Order this document
by AN1010/D

MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

AN1010

MC68HC11 EEPROM Programming from a Personal Computer

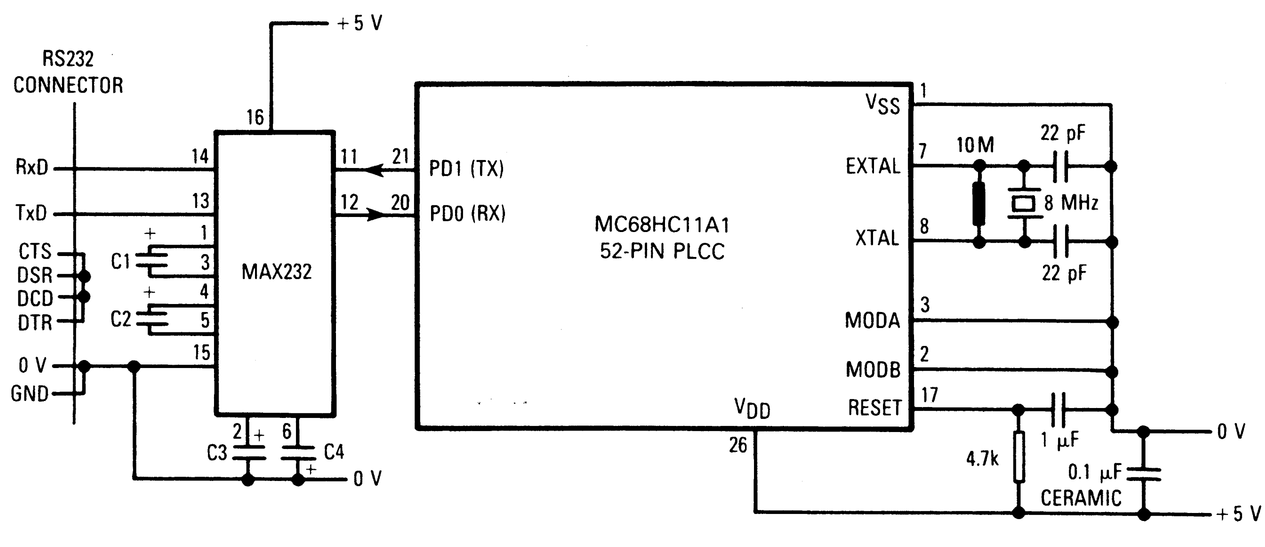
This application note describes a simple and reliable method of programming either the MC68HC11's internal EEPROM, or EEPROM connected to the MCU's external bus. The data to be programmed is downloaded from any standard personal computer (PC) fitted with a serial communications port. In addition to the programming procedure, the software incorporates the facility to verify the contents of the MCU's internal or external memory against code held on a PC disc. Both program and verify options use data supplied in S record format, which is downloaded from the PC to the MC68HC11 using the RS232 protocol supported by the MCU's SCI port.

The minimum MCU configuration required to program the MC68HC11's internal EEPROM is shown in Figure 1. This consists only of the MCU, an RS232 level shifting circuit, plus an 8 MHz crystal and a few passive components.

To initiate the download, the PC is connected to the MC68HC11 SCI transmit and receive lines via a level shifter. The circuit of Figure 1 uses a Maxim MAX232 to eliminate the need for additional ±12 V supplies. The MCU's special bootstrap mode is invoked by applying a logic zero to the MODA and MODB pins, followed by a hardware RESET.

Removing the RESET condition causes the MCU to start execution of its bootloader program, located in internal ROM, between addresses $BF40 and $BFFF. In normal single-chip or expanded modes, the boot ROM is not accessible, and reads from these memory locations will result respectively in irrelevant data or external memory fetches.

An additional consequence of bootstrap operation is that all vectors are relocated to the boot ROM area. With the exception of the RESET vector, which points to the start of the boot ROM, the remaining interrupt vectors all point to an uninitialized jump table in RAM. Three bytes are reserved for each entry in the jump table, to allow for an extended jump instruction. Tables 1 and 2 detail the memory map of the bootstrap vectors and an example RAM jump table.



C1, C2, C3, C4 — 22 µF 25 V Aluminium or Tantalum

NOTE: To improve reliability of the MCU, all its unused inputs should be connected to V_{SS} or V_{DD}

Figure 1. MC68HC11 Bootstrap Mode Connection to RS232 Line

MOTOROLA

Table 1. Bootstrap Vector Assignments

Boot ROM		
Address	**Vector**	**Description**
BFFE	BF40	Bootstrap Reset
BFFC	00FD	Clock Monitor
BFFA	00FA	COP Fail
BFF8	00F7	Illegal Opcode
BFF6	00F4	SWI
BFF4	00F1	XIRQ
BFF2	00EE	IRQ
BFF0	00EB	Real Time Interrupt
BFEE	00E8	Timer Input Capture 1
BFEC	00E5	Timer Input Capture 2
BFEA	00E2	Timer Input Capture 3
BFE8	00DF	Timer Output Compare 1
BFE6	00DC	Timer Output Compare 2
BFE4	00D9	Timer Output Compare 3
BFE2	00D6	Timer Output Compare 4
BFE0	00D3	Timer Output Compare 5
BFDE	00D0	Timer Overflow
BFDC	00CD	Pulse Accumulator Overflow
BFDA	00CA	Pulse Accumulator Input Edge
BFD8	00C7	SPI
BFD6	00C4	SCI

Table 2. RAM Jump Table

Internal RAM	
Address	**Typical Instruction**
00FD	JMP CLKMON
00FA	JMP COPFL
....etc	

Note that, before any interrupts are enabled in bootstrap mode, it is the software designer's responsibility to initialize all appropriate entries in the jump table.

