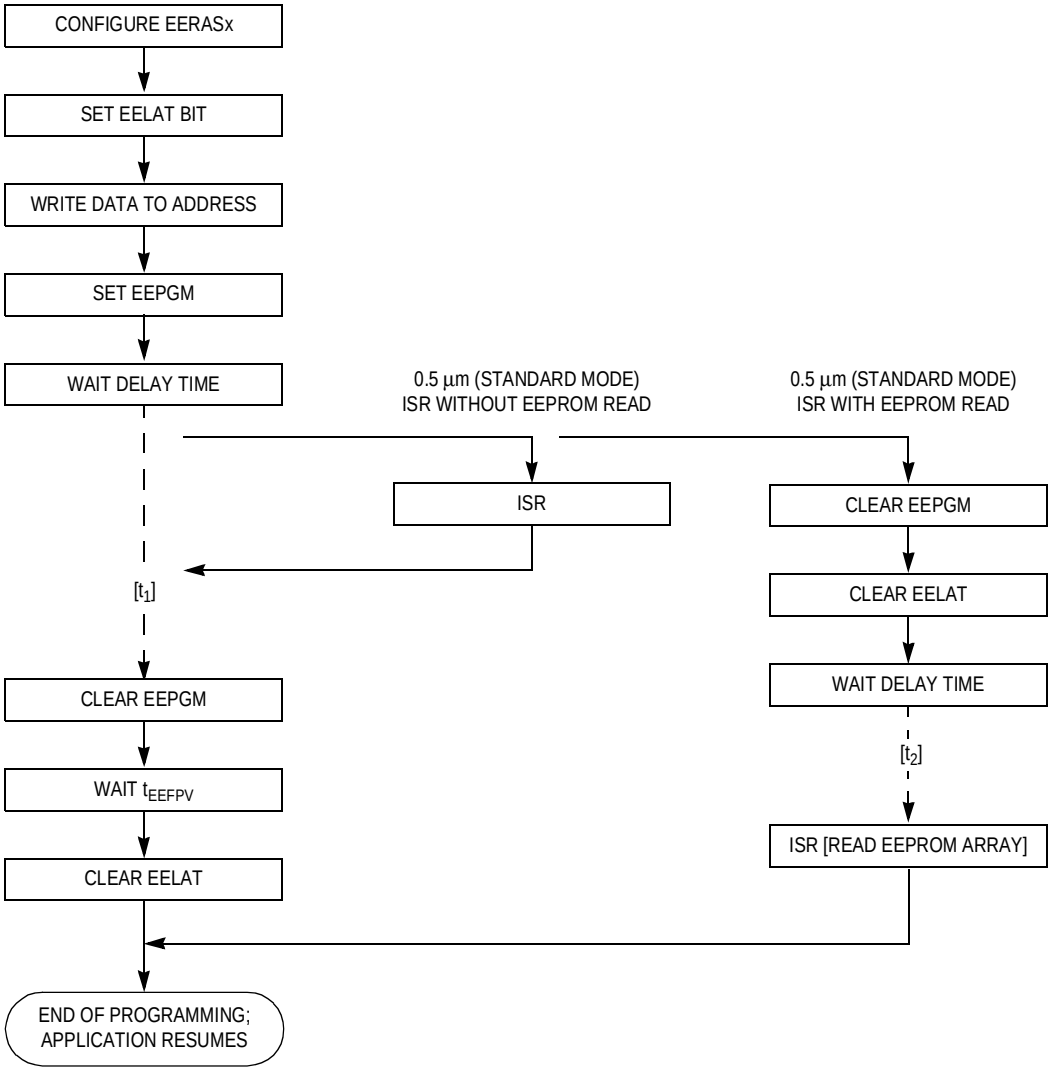
In order to avoid illegal extension of critical timing delays, which can cause improper programming or erase operations, it is normally recommended that interrupts be blocked during EEPROM programming operations. In applications where it is necessary to keep interrupts enabled, care must be taken in the way EEPROM operations are halted. In addition, there are certain periods of time in a programming sequence when the EEPROM cannot be accessed. The delay from when an operation is interrupted and halted, until the EEPROM array can be accessed, depends on whether a 0.65 μm or a 0.5 μm device is used and what operation was being executed when the operation was halted.

