Order this document
by EB183/D
REV. 1.0

Motorola Semiconductor Engineering Bulletin

EB183

Erasing and Programming the FLASH EEPROM on the MC68HC912B32

By Matt Ruff
M68HC11 and M68HC12 Applications
Austin, Texas

Introduction

This document outlines basic routines to program the FLASH EEPROM through the background debug mode interface (BDM) using a Motorola serial debug interface (SDIL) and the SDBUG12 (version 2.15) software from P & E Microcomputer Systems, Inc.

One of the reasons the MC68HC912B32 device is so useful is that it contains 32 Kbytes of embedded FLASH EEPROM. This module serves as electrically programmable and erasable, non-volatile ROM emulation memory, allowing for storage of program code which must be executed frequently, must execute at high speeds, or which might need to be upgraded in the field at a later time. Commonly used code segments, such as standard subroutines or even operating systems, as well as static data tables can be stored in the FLASH EEPROM.

© Motorola, Inc., 1998

FLASH EEPROM Control Block

The FLASH EEPROM is controlled by a 4-byte register block, which is located at address $00F4 upon reset. Within this block are four single-byte registers:

- Lock control register (FEELCK)
- Module configuration register (FEEMCR)
- Module test register (FEETST)
- Module control register (FEECTL)

For more detail on these control registers, refer to Section 7.4 FLASH EEPROM Registers of the *MC68HC912B32 Technical Summary* (Motorola order number MC68HC912B32TS/D). The sequence of how to use these registers is covered later in this document.

FLASH EEPROM Lock Control Register

The FEELCK register (located at $00F4) contains only the LOCK bit (bit 0), which allows or prevents writing to the FEEMCR register. This must be cleared in order to change the FEEMCR. Note that it is cleared out of reset.

Address: $00F4

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	LOCK
Reset:	0	0	0	0	0	0	0	0

Figure 1. FLASH EEPROM Lock Control Register (FEELCK)

FLASH EEPROM Module Configuration Register

The FEEMCR register (located at $00F5) contains only the BOOTP bit (bit 0), which protects the 2-Kbyte boot block (1 Kbyte in early mask sets G86W or G75R) located at $7800–$7FFF or $F800–$FFFF, depending on the mapped location of the FLASH array at power-up. This bit must be cleared, after the FEELCK (LOCK bit) is cleared, in order to write or erase the boot block.

Address: $00F5

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	BOOTP
Reset:	0	0	0	0	0	0	0	1

Figure 2. FLASH EEPROM Module Configuration Register (FEEMCR)

FLASH EEPROM Module Test Register

The FEETST register (located at $00F6) has no effect and always reads 0 in normal modes of operation.

Address: $00F6

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 3. FLASH EEPROM Module Test Register (FEETST)

FLASH EEPROM Control Register

The FEECTL register (located at $00F7) controls the actual programming and erasing of the FLASH EEPROM. In this register, five bits are used to control the FLASH. All bits are 0 upon reset.

Address: $00F7

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	FEESWAI	SVFP	ERAS	LAT	ENPE
Reset:	0	0	0	0	0	0	0	0

Figure 4. FLASH EEPROM Control Register (FEECTL)

FEESWAI

FEESWAI (bit 4) controls the behavior of the FLASH EEPROM clock while in wait mode.

SVFP

SVFP (bit 3), the V_{FP} status bit, is set when V_{FP} is at or above normal programming voltage levels; clear otherwise (read only)

ERAS

ERAS (bit 2), when set, configures the array for erasure.

LAT

LAT (bit 1), when set, enables the programming latches.

ENPE

ENPE (bit 0), when set, applies the programming/erase voltage to the array.

Hardware Configuration

Setting up the Debugging Hardware

Since programming the FLASH EEPROM takes a finite amount of time and is dependent on a reliable programming voltage from an exterior source, it is difficult to tell if the procedure worked immediately. To simplify the debugging of the process, try using a few hardware tricks that are listed in this bulletin. For debugging hardware, simply use an light-emitting diode (LED) connected to a port pin on the MC68HC912B32 device that blinks when an error is encountered, as well as an LED attached to another port pin which lights when the process is complete. For use with the code in this bulletin, connect a red LED to PA0, with a 1-KΩ current limiting resistor to indicate errors. In like manner, connect a green LED to PA1 to indicate that the process has completed.