As this application note does not make use of the MC68HC11's interrupt system, the jump table is not set up.

The bootstrap program continues by initializing the SCI transmitter and receiver to 7812 baud and proceeds to examine the state of the NOSEC bit in the CONFIG register. If this is at logic zero (security enabled) the bootloader will erase the entire contents of internal EEPROM and also the CONFIG register.

This feature is particularly useful for security conscious applications, where the internal EEPROM contains information of a proprietary or confidential nature. If the NOSEC bit is at logic one, then the erasing sequence is not carried out.

Note also that erasing the CONFIG register disables the security feature.

The bootstrap program then issues a break condition on the SCI transmit line, and waits for the reception of the first byte. In this application, no use is made of the break transmitted by the SCI.

At this point, it is necessary to initiate the PC S record downloader program, called EELOAD.BAS (written in BASIC). It will display a header message, and prompt the user for the number of the COM channel (either one or two) which is connected to the MC68HC11. A listing of EELOAD.BAS is given at the back of this application note.

The PC-resident program will now configure the appropriate COM channel to 1200 baud, one stop bit, no parity, and download the binary file EEPROGIX.BOO from the PC to the MC68HC11.

The MC68HC11's bootloader will automatically detect the fact that the first incoming character is received at a different baud rate, and change its SCI rate to 1200 baud.

It will then proceed to load the binary file into all 256 RAM locations and then jump to address $0000 (i.e., the first RAM location).

EEPROGIX.BOO consists of the MC68HC11 executable code shown in the source listing at the back of this application note, with the addition of $FF at the head of the file, and $00 appended up to the 256th byte. This program is designed to receive S records from the PC and program the data fields into the appropriate EEPROM memory locations.

A point to note is that the initial $FF byte in EEPROGIX.BOO is only used to detect the baud rate of the PC, and is not echoed back, while the remaining 256 bytes are echoed by the MC68HC11's SCI transmitter. However, during download of EEPROGIX.BOO, the PC does not detect the echo, as this feature is unnecessary at this stage.

Once the newly downloaded S record programmer starts execution in the MC68HC11, it configures the SCI to 9600 baud, then waits for a control character from the PC. This character will determine the operating mode of the S record programmer. The options available are shown in Table 3. Note that these programming utilities can be used to load and verify external RAM as well as external EEPROM.

Table 3. S Record Downloader Operating Mode Options

Control Character	Operating Mode
X	Program External EEPROM/RAM
I	Program Internal EEPROM
V	Verify Internal or External EEPROM/RAM

If the S record programmer has been downloaded successfully, the PC resident program will now —

1. Request whether the downloaded data must be echoed to the screen.
2. Prompt the user for the required operating mode.
3. Request the name of the S record file to be downloaded from the PC.

Once the download starts, every character in the S record file is immediately echoed back to the PC. This ensures synchronism between the PC and the MC68HC11, and at the same time, removes some of the overhead associated with the EEPROM programming delay time. It also removes the need for a hardware handshake.

VERIFY OPTION

If a verify error occurs, the actual stored byte value is returned to the PC, where it is displayed with a preceding colon delimiter. In this way, EEPROM data and address faults can be quickly identified by inspection. At the end of the verify download, the total number of errors is displayed.

INTERNAL OR EXTERNAL OPTION

If a programming error occurs in either internal or external programming mode, i.e., if the read back data after programming does not correspond to the expected data, the MC68HC11-resident software will hang up. This condition is detected by the PC-resident program, which will then abort the download and display an error message. This same error message is displayed if a fault or incorrect connection exists on the serial link between the PC and MC68HC11.

There is one exception to this operation. It stems from the fact that changes to the MC68HC11's CONFIG register can only be detected after a subsequent hardware RESET. If the CONFIG register address ($103F) is detected, then the CONFIG register is not read directly after programming. This prevents premature termination of the download.

To allow programming of the CONFIG register in all mask set versions of the MC68HC11A series, and to permit expanded mode operation, the MCU resident program switches from bootstrap mode to special test mode, by setting the MDA bit (bit 5) in the HPRIO register (address $103C).

If the user wishes to maintain operation in bootstrap mode, (to verify internal ROM code, for instance), then the 'BSET HPRIO,X,#MDA' instruction on the 8th line of program code in EEPROGIX.ASC should be removed, and the program reassembled.

PROGRAMMING INTERNAL EEPROM

The techniques for programming internal and external EEPROM are quite different.

With internal EEPROM, it is first generally necessary to erase the required byte (erased state is $FF), and follow with a write of data to the same address.

The internal programming sequence involves accessing the PPROG register (address $103B) to latch the EEPROM address and data buses for the duration that the programming voltage is applied. Also, the programming time delay must be implemented or initiated by software. In this application, a software timing loop is used, but one of the internal MC68HC11 timer functions could equally well be used to provide the time delay.