Flow Diagrams



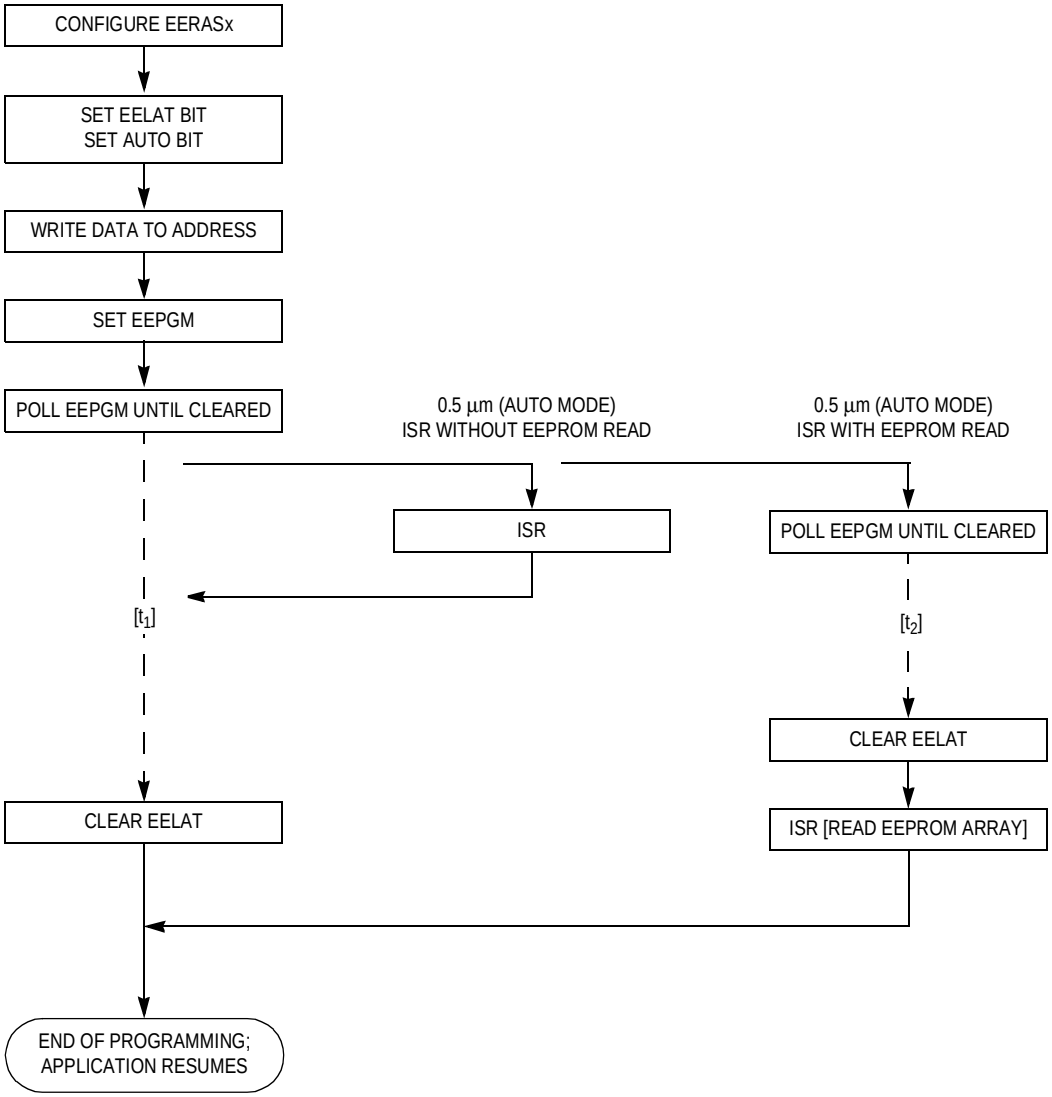
$t_1 = t_{EEPGM}$, t_{EEBYTE}, t_{BLOCK}, **or** t_{EEBULK}

Figure 1. Program/Erase Flow — 0.65 µm



$t_1 = t_{EEPGM}, t_{EEBYTE}, t_{BLOCK}$, **or** t_{EEBULK}
t_2 = Remainder of t_1 **or** EEPROM programming/erasing maximum time to AUTO bit set

Figure 2. Program/Erase Flow — 0.5 µm (Standard Mode)



t_1 = EEPROM programming/erasing time to AUTO bit set
t_2 = Remainder of t_1

Figure 3. Program/Erase Flow — 0.5 µm (AUTO Mode)

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