Setting up the M68HC12B32EVB

Be sure to connect your V_{FP} source to W8 on the M68HC12B32EVB with the proper polarity. The ground should be connected to the pin closest to the edge of the board and farthest from the microcontroller. This applies V_{FP} to the board, but the jumper on W7 actually transfers V_{FP} to the V_{FP} pin (pin 69) on the microcontroller. When a jumper is placed on the left two pins of W7 (with board facing so that silk screening can be read), V_{FP} is connected to the chip. When the jumper is moved to the right, so that the center pin and right pin are shorted, then V_{DD} is applied to the chip. This is the default location, and a jumper should always be located here to maintain the voltage on the V_{FP} pin on the MCU at V_{DD} when programming and erasing are not occurring.

NOTE: *V_{FP} should be 11.4–11.8 volts for mask sets 1H91F and 3H91F. For all other masks, use 11.4–12.6 volts (12 volts ±5%).*

Software Considerations

Using SDBUG12 to manipulate the FLASH EEPROM requires some special considerations. First, a few bugs in some versions of the software can cause some confusion when manipulating the FLASH memory array. The memory display windows sometimes do not refresh properly, especially when manually erasing the array by manipulating the control registers using the mm command. Once the erase voltage has been applied by modifying the FEECTL register, the display often shows all of the odd addresses as one value and all the even addresses as another value. To fix this problem, issue a reset command from the SDBUG12 command prompt to force SDBUG12 to refresh all of its display windows from the microcontroller once the part comes out of reset.

The routines that follow were tested with version 2.15 of SDBUG12 running on a Windows NT workstation in a DOS window. The problem described in the preceding paragraph did not appear when executing these routines. SDBUG12 displayed the proper values for the FLASH array when the routines were allowed to run to completion. The code segments included here can both be loaded into the RAM of the part, using the load command in SDBUG12. Notice that the entry point of the program routine is $80A and the entry point of the erase routine is $90A. Once loaded into RAM, g 80A will begin the programming process or g 90A will begin the erase process.

NOTE: *Once the FLASH array has been erased or programmed, reloading the DBUG12 monitor into the FLASH array is necessary if you wish to use it, as manipulation of the array will destroy the monitor program code. This can be accomplished with the bootloader in the boot block of the part or by using Prog12s.*

If using the M68EVB912B32 evaluation board, refer to Appendix E of the *Evaluation Board User's Manual* (Motorola order number 68EVB912B32UM/D), for further information on how to reload the monitor program into the device using the on-board bootloader.

Erasing the FLASH Array

This code segment follows the recommended procedure for erasing the FLASH array. Following the code is a flowchart which outlines this same procedure. The general idea is to apply the erase voltage to the FLASH module within the chip (by setting the ENPE bit), set the erase flag, write to any location in the array, then check to make sure the entire array is erased. If the whole array is erased, then the number of times the erase voltage was applied to get this erasure will have been preserved in the Nep variable. The erase voltage is then pulsed that many times again to ensure that the array remains erased. This is 100% erase margin.