Figures 2 and 3 show the flowcharts of the internal EEPROM erase and write sequences.

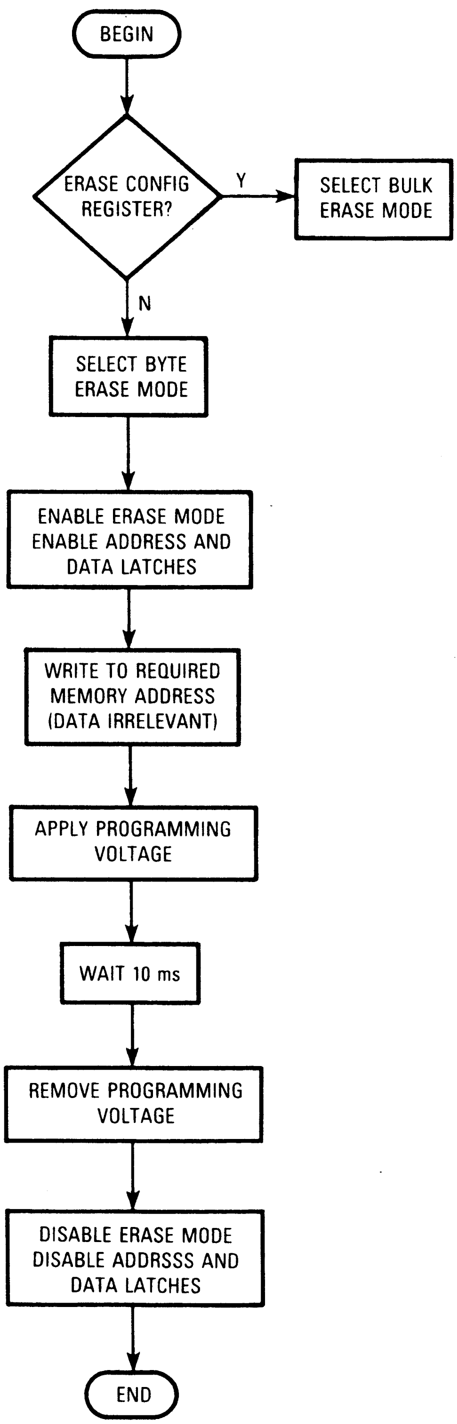


Figure 2. Internal EEPROM Erase Sequence

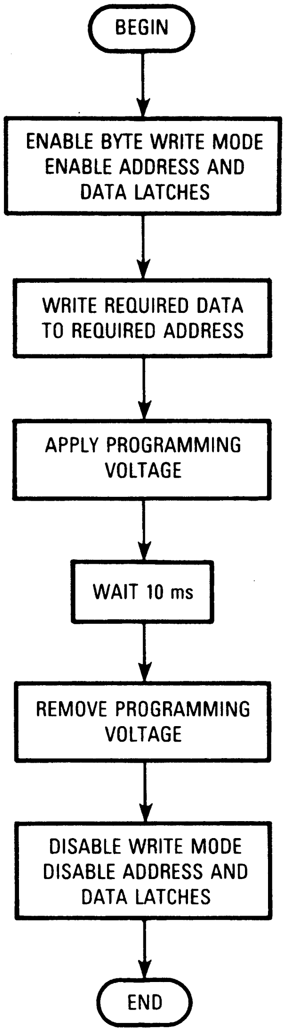


Figure 3. Internal EEPROM Write Sequence

PROGRAMMING EXTERNAL EEPROM

Figure 4 shows the hardware needed to interface the MC68HC11 to an external 2864 EEPROM, which provides a total of 8K bytes of reprogrammable memory. The addition of the MC68HC24 gives a minimal component count implementation of a circuit which accurately emulates the MC68HC11A8 single-chip MCU. The added benefit of using the 2864 is that the software designer's program and/or data can be modified without removing the emulator from the target system. This can be particularly useful in applications where the emulator may be enclosed in a confined space or in an environmental chamber.

To program the 2864 from the PC, the external operating mode option (X) must be selected from the EELOAD menu.

Programming the 2864 involves fewer operations than are needed for internal EEPROM, as the former has no equivalent of the PPROG control register. In addition, the erase sequence and delay time are handled automatically by the 2864 on-chip logic.

A data polling technique is used to determine the end of the programming delay time. This involves examining the most significant bit of the data programmed, by reading from the address just written to, until the data becomes true. (During the programming delay time, the MS bit will read as the complement of the expected data).

This means that the same software algorithm can be used to download code or data to external RAM as well as external EEPROM.

EMULATOR ADDRESS DECODING

The emulator circuit in Figure 4 shows the MC68HC11's address line A13 connected to pin 26 of the 2864. Though this pin is actually unused by the 2864, its inclusion permits the replacement of the 2864 with a 27128 16K byte EEPROM memory.

An important outcome of this is that, when a 2864 is used, the memory range \$C000–\$DFFF is mapped over the normally used 8K byte range of \$E000–\$FFFF. In practice, this should never pose a problem. When a 27128 memory is used, its full 16K byte address range of \$C000–\$FFFF is available to the MCU.

Included in the S record programmer, irrespective of the selected programming mode, is a feature to force program execution at the address specified in the S9, S record address field, provided the address is not \$0000.

