Motorola Semiconductor Engineering Bulletin

EB301

Programming EEPROM on the MC68HC811E2 during Program Execution

**By Brian Scott Crow
Austin, Texas**

Introduction

The MC68HC811E2 microcontroller (MCU) has the largest EEPROM array in the MC68HC11 Family of microcontrollers. This 2-K array of EEPROM can be used for both program code and data values. In addition to the 2-K array of EEPROM, the device has 256 bytes of RAM. While some users may choose to use RAM for program variables during execution, others may want to store data and variables in the non-volatile EEPROM, so that data will still be valid after a power-off and power-on sequence.

The Problem

For expanded mode applications of this part, Motorola publishes programming and erasing algorithms in the technical data book for the MC68HC811E2, Motorola document order number MC68HC811E2/D. However, in single-chip mode applications, the only memory resources available for program code are the internal RAM and EEPROM. Code which programs or erases internal EEPROM cannot reside in internal EEPROM. The only other memory resource is RAM, which cannot

MOTOROLA

contain program code immediately after power-up because of its volatile nature. Therefore, code in EEPROM must copy the programming algorithm into RAM, load parameters into the registers or in global RAM variables, and then jump (JSR) to the programming algorithm copy in RAM.

At the end of the subroutine, execution resumes in EEPROM, and the data is stored until it is erased using the same procedure.

Users often wonder why code executing out of internal EEPROM cannot write to or erase EEPROM, especially users of the MC68HC811E2, where the only other memory resource in single-chip applications is the internal RAM.

Programming and erasing of EEPROM requires applying a high voltage to the EEPROM array. In MC68HC11 devices, this programming voltage is developed with an on-chip charge pump.

High voltage is applied to the EEPROM array only when the EEPGM bit in the PPROG register is set to logic 1. While high voltage is applied to the EEPROM array, the read circuitry in the EEPROM array is disabled. This is why programs that try to use the programming algorithm while executing code from EEPROM "hang" or "get lost." After turning on the high voltage, the processor executes a read cycle to fetch the opcode for the next instruction. Because the internal EEPROM read circuitry is disabled, the processor can no longer fetch opcodes.

EB301

The Solution

The code listing provided here shows how to overcome the problem of programming EEPROM in single-chip applications. The code will work in expanded mode applications as well; however, expanded mode applications simply need to put programming and erase algorithms in an external memory resource to overcome this problem.

The code segment has three major portions:

- The main routine, which initializes the stack pointer, clears the EEPROM block protection register (BPROT), and calls the other routines

- The cpy2ram routine, which copies the algorithms from EEPROM to RAM

- The program, erase, and delay algorithms for EEPROM

The code in the main routine illustrates four important points:

- First, every program must initialize the stack pointer. This must happen before any subroutine calls, interrupts, or pushes and pulls from the stack occur.

- Next, the BPROT register (or single bits in the BPROT register) must be cleared within the first 64 clock cycles after reset, so that the EEPROM array is not protected from programming and erasure.

- If an application separates data and program space, then only the bits which protect data space should be cleared. The main routine calls cpy2ram to ensure that the algorithms are in place every time out of reset.

- Finally, the main routine shows how to set up the parameters and call the program and erase routines. Parameter passing schemes may use registers, specified RAM locations, or the stack.

This code segment only programs one byte of data to the EEPROM. Additional instructions could be substituted for lines 19 through 23, for instance, to fetch data from the serial communications interface (SCI). Each individual byte could then be programmed into EEPROM by calling the bytprgram routine.

EB301

Freescale Semiconductor, Inc.