NOTE: *In the following code, the STEP labels refer to Section 7.7 of the MC68HC912B32 Technical Summary (Motorola order number MC68HC912B32TS/D).*

```
;---------------------------------------------------------
;---  FLASH EEPROM erase routine                       ---
;---                                                   ---
;---  Rev. 1.0  April 16,1998                          ---
;---     Changed to 100ms delay for tepulse            ---
;---  Written November 6, 1997                         ---
;---                                                   ---
;---------------------------------------------------------

;----  Equates   ----------------------
FEELCK  EQU     $F4
FEEMCR  EQU     $F5
FEECTL  EQU     $F7
FEESTART EQU    $8000           ;FLASH Start address
FEEEND  EQU     $FFFF           ;FLASH End address
MAXNep  EQU     !5              ;5  pulses maximum
;----  Equates   ----------------------

        ORG     $0900
Nep     DS      1               ;Number of programming pulses applied
MARGINF DS      1               ;Programming margin flag
ERASED  DS      1               ;Array Erased Flag

        ORG     $90A
START   LDS     #$B00           ;(Turn on Vfp supply to board here)
        LDX     #$0000
        CLR     Nep             ;Clear number of pulses
        CLR     MARGINF         ;Clear margin flag
        CLR     ERASED          ;Clear erased flag
```

```
STEP2    MOVB    #$06,FEECTL     ;Set ERAS and set LAT in FEECTL
         BRCLR   FEECTL,$08,ERROR ;If Vfp not present, output an error
         LDAB    #$FF
STEP3    STAB    FEESTART,X      ;Write data to a valid Flash address
STEP4    BSET    FEECTL,$01      ;Apply erase voltage (Set ENPE)
STEP5    JSR     dly_100ms       ;Delay time for erase pulse (Tepulse)
STEP6    BCLR    FEECTL,$01      ;Remove erase voltage (Clear ENPE)
STEP7    JSR     dly_10ms        ;Delay for high voltage turn off (Tverase)
         LDAA    #$01
         CMPA    MARGINF         ;Is margin flag set??
         BNE     NOFLAG          ;If not, go bump counter and check data

YESFLAG  DEC     Nep             ;Decrement Nep
         LDAA    #$00
         CMPA    Nep             ;Is Nep=0?
         BNE     STEP4           ;If not, go to Step 4
         JSR     READARRY        ;Verify entire array is erased
         LDAA    #$01
         CMPA    ERASED          ;Is the array erased?
         BEQ     ERROR           ;Erase failed, output an error
STEP10   BCLR    FEECTL,$02      ;Clear LAT in FEECTL
         BRA     DONE            ;If so, quit.

NOFLAG   INC     Nep             ;Increment number of erase pulses applied
         BSR     READARRY        ;Verify entire array is erased
         LDAA    #$00
         CMPA    ERASED          ;Is it erased?
         BEQ     SETMARF         ;If so, set margin flag
         LDAB    Nep
         CMPB    #MAXNep         ;Have we applied max number of pulses?
         BLS     STEP4           ;If not, continue erasing
         BSR     ERROR           ;If so, we have a problem

SETMARF  INC     MARGINF         ;Set Margin Flag
         BRA     STEP4

DONE     MOVB    #$00,$0000      ;Clear Port A
         MOVB    #$FF,$0002      ;Set DDRA to outputs
         MOVB    #$02,$0000      ;Turn on PA1 to indicate complete
         BRA     DONE            ;(Turn off Vfp)
```

```
;-------------------------------------------------
;----   Read and Verify Erase subroutine     ----
;-------------------------------------------------
READARRY LDY    #$FFFF
        LDX     #FEESTART
LOOP    CPY     0,X             ;Is this word erased?
        BNE     EXITverf        ;If not, leave without setting flag
        CPX     #FEEEND         ;Are we at the end of the array?
        BEQ     EXITverf
        INX                     ;Go to the next address
  INX
        BRA     LOOP
        INC     ERASED          ;Set erased flag
EXITverf RTS

;-------------------------------------------------
;---------     Error Subroutine     -------------
;-------------------------------------------------
ERROR:  MOVB    #$00,$0000      ;Clear Port A
        MOVB    #$FF,$0002      ;Set DDRA to outputs
BLINK   MOVB    #$01,$0000      ;Turn PA0 on for error output
        BSR     dly_100ms
        BSR     dly_100ms
        BSR     dly_100ms
        MOVB    #$00,$0000      ;Turn PA0 off
        BSR     dly_100ms
        BSR     dly_100ms
        BSR     dly_100ms
        BRA     BLINK           ; Repeat ad nauseum....

;-------------------------------------------------------------
;---------     Delay Subroutines  (8MHz e clock) -------------
;-------------------------------------------------------------
dly_100ms: LDY     #$000A       ; Delay for 100ms (8MHz E clock)
DLOOP10: DEY
        BSR     dly_10ms
        BNE     DLOOP10
        RTS

dly_10ms:  LDD     #$3E7E        ; Delay for 10ms (8MHz E clock)
DLOOP:  SUBD    #1
        BNE     DLOOP
        RTS

        END
```