Figure 5 shows the general format of S record files.

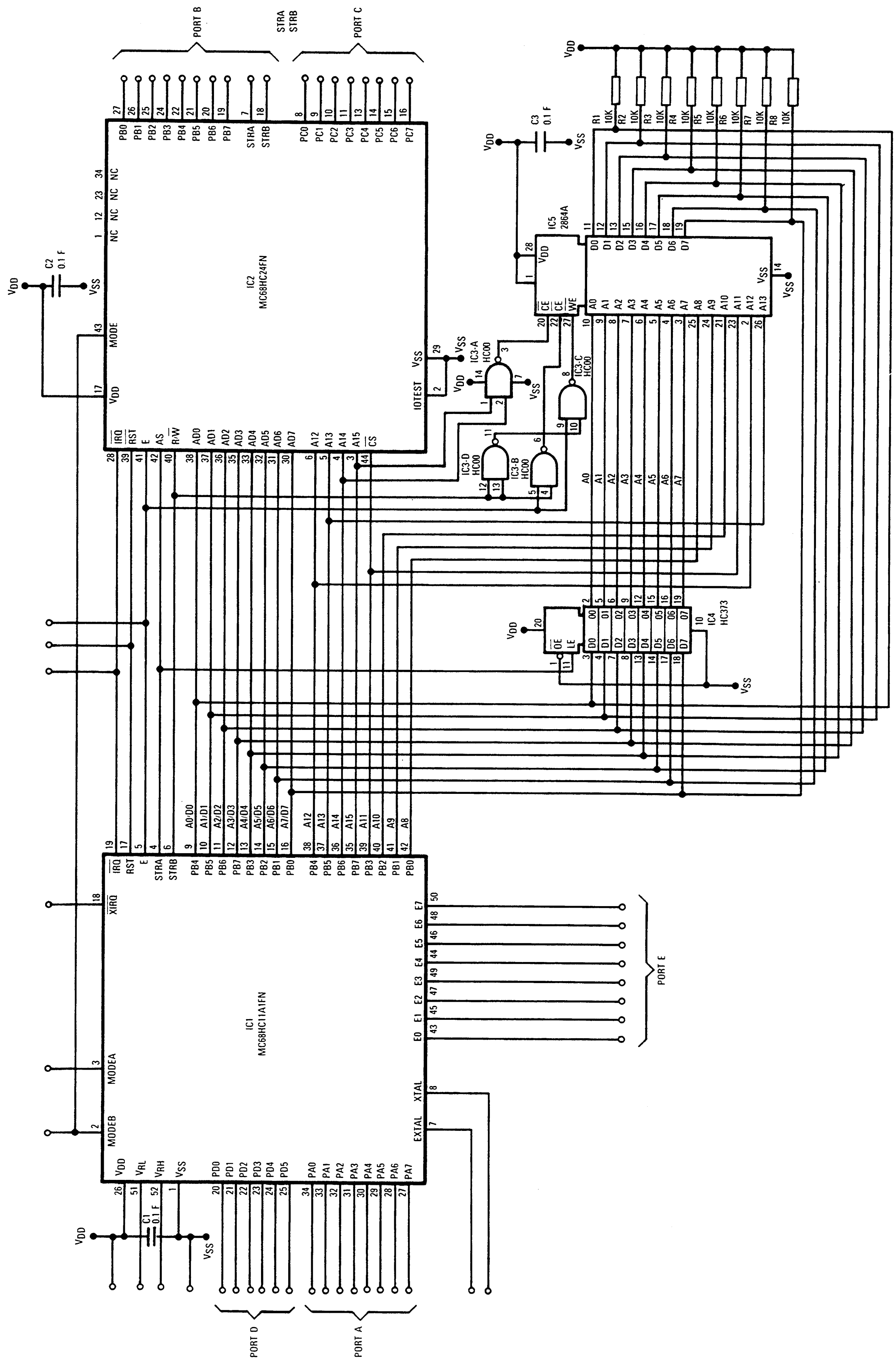
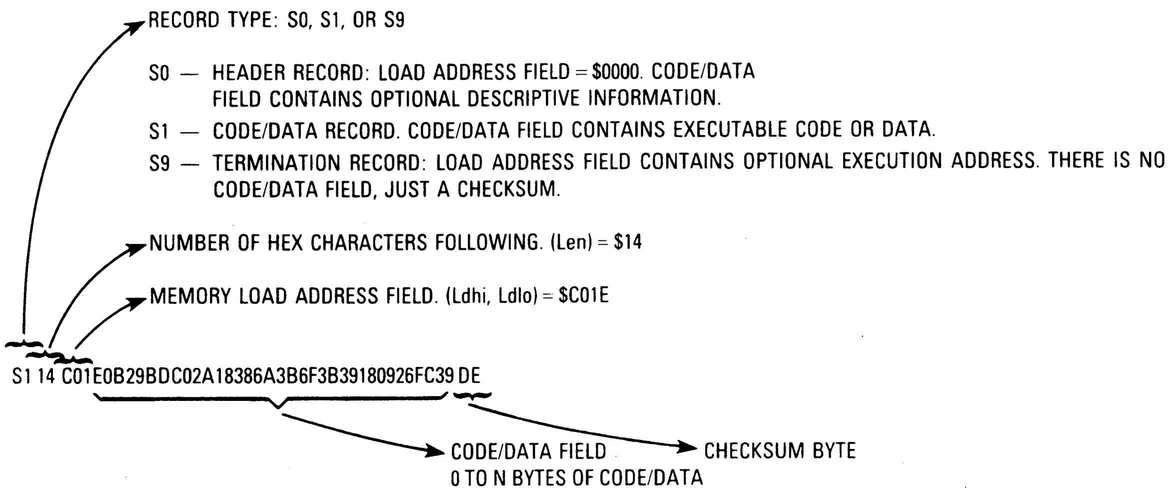


