

Motorola Semiconductor Engineering Bulletin

EB422

Enhanced M68HC11 Bootstrap Mode

By Steven McAslan
CSIC Development Systems
Motorola Ltd., East Kilbride, Scotland

Introduction

Motorola has enhanced the capability of the special bootstrap mode operation of many M68HC11 Family MCUs. The enhancements are possible by the addition of larger boot ROM memories, thereby expanding the functionality of the mode.

The bootstrap mode listings enclosed here cover a wider range of application possibilities than before and offer enhanced or modified operation over earlier offerings. Recent enhancements include the addition of autostart facilities for PLL (phase-locked loop) systems, enhanced security options, and embedding of PCbug11 talkers in the boot ROM.

This engineering bulletin describes the boot ROMs from these MCUs: MC68HC11ED0, MC68HC711EA9, MC68HC11PH8, MC68HC711PH8, secured MC68HC711E20, secured MC68HC711E32, and secured MC68HC11E32.



Special Bootstrap Mode

References [1] contains a complete description of the operation of the M68HC11 bootstrap mode. However, for completeness, a brief review follows.

Bootstrap mode differs from other M68HC11 modes because of its pre-programmed capability. When the M68HC11 enters bootstrap mode, it enables and then executes code from a special internal ROM (boot ROM). The standard function of this code is to allow the user to download a program into the internal RAM of the M68HC11 device and then execute the downloaded code. The MCU downloads the code through its SCI (serial communication interface) serial module and then passes control to the code in the RAM. In addition, the MCU allows users access to its internal memory and registers without restriction.

Since this internal ROM provides the functionality of the bootstrap mode, extensions or reductions in the size of this ROM can enhance or restrict the functionality of the mode. This bulletin describes some functionality enhancements (and restrictions) that Motorola has made to recent additions to the M68HC11 Family.

Motorola supplies a PC-based software package that communicates with M68HC11 devices in bootstrap mode. PCbug11 is available from your local Motorola supplier. See [3] in **References**.

MC68HC11ED0 Bootstrap Mode

The MC68HC11ED0 (ED0) is a low-end addition to the M68HC11 Family. It has only 512 bytes of RAM available internally and a limited pin count. Therefore, the ED0 has a boot ROM much reduced in size and functionality from the standard offering.

Unlike other M68HC11 boot ROMs, the ED0 does not allow the user to download code into all the internal RAM. In this case, the bootstrap mode only allows the user to load 256 bytes from address \$0100 to \$01FF. Additionally, unlike most other M68HC11s, the ED0 forces the user to download exactly 256 bytes before it can begin executing the code. A further restriction is that all the interrupt vectors point to a single address while in bootstrap mode, \$00FD. These restrictions limit the capability of the MCU to support system debugging.

Table 1 describes the relevant capability of the ED0 compared to a similar device, the MC68HC11D3 (D3).

Table 1. ED0 versus Boot ROM Capability

	MC68HC11D3	MC68HC11ED0
Total RAM size	512	512
Maximum download	512	256
Minimum download	1	256
Download start address	\$0000	\$0100
Unique vectors	21	1 at \$00FD

MC68HC(7)11EA9 Bootstrap Modes

These MCUs share many features with the MC68HC(7)11E9 (E9) devices and as described later, the EPROM version also contains some enhancements from the basic bootloader. A further complication is that the EA9 device has a PLL available for use with the MCU oscillator. Since this boot ROM does not contain any facility to allow it to control the PLL, the user should disable the PLL and connect a standard high frequency crystal or external clock (8 MHz). Disable the PLL by ensuring that the V_{DDSYN} voltage connects to ground.

On the MC68HC711EA9 (EA9), two additional subroutines are available to simplify the programming and verification of the internal EPROM array. The user may access these routines by downloading a jump instruction through the bootloader and then executing that jump.

[References](#) [1] describes these routines (PROGRAM and UPLOAD) in greater detail for the MC68HC11E9 MCU.