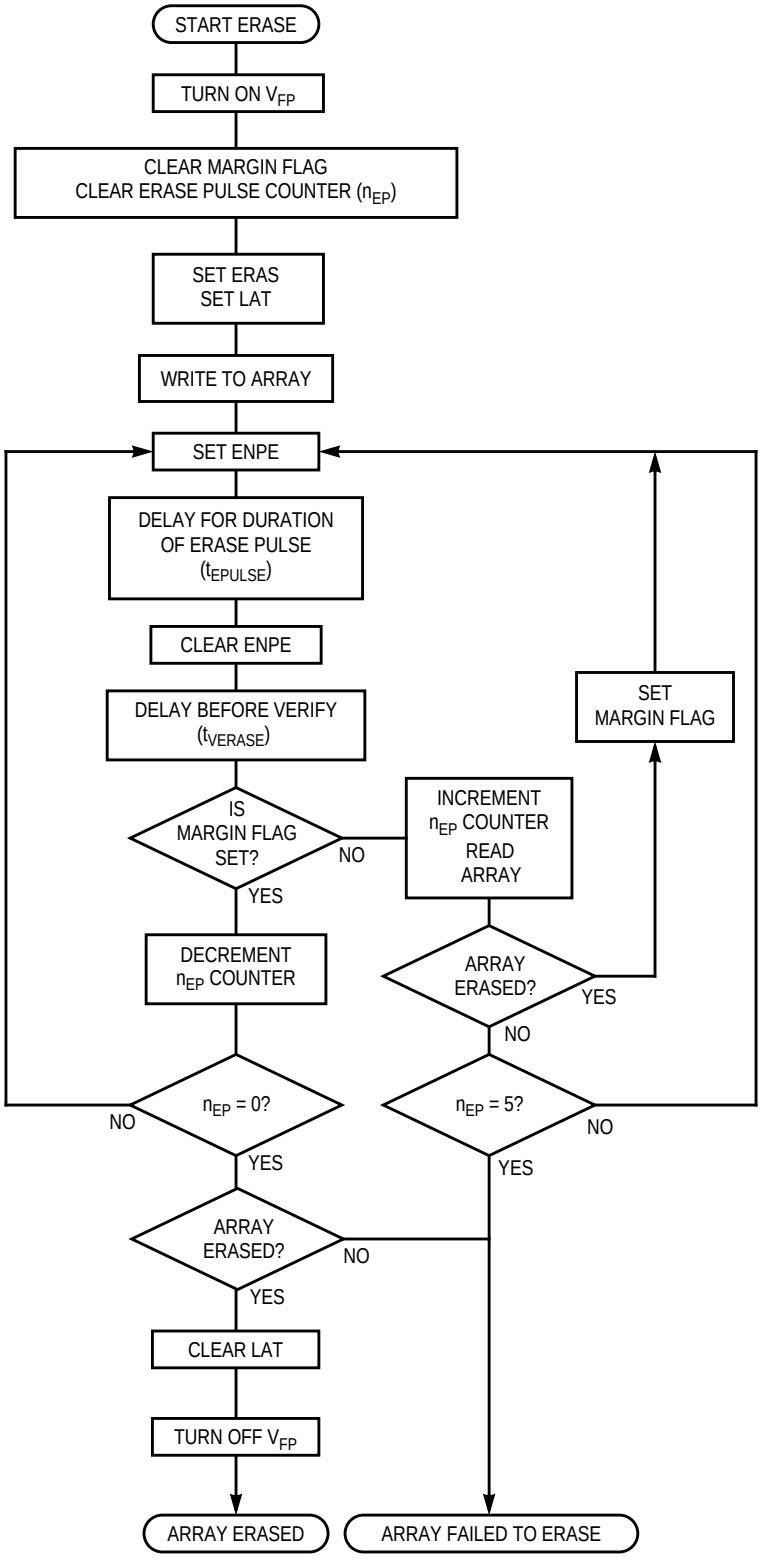

Figure 5. Erase Sequence Flow

Programming the FLASH Array

The following code segment follows the recommended procedure for programming the FLASH array. Following the code is a flowchart which outlines this same procedure. The general idea is to apply the programming voltage to the FLASH module within the chip (by setting the ENPE bit), set the programming latches, write the desired byte/word to the location in the array, then check to make sure the location is programmed properly. If the data is correct, then the number of times the programming voltage was applied to get this byte programmed will have been preserved in the N_{PP} variable. The programming voltage is then pulsed that many times again to ensure that the byte/word remains programmed. This is 100% programming margin. This whole process is repeated for each byte/word to be programmed. The code segment below simply copies a string of characters from RAM and stores it at the beginning of the FLASH array.