Figure 4. MC68HC11A8 Emulator Using 2864 EEPROM



APART FROM THE LETTER S AT THE START, ALL CHARACTERS IN THE RECORDS ARE HEXADECIMAL DIGITS REPRESENTED IN ASCII FORMAT.

CHECKSUM ALGORITHM: LSB OF $\overline{\left[\text{Len} + \text{Ldhi} + \text{Ldlo} + \sum_{k=0}^{n} \text{byte}_k \right]}$

NOTE: The S-record programmer in this application ignores the checksum byte.

Figure 5. S-Record Format

```
10  '  *******  EELOAD.BAS  20/3/87    Version 1.0 ********'
20 ' Written by R.Soja, Motorola East Kilbride'
30 ' Motorola Copyright 1987'
40 ' This program downloads S record file to the MC68HC11 through special'
50 ' bootstrap program, designed to program either internal or external '
60 ' EEPROM in the 68HC11's memory map'
70 ' The loader can also verify memory against an S record file.'
80 ' Downloaded data is optionally echoed on terminal.'
90 ' =================================
100 CR$=CHR$(13)
110 MIN$=CHR$(32)
120 MAX$=CHR$(127)
130 ERM$="Can't find "
140 LOADER$="EEPROGIX.BOO"
150 CLRLN$=SPACE$(80)
160 VER$="1.0": 'Version number of EELOAD'
170 ERRTOT%=0:  'Number of errors found by verify operation'
180 CLS
190 PRINT "  <<<<<<<<          EELOAD Version ";VER$;"                    >>>>>>>>>"
200 PRINT "  <<<<<<<< 68HC11 Internal/External EEPROM loader/verifier >>>>>>>>>"
210 PRINT
220 PRINT "==>  Before continuing, ensure 68HC11 is in bootstrap mode,"
230 PRINT "     RESET is off, and COM1 or COM2 is connected to the SCI"
240 PRINT
250 ' First make sure loader program is available'
260 ON ERROR GOTO 880
270 OPEN LOADER$ FOR INPUT AS #2
280 CLOSE #2
290 ON ERROR GOTO 0
300 CHAN$="0"
310 ROW=CSRLIN: 'Store current line number'
320 WHILE CHAN$<>"1" AND CHAN$<>"2"
330   GOSUB 1070
340   LINE INPUT "Enter COM channel number (1/2):",CHAN$
350 WEND
360 CM$="COM"+CHAN$
370 ' Now set baud rate to 1200 and load EEPROG through boot loader'
380 ' by executing DOS MODE and COPY commands'
390 SHELL "MODE "+CM$+":1200,N,8,1"
400 SHELL "COPY "+LOADER$+" "+CM$
401 GOSUB 1070
402 FOR I%=1 TO 4:PRINT CLRLN$;:NEXT I%:PRINT: 'Clear DOS commands from screen'
410 ECHO$=" "
420 WHILE ECHO$<>"Y" AND ECHO$<>"N"
430   GOSUB 1070
440   LINE INPUT "Do you want echo to screen (Y/N):",ECHO$
450 WEND
470 ROW=CSRLIN: 'Store current line number'
480 EEOPT$=" ": 'Initialise option char'
490 WHILE EEOPT$<>"X" AND EEOPT$<>"I" AND EEOPT$<>"V"
500   GOSUB 1070
510   LINE INPUT "Select Internal,eXternal or Verify EEPROM option (I/X/V):",EEOPT$
520 WEND
530 OPT$="Verify"
540 IF EEOPT$="I" THEN OPT$="Internal"
550 IF EEOPT$="X" THEN OPT$="External"
560 ROW=CSRLIN: 'Store current line position in case of file error'
570 RXERR=0:    'Initialise number of RX errors allowed'
580 ON ERROR GOTO 910
590 GOSUB 1070
```

```
600 IF OPT$="Verify" THEN INPUT "Enter filename to verify: ",F$ ELSE INPUT "Enter filename to download:",F$
610 CLOSE
620 OPEN F$ FOR INPUT AS #2
630 ON ERROR GOTO 0
640 'COM1 or 2 connected to SCI on HC11'
650  OPEN CM$+":9600,N,8,1" AS #1
660 'Establish contact with HC11 by sending CR char & waiting for echo'
670 ON ERROR GOTO 860: 'Clear potential RX error'
680 PRINT #1,CR$;
690 GOSUB 990: 'Read char into B$'
700 'Transmit Internal,External or Verify EEPROM option char to 68HC11'
710 PRINT #1,EEOPT$;:GOSUB 990:              'No echo to screen'
720 ON ERROR GOTO 930
730 PRINT "Starting download of <";F$;"> to: ";OPT$;" Eeprom"
732 IF ECHO$="Y" THEN E%=1 ELSE E%=0
734 IF EEOPT$="V" THEN V%=1 ELSE V%=0
740 WHILE NOT EOF(2)
750  INPUT #2,S$
751  L%=LEN(S$)
752  FOR I%=1 TO L%
760      PRINT #1,MID$(S$,I%,1);:GOSUB 990:IF E% THEN PRINT B$;
770      IF V% THEN GOSUB 1030:IF C$<>"" THEN PRINT ":";HEX$(ASC(C$));
785  NEXT I%
787  IF E% THEN PRINT
790 WEND
795 PRINT
800 PRINT "Download Complete"
810 IF V% THEN PRINT ERRTOT%;" error(s) found"
820 CLOSE #2
830 SYSTEM
840 END
850 ' -----------------'
860 IF RXERR>5 THEN 940 ELSE RXERR=RXERR+1:RESUME 610
870 ' -----------------'
880 PRINT:PRINT ERM$;LOADER$:PRINT "Program aborted"
890 GOTO 830
900 ' -----------------'
910 PRINT ERM$;F$;SPACE$(40)
920 RESUME 580
930 '-----------------'
940 PRINT:PRINT "Communication breakdown: Download aborted"
950 GOTO 820
960 '--------------------'
970 '--SUB waits for received character, with time limit'
980 '--    returns with char in B$, or aborts if time limit exceeded'
990 TO%=0:WHILE LOC(1)=0:IF TO%>100 THEN 940 ELSE TO%=TO%+1:WEND
1000 B$=INPUT$(1,#1):RETURN
1010 '--------------------'
1020 '--SUB waits for received character, with time limit'
1025 '--    returns with char in C$, or null in C$ if time limit exceeded'
1030 TO%=0:C$="":WHILE LOC(1)=0 AND TO%<1:TO%=TO%+1:WEND
1040 IF LOC(1)>0 THEN C$=INPUT$(1,#1):ERRTOT%=ERRTOT%+1
1050 RETURN