Sample Code

```
EE811.ASM    Assembled with IASM    02/15/1993    16:05


0000            1        ramhi   equ   $00ff    ;last address of internal RAM
0000            2        regbas  equ   $1000    ;register base address
000             3        bprot   equ   $0035    ;offset from base for BPROT
0000            4        pprog   equ   $003b    ;offset from base for PROG
0000            5        eestrt  equ   $f800    ;first address EEPROM on
                                                ;811E2
                6
0000            7        rsetvec equ   $fffe    ;address of reset vector
0000            8        ersram  equ   $0000    ;the address in RAM that
                9                               ;routines will be copied to
                10
FFFE            11       org           rsetvec  ;reset points to main
FFFEF800        12       fdb                 main
                13
F800            14       org           eestrt   ;main begins at start of
                                                ;eeprom
F800 8E00FF     15       main  lds     #ramhi   ;set the stack pointer
F803 CE1000     16            ldx      #regbas  ;used for index X access
F806 6F35       17            clr      bprot,x  ;clear reg for eeprom prog
                                                ;and erase
F808 BDF81C     18            jsr      cpy2ram  ;copy routines to ram on the
                                                ;811e2
F80B 18CEF800   19            ldy      #eestrt  ;load address parameter into
                                                ;index Y
F80F 9D00       20            jsr      ersram   ;for sample byte erase
F811 18CEF800   21            ldy      #eestrt  ;then program it to $00
F815 8600       22            ldaa     #$00     ;get data parameter into Acc A
F817 BD0020     23            jsr      bytprg   ;REMEMBER: you must jump to
                24                              ;in RAM not those in EEPROM
F81A 20FE       25       bra *                  ;infinite self-loop to ends
                                                ;example
                26
                27                              ;this subroutine will copy the
                                                ;program, byte erase
                28                              ;and delay 10 ms routines into
                                                ;RAM at address $0100
                29                              ;this will let the user
                                                ;jumpsubroutine to these
                                                ;routines
                30                              ;which will allow proper
                                                ;programming of the EEPROM
                31                              ;on the 811E2
                32
F81C 3C         33       cpy2ram pshx
F81D 183C       34            pshy
F81F CEF837     35            ldx   #byteras
F822 18CE0000   36            ldy   #ersram
```

```
F826 E600      37          cpyloop ldab 0,x
F828 18E700    38                  stab 0,y
F82B 08        39                  inx
F82C 1808      40                  iny
F82E 8CF86E    41                  cpx  #endprg
F831 26F3      42                  bne  cpyloop
F833 1838      43                  puly
F835 38        44                  pulx
F836 39        45                  rts
               46
               47                              ;this subroutine expects the
                                               ;address of the byte to be
               48                              ;erased to be passed in the y
                                               ;register, and pprog is
               49                              ;declared in an equate file
F837 3C        50          byteras pshx        ;make reentrant and
F838 37        51                  pshb        ;save registers
F839 CE1000    52                  ldx  #regbas ;used indexed x access
F83C C616      53                  ldab #$16   ;eelat=1, byte erase
F83E E73B      54                  stab pprog,x
F840 18E700    55                  stab 0,y    ;access byte pointed to by Y
F843 C617      56                  ldab #$17   ;turn on eepgm
F845 E73B      57                  stab pprog,x
F847 8D05      58                  bsr  delay10 ;relative addressing is
               59                              ;location independent!
F849 6F3B      60                  clr  pprog,x ;turn off eepgm and eelat
F84B 33        61                  pulb
F84C 38        62                  pulx        ;restore registers
F84D 39        63                  rts
F84E           64                  enderas equ *
               65
               66                              ;this subroutine delays the mcu
                                               ;for 10 milliseconds
F84E 3C        67                  delay10 pshx ;make reentrant and
               68                              ;save registers
F84F CE0D05    69                  ldx    #$0d05 ;constant for 10mS at 2MHz
                                               ;E-clock
F852 09        70                  loop10  dex
F853 26FD      71                  bne loop10  ;relative addressing is
               72                              ;position independent
F855 38        73                  pulx        ;restore registers
F856 39        74                  rts
F857           75                  enddly  equ *
               76
               77                              ;this subroutine programs the
                                               ;byte whose address is in
               78                              ;register y with the value
                                               ;passed in register a
F857 3C        79                  bytprg pshx ;make reentrant and
F858 37        80                  pshb        ;save registers
F859 CE1000    81                  ldx  #regbas ;for indexed x access
```

EB301

```
F85C C602      82              ldab   #$02       ;eelat=1, eepgm=0
F85E E73B      83              stab   pprog,x
F860 18A700    84              staa   0,y        ;write data to address
               85                                ;pointed to by Y
F863 C603      86              ldab   #$03       ;eepgm=1
F865 E73B      87              stab   pprog,x
F867 8DE5      88              bsr    delay10    ;relative addressing is
               89                                ;position independent
F869 6F3B      90              clr    pprog,x    ;eelat=eepgm=0
F86B 33        91              pulb
F86C 38        92              pulx              ;restore registers
F86D 39        93              rts
               95
F86E           96         dlyram equdelay10-byteras  ;compute
                                                 ;addresses for the routines
               97                                ;that will be in RAM
F86E           98         bytprg  equ bytprg-byteras
               99
               100


Symbol Table

BPROT                       0035
BYTERAS                     F837
BYTERSRAM                   0000
BYTPRG                      F857
BYTPRGRAM                   0020
CPY2RAM                     F81C
CPYLOOP                     F826
DELAY10                     F84E
DLYRAM                      0017
EESTRT                      F800
ENDDLY                      F857
ENDERAS                     F84E
ENDPRG                      F86E
LOOP10                      F852
MAIN                        F800
PPROG                       003B
RAMHI                       00FF
REGBAS                      1000
RESETVECTOR                 FFFE
```

This code was assembled using IASM11 from P&E Microsystems on an IBM-compatible PC.

Conclusion

The MC68HC811E2 is a unique member of the MC68HC11 Family because of its large EEPROM array. This memory resource is available to users for program space, as well as for data and variables. Single-chip application designers can run into trouble trying to use the EEPROM for variables if they do not realize that internal EEPROM programming or erase algorithms cannot be executed from internal EEPROM.

Once the solution discussed in this engineering bulletin is implemented, users will be able to successfully use the MC68HC811E2 to its full potential in control applications.

EB301

Freescale Semiconductor, Inc.

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: Motorola Japan Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan, 03-5487-8488

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd., Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, New Territories, 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., 1999

MOTOROLA

EB301/D

For More Information On This Product,
Go to: www.freescale.com