NOTE: *In the following code, the STEP labels refer to Section 7.6 of the MC68HC912B32 Technical Summary (Motorola order number MC68HC912B32TS/D).*

```
;---------------------------------------------------------
;---  FLASH EEPROM program routine                     ---
;---                                                   ---
;---  Rev. 1.0 - April 23,1998                         ---
;---    Fixed Tppulse = 25µs and Tvprog = 10µs         ---
;---  Written November 6, 1997                         ---
;---                                                   ---
;---------------------------------------------------------


;-------------- Equates -----------------
FEELCK  EQU     $F4
FEEMCR  EQU     $F5
FEECTL  EQU     $F7
FEESTART EQU    $8000           ;FLASH Start address
FEEEND  EQU     $FFFF           ;FLASH End address
MAXNpp  EQU     !50             ;50 pulses maximum

        ORG     $0800
Npp     DS      1               ;Number of programming pulses applied
MARGINF DS      1               ;Programming margin flag

        ORG     $80A
```

```
START   LDS     #$B00            ;(Turn on your Vfp power supply to board)
        BRCLR   FEECTL,$08,ERROR ;If Vfp not present, output an error
        LDX     #$0000

LOOP    CLR     Npp              ;Clear number of pulses
        CLR     MARGINF          ;Clear margin flag
STEP2   MOVB    #$02,FEECTL      ;Clear ERAS and set LAT in FEECTL
        LDAB    DATA,X
STEP3   STAB    FEESTART,X       ;Write data to address
STEP4   BSET    FEECTL,$01       ;Apply programming voltage (Set ENPE)
STEP5   JSR     dly_22us         ;Delay time for prog pulse (Tppulse)
STEP6   BCLR    FEECTL,$01       ;Remove programming voltage (Clear ENPE)
STEP7   JSR     dly_10us         ;Delay for high voltage turn off (Tvprog)
        LDAA    #$01
        CMPA    MARGINF          ;Is margin flag set??
        BNE     NOFLAG           ;If not, go bump counter and check data

YESFLAG DEC     Npp              ;Decrement Npp
        LDAA    #$00
        CMPA    Npp              ;Is Npp=0?
        BNE     STEP4            ;If not, go to Step 4
STEP9   LDAA    FEESTART,X       ;Read FEEPROM location to verify programming
        CMPA    DATA,X           ;Is it the same as the byte to be programmed?
        BNE     ERROR            ;Programming failed, output an error
STEP10  BCLR    FEECTL,$02       ;Clear LAT in FEECTL
        INX
        CMPA    #$00             ;Check to see if we're done
        BNE     LOOP             ;If not, go back to start!
        BRA     DONE             ;If so, quit.

NOFLAG  INC     Npp              ;Increment number of prog pulses applied
        LDAA    FEESTART,X       ;Read FEEPROM location to verify programming
        CMPA    DATA,X           ;Is it the same as the byte to be programmed?
        BEQ     SETMARF          ;If so, set the margin flag
        LDAB    Npp
        CMPB    #MAXNpp          ;Have we applied max number of pulses?
        BLS     STEP4            ;If not, continue programming
        BSR     ERROR            ;If so, we have a problem

SETMARF INC     MARGINF          ;Set Margin Flag
        BRA     STEP4

DONE    MOVB    #$00,$0000       ;Clear Port A
        MOVB    #$FF,$0002       ;Set DDRA to outputs
        MOVB    #$02,$0000       ;Turn on PA1 to indicate complete
        BRA     *                ;(Turn off Vfp supply - programming complete)
```

```
;-----------------------------------------------
;---------     Error Subroutine     -------------
;-----------------------------------------------
ERROR:  MOVB    #$00,$0000      ;Clear Port A
        MOVB    #$FF,$0002      ;Set DDRA to outputs
BLINK   MOVB    #$01,$0000      ;Turn PA0 on for error output
        BSR     dly_100ms
        BSR     dly_100ms
        BSR     dly_100ms
        MOVB    #$00,$0000      ;Turn PA0 off
        BSR     dly_100ms
        BSR     dly_100ms
        BSR     dly_100ms
        BRA     BLINK           ; Repeat ad nauseum....


;-------------------------------------------------------------
;---------     Delay Subroutines  (8MHz e clock) -------------
;-------------------------------------------------------------
dly_100ms: LDY     #$000A       ; Delay for 100ms
DLOOP10: DEY
        BSR     dly_10ms
        BNE     DLOOP10
        RTS

dly_10ms:  LDD     #$3E7E       ; Delay for 10ms
DLOOP:  SUBD    #1
        BNE     DLOOP
        RTS


dly_22us:  LDD     #$0023       ; Delay for almost 22µs
d_22u   SUBD    #1
        BNE     d_22u
        RTS

dly_10us:  LDD     #$0010       ; Delay for 10µs
d_10u   SUBD    #1
        BNE     d_10u
        RTS


DATA    FCB     "Motorola Microcontrollers"
        FCB     $00

        END
```