1060 '--------------------'
1070 '--SUB Clear line '
1080 LOCATE ROW,1,1:PRINT CLRLN$
1090 LOCATE ROW,1,1:RETURN
1100 '--------------------'
```

```
**************************************************************
*               EEPROGIX.ASC 19/3/87     Revision 1.0        *
*                                                            *
* Written by R.Soja, Motorola, East Kilbride                 *
* Motorola Copyright 1987.                                   *
*                                                            *
*        This program loads S records from the host to       *
* either a 2864 external EEPROM on the 68HC11 external bus,  *
* or to the 68HC11's internal EEPROM. It can also be used    *
* verify memory contents against an S record file or just    *
* load RAM located on the 68HC11's external bus.             *
* Each byte loaded is echoed back to the host.               *
* When programming a 2864, data polling is used to detect    *
* completion of the programming cycle.                       *
* As the host software always waits for the echo before      *
* downloading the next byte, host transmission is suspended  *
* during the data polling period.                            *
* Because the serial communication rate (~1mS/byte) is       *
* slower than the 2864 internal timer timeout rate (~300uS)  *
* page write mode cannot be used. This means that data       *
* polling is active on each byte written to the EEPROM,      *
* after an initial delay of approx 500uS.                    *
*                                                            *
* When the internal EEPROM is programmed,instead of data     *
* polling, each byte is verified after programming.          *
* In this case, the 500uS delay is not required and is       *
* bypassed.                                                  *
* If a failure occurs, the program effectively hangs up. It  *
* is the responsibility of the host downloader program to    *
* detect this condition and take remedial action.            *
* The BASIC program EELOAD just displays a 'Communication    *
* breakdown' message, and terminates the program.            *
*                                                            *
* When used in the verify mode, apart from the normal echo   *
* back of each character, all differences between memory     *
* and S record data are also sent back to the host.          *
* The host software must be capable of detecting this, and   *
* perform the action required.                               *
* The BASIC loader program EELOAD simply displays the        *
* returned erroneous byte adjacent to the expected byte,     *
* separated by a colon.                                      *
*                                                            *
* Before receiving the S records, a code byte is received    *
* from the host. i.e.:                                       *
*                      ASCII 'X' for external EEPROM         *
*                      ASCII 'I' for internal EEPROM         *
*                      ASCII 'V' for verify EEPROM           *
*                                                            *
* This program is designed to be used with the BASIC EELOAD  *
* program.                                                   *
* Data transfer is through the SCI, configured for 8 data    *
* bits, 9600 baud.                                           *
*                                                            *
          PAGE
```

```
A                  * Constants
A       0080       TDRE      EQU     $80
A       0020       RDRF      EQU     $20
A       0020       MDA       EQU     $20
A       0040       SMOD      EQU     $40
A       0D05       mS10      EQU     10000/3          10mS delay with 8MHz xtal.
A       00A6       uS500     EQU     500/3            500uS delay.
A                  *
A                  * Registers
A       002B       BAUD      EQU     $2B
A       002C       SCCR1     EQU     $2C
A       002D       SCCR2     EQU     $2D
A       002E       SCSR      EQU     $2E
A       002F       SCDR      EQU     $2F
A       003B       PPROG     EQU     $3B
A       003C       HPRIO     EQU     $3C
A       103F       CONFIG    EQU     $103F
A                  *
A                  * Variables. Note: They overwrite initialisation code!!!!
A       0000                 ORG     $0
P 0000  0001       EEOPT     RMB     1
P 0001  0001       MASK      RMB     1
P 0002  0001       TEMP      RMB     1
P 0003  0001       LASTBYTE  RMB     1
A                  *
A                  * Program
A       0000                 ORG     $0
A 0000  8E00FF               LDS     #$FF
A 0003  CE1000               LDX     #$1000           Offset for control registers.
A 0006  6F2C                 CLR     SCCR1,X          Initialise SCI for 8 data bits, 9600 baud
A 0008  CC300C               LDD     #$300C
A 000B  A72B                 STAA    BAUD,X
A 000D  E72D                 STAB    SCCR2,X
A 000F  1C3C20               BSET    HPRIO,X,#MDA     Force Special Test mode first,
A                  *=>> MAINTAIN SPECIAL TEST MODE TO ALLOW B96D CONFIG REGISTER PROGRAMMING <<==
A                  *         BCLR HPRIO,X,#SMOD  and then expanded mode. (From Bootstrap mode)
A 0012  9F00       ReadOpt   STS     <EEOPT           Default to internal EEPROM: EEOPT=0; MASK=$FF;
A 0014  8D7C                 BSR     READC            Then check control byte for external or internal
A 0016  C149                 CMPB    #'I'             EEPROM selection.
A 0018  2714                 BEQ     LOAD
A 001A  C158                 CMPB    #'X'             If external EEPROM requested
A 001C  2609                 BNE     OptVerf
A 001E  7C0000               INC     EEOPT            then change option to 1
A 0021  8680                 LDAA    #$80
A 0023  9701                 STAA    <MASK            and select mask for data polling mode.
A 0025  2007                 BRA     LOAD
A                  *
A 0027  C156       OptVerf   CMPB    #'V'             If not verify then
A 0029  26E7                 BNE     ReadOpt          get next character else
A 002B  7A0000               DEC     EEOPT            make EEOPT flag negative.
A                  *
A       002E       LOAD      EQU     *
A 002E  8D62                 BSR     READC
A 0030  C153                 CMPB    #'S              Wait until S1 or S9 received,
A 0032  26FA                 BNE     LOAD             discarding checksum of previous S1 record.
A 0034  8D5C                 BSR     READC
A 0036  C131                 CMPB    #'1
A 0038  2719                 BEQ     LOAD1
```

```
A 003A C139                CMPB    #'9
A 003C 26F0                BNE     LOAD
A 003E 8D5F                BSR     RDBYTE       Complete reading S9 record before terminating
A 0040 17                  TBA
A 0041 8002                SUBA    #2           # of bytes to read including checksum.
A 0043 8D6B                BSR     GETADR       Get execution address in Y
A 0045 8D58      LOAD9     BSR     RDBYTE       Now discard remaining bytes,
A 0047 4A                  DECA                 including checksum.
A 0048 26FB                BNE     LOAD9
A 004A 188C0000            CPY     #0           If execution address =0 then
A 004E 27FE                BEQ     *            hang up else
A 0050 186E00              JMP     ,Y           jump to it!
A                *
A      0053      LOAD1     EQU     *
A 0053 8D4A                BSR     RDBYTE       Read byte count of S1 record into ACCB
A 0055 17                  TBA                  and store in ACCA
A 0056 8003                SUBA    #3           Remove load address & checksum bytes from count
A 0058 8D56                BSR     GETADR       Get load address into X register.
A 005A 1809                DEY                  Adjust it for first time thru' LOAD2 loop.
A 005C 2017                BRA     LOAD1B
A                *
A 005E D600      LOAD1A    LDAB    EEOPT        Update CC register
A 0060 2B25                BMI     VERIFY       If not verifying EEPROM then
A 0062 2705                BEQ     DATAPOLL     If programming external EEPROM
A 0064 C6A6                LDAB    #uS500
A 0066 5A        WAIT1     DECB                 then wait 500uS max.
A 0067 26FD                BNE     WAIT1
A 0069 18E600    DATAPOLL  LDAB    ,Y           Now either wait for completion of programming
A 006C D803                EORB    <LASTBYTE    cycle by testing MS bit of last data written to
A 006E D401                ANDB    <MASK        memory or just verify internal programmed data.
A 0070 26F7                BNE     DATAPOLL
A 0072 4A        LOAD1E    DECA                 When all bytes done,
A 0073 27B9                BEQ     LOAD         get next S record (discarding checksum) else
A 0075 8D28      LOAD1B    BSR     RDBYTE       read next data byte into ACCB.
A 0077 1808                INY                  Advance to next load address
A 0079 7D0000              TST     EEOPT
A 007C 2B05                BMI     LOAD1D       If verifying, then don't program byte!
A 007E 2743                BEQ     PROG         If internal EEPROM option selected then program
A 0080 18E700              STAB    ,Y           else just store byte at address.
A 0083 D703      LOAD1D    STAB    <LASTBYTE    Save it for DATA POLLING operation.
A 0085 20D7                BRA     LOAD1A
A                *
A 0087 18E600    VERIFY    LDAB    ,Y           If programmed byte
A 008A D103                CMPB    <LASTBYTE    is correct then
A 008C 27E4                BEQ     LOAD1E       read next byte
A 008E 8D08                BSR     WRITEC       else send bad byte back to host
A 0090 20E0                BRA     LOAD1E       before reading next byte.
A                *
A      0092      READC     EQU     *            ACCA, X, Y regs unchanged by this routine.
A 0092 1F2E20FC            BRCLR   SCSR,X,#RDRF,*
A 0096 E62F                LDAB    SCDR,X       Read next char
A 0098 1F2E80FC  WRITEC    BRCLR   SCSR,X,#TDRE,*
A 009C E72F                STAB    SCDR,X       and echo it back to host.
A 009E 39                  RTS                  Return with char in ACCB.
A                *
A 009F 8DF1      RDBYTE    BSR     READC        1st read MS nibble
A 00A1 8D17                BSR     HEXBIN       Convert to binary
A 00A3 58                  LSLB                 and move to upper nibble
```

```
A 00A4 58                 LSLB
A 00A5 58                 LSLB
A 00A6 58                 LSLB
A 00A7 D702               STAB    <TEMP
A 00A9 8DE7               BSR     READC         Get ASCII char in ACCB
A 00AB 8D0D               BSR     HEXBIN
A 00AD DA02               ORAB    <TEMP
A 00AF 39                 RTS                   Return with byte in ACCB
A               *
A      00B0     GETADR    EQU     *
A 00B0 36                 PSHA                  Save byte counter
A 00B1 8DEC               BSR     RDBYTE        Read MS byte of address
A 00B3 17                 TBA                   and put it in MS byte of ACCD
A 00B4 8DE9               BSR     RDBYTE        Now read LS byte of address into LS byte of ACCD
A 00B6 188F               XGDY                  Put load address in Y
A 00B8 32                 PULA                  Restore byte counter
A 00B9 39                 RTS                   and return.
A               *
A      00BA     HEXBIN    EQU     *
A 00BA C139               CMPB    #'9           If ACCB>9 then assume its A-F
A 00BC 2302               BLS     HEXNUM
A 00BE CB09               ADDB    #9
A 00C0 C40F     HEXNUM    ANDB    #$F
A 00C2 39                 RTS
A               *
A      00C3     PROG      EQU     *
A 00C3 36                 PSHA                  Save ACCA.
A 00C4 8616               LDAA    #$16          Default to byte erase mode
A 00C6 188C103F           CPY     #CONFIG       If byte's address is CONFIG then use
A 00CA 2602               BNE     PROGA
A 00CC 8606               LDAA    #$06          bulk erase, to allow for A1 & A8 as well as A2.
A 00CE 8D10     PROGA     BSR     PROGRAM       Now erase byte, or entire memory + CONFIG.
A 00D0 8602               LDAA    #2
A 00D2 8D0C               BSR     PROGRAM       Now program byte.
A 00D4 188C103F           CPY     #CONFIG       If byte was CONFIG register then
A 00D8 2603               BNE     PROGX
A 00DA 18E600             LDAB    ,Y            load ACCB with old value,to prevent hangup later.
A 00DD 32       PROGX     PULA                  Restore ACCA
A 00DE 20A3               BRA     LOAD1D        and return to main bit.
A               *
A      00E0     PROGRAM   EQU     *
A 00E0 A73B               STAA    PPROG,X       Enable internal addr/data latches.
A 00E2 18E700             STAB    ,Y            Write to required address
A 00E5 6C3B               INC     PPROG,X       Enable internal programming voltage
A 00E7 3C                 PSHX
A 00E8 CE0D05             LDX     #mS10         and wait 10mS
A 00EB 09       WAIT2     DEX
A 00EC 26FD               BNE     WAIT2
A 00EE 38                 PULX
A 00EF 6A3B               DEC     PPROG,X       Disable internal programming voltage
A 00F1 6F3B               CLR     PPROG,X       Release internal addr/data latches
A 00F3 39                 RTS                   and return
A               *
A                         END
```

SYMBOL TABLE: Total Entries= 41

BAUD	002B	PROGA	00CE
CONFIG	103F	PROGRAM	00E0
DATAPOLL	0069	PROGX	00DD
EEOPT	0000	RDBYTE	009F
GETADR	00B0	RDRF	0020
HEXBIN	00BA	READC	0092
HEXNUM	00C0	ReadOpt	0012
HPRIO	003C	SCCR1	002C
LASTBYTE	0003	SCCR2	002D
LOAD	002E	SCDR	002F
LOAD1	0053	SCSR	002E
LOAD1A	005E	SMOD	0040
LOAD1B	0075	TDRE	0080
LOAD1D	0083	TEMP	0002
LOAD1E	0072	VERIFY	0087
LOAD9	0045	WAIT1	0066
MASK	0001	WAIT2	00EB
MDA	0020	WRITEC	0098
OptVerf	0027	mS10	0D05
PPROG	003B	uS500	00A6
PROG	00C3		

This page intentionally left blank.

This page intentionally left blank.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.

ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

MOTOROLA

AN1010/D

A23405 PRINTED IN USA (1993) MPS/POD YGACAA