The ROM EA9 performs in a similar fashion to the E9 ROM version.

MC68HC(7)11PH8 Bootstrap Modes

Motorola added a much larger boot ROM to the MC68HC11PH8 (PH8) MCUs. This allows a greater range of flexibility for the user of these devices: automatic startup, embedded PCbug11 talker software, and enhanced automatic baud rate selection.

Both the ROM and EPROM versions of the PH8 have automatic startup on the PLL when in bootstrap mode. However, the user still has the option to use a high-frequency crystal; in this case, disable the PLL. The autostart option allows the user to specify the VCO control register (SYNR) value immediately after startup. The MCU reads the value on its port F and then starts the PLL using that value. If the value found on port F is \$FF, then the MCU writes \$CC to SYNR. Port F can have pullups present and so \$FF is the value that the CPU would read with no connections on the port. The multiplier \$CC gives an operating frequency of 8 MHz from a 38.4-kHz crystal. [Table 2](#) shows the options

available to the user on both the MC68HC11PH8 and the MC68HC711PH8.

Table 2. PLL Bootstrap Autostart Configuration

V _{DDSYN}	Port F	Crystal	MCU Frequency	Comment
GND	Don't care	8 MHz	8 MHz	PLL disabled; crystal frequency used
V _{DD}	NC ⁽¹⁾ or \$FF	X _{Value}	X _{Value} * 208	PLL active; crystal * 208 used
V _{DD}	Other F _{Set}	X _{Value}	F _{Set} * X _{Value}	PLL active; port F multiplier used

1. When pullups are enabled in CONFIG register

The MCU uses a 10-ms delay timed for a 38.4-kHz crystal before completing the PLL initialization. For a much higher value crystal, this delay may be insufficient to allow the PLL to settle.

A useful function of bootstrap mode is the ability to make the address and data buses active once bootstrap mode is active. Users should take great care when using this feature, if port F has any pullup or down hardware connected to it. Port F becomes the low byte of the address bus when enabled and damage could result unless the user takes suitable precautions.

Both PH8 devices also support the enhanced baud rate selection. This feature is present on the MC68HC11K4 (K4) MCU and [1] [References](#) describes its operation. By combining the wide range of baud rates available with the flexibility of the PLL autostart function, users can access almost every baud rate available.

The EPROM version of the PH8 contains the PROGRAM subroutine described in [References](#) [1]. The PH8 implementation allows dynamic relocation of the EPROM where the boot ROM overlaps the internal EPROM. It achieves this by detecting when the EPROM overlaps the boot ROM, moving the EPROM, adjusting the user's current address, programming the byte, restoring the user's address, and then moving the EPROM back again. The PROGRAM version of the K4, by contrast, moves the EPROM once before the user begins programming.

The ROM version of the PH8 also has an embedded version of a PCbug11 talker. Embedding talkers in boot ROM has the advantage of freeing memory in RAM (from where talkers normally run). The disadvantage of such a talker is that there is no possibility to alter it if required, for example, to change programming delay times. Also, due to the limited memory available, the talker in the PH8 boot ROM does not support the trace and breakpoint functions of PCbug11.

To use the boot ROM talker, the user must force a jump to the start of the boot ROM talker. [Listing 1. PCbug11 MC68HC\(7\)11PH8 Talker Initialization Code](#) shows the code required to perform the initialization. Note that the boot ROM code only initializes the SCI vector by default; the user should add further initialization to this talker file, if required. To generate a machine readable talker, assemble the file using ASMHC11 with the ;B option. The .MAP shown in [Listing 2. PCbug11 PH8 Talker Map File](#) tells PCbug11 where the talker code is. This must have the same name as the talker file and both files require the use of PCbug11 version 3.40 or later.

Listing 1. PCbug11 MC68HC(7)11PH8 Talker Initialization Code

```
M68HC11 Absolute Assembler Version 2.70C:\talkph8.ASC
1 A          0000          ORG          $0
2 A    0000    7EBE40    start    JMP          $BE40
3 A                                     END
```

Listing 2. PCbug11 PH8 Talker Map File

Name of constant must not exceed 14 characters.

Value of constant must start in column 15 or higher.

```
talker_start    $BE40          Talker code start address. (TLKRSTART)
talker_idle     $BE5E          Talker code idle loop address. (IDLE)
user_start      $BE49          User's reset entry into talker code. (USERSTART)
xirq_ujmp       $00F2          Address of user's XIRQ server address.
relocate_buf    $00A0          PCbug11 workspace in MCU RAM
xirq_srv        $BE61          Talker's XIRQ service address. (SCISRV)
swi-srv         $BE99          Talker's SWI service address for break points. (SWISRV)
swi_idle        $BE5E          Talker's SWI idle loop. (SWIIDLE)
null_srv        $BE99          Talker RTI. (NULLSRV)
xirq_jmp        $00F2          XIRQ vector.
swi_jmp         $00F5          SWI vector.
cme_jmp         $00FE          COP clock monitor vector.
```

MC68HC(7)11E20/32 Bootstrap Modes

These devices are derivatives of the popular MC68HC(7)11E9 devices. The EPROM versions have two enhanced features:

- A revised security mode to protect EPROM
- An embedded PCbug11 talker

A standard security mode exists on many M68HC11 MCUs. This protects internal EEPROM and RAM from access by erasing them before the MCU enters bootstrap mode. The user enables this feature by clearing the NOSEC bit in the CONFIG register on the MCU.

On the MC68HC711E20 (711E20) and MC68HC711E32 (711E32) MCUs, the same control bit activates the security mode. However, the new security mode protects the internal EPROM as well as EEPROM and RAM. The new protection takes the form of a blank check on the entire EPROM array. If any EPROM cell is not blank (\$FF), then the CPU enters an infinite loop doing nothing. While this enhancement greatly improves the security of customer information, it also brings greater responsibility. In particular, users should take great care before clearing the NOSEC bit on one-time programmable (OTP) devices. Since these devices are not erasable, it is impossible to re-enter bootstrap mode again. In addition, the order in which the CPU checks internal memory means that the user may be unable to use the MCU again. For erasable devices with quartz window, the mode is re-usable once the user erases the internal EPROM. Of course, if the EPROM is blank, then there is no security breach anyway, and the NOSEC bit being cleared will erase the internal EEPROM as normal in bootstrap mode.

The enhanced security mode secures memory in this order:

1. EEPROM
2. RAM
3. EPROM
4. CONFIG register

This means that MCU erases its internal EEPROM and RAM before checking the internal EPROM. For applications with internal variables or

preset values in the EEPROM, placing the device in bootstrap mode would erase those values even if the EPROM is not blank. If in addition the device is an OTP, then bootstrap mode is not usable again to reprogram the contents of the EEPROM.

In summary, this security mode offers a degree of protection for internal memory contents much greater than standard M68HC11 security mode. However, users must take great care to ensure that they only enable the mode when convinced that they have no further need for bootstrap mode, except, of course, for erasable devices.

The PCbug11 talkers embedded in the 711E20 and 711E32 are full implementations unlike the PH8 version listed earlier. In this case, as well as causing a jump to the talker, it is also advisable to initialize the SWI interrupt vector to point to the null service routine (RTI). This precaution avoids system problems where the user has an SWI interrupt. Since the 7E20 and 7E32 talkers vary slightly, the user must generate two sets of talker and .MAP files. The files are described in:

- [Listing 3. PCbug11 MC68HC\(7\)11E20 Talker Initialization Code](#)
- [Listing 4. PCbug11 MC68HC\(7\)11E20 Talker Map File](#)
- [Listing 5. PCbug11 MC68HC\(7\)11E32 Talker Initialization Code](#)
- [Listing 6. PCbug11 MC68\(7\)11E32 Talker Map File](#)

Listing 3. PCbug11 MC68HC(7)