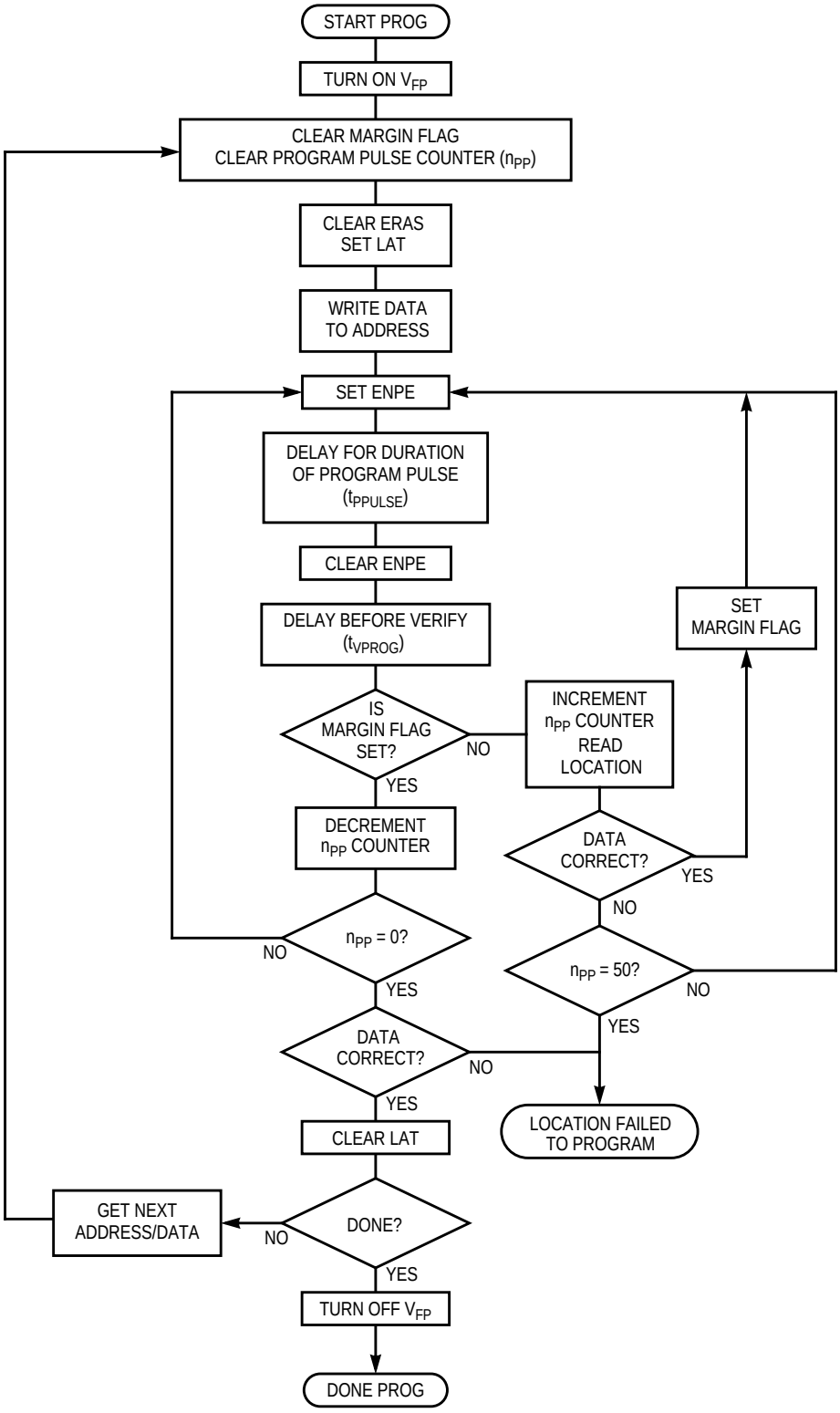

Figure 6. Program Sequence Flow

Conclusion

This bulletin gives an overview of the basics of erasing and programming the FLASH array on the MC68HC912B32 microcontroller. Knowing these basics, it is easy to progress to writing a bootloader, designing a field programming unit, or anything else needed to manipulate the FLASH memory.

For an example of a serial bootloader for this microcontroller, refer to *Serial Bootloader for Reprogramming the MC68HC912B32 FLASH EEPROM* (Motorola order number AN1718/D).

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.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217, 1-800-441-2447 or 1-303-675-2140. Customer Focus Center, 1-800-521-6274

JAPAN: Nippon Motorola Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinigawa-Ku, Tokyo, Japan. 03-5487-8488

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd., 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/; TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848

HOME PAGE: http://motorola.com/sps/

Mfax is a trademark of Motorola, Inc.

© Motorola, Inc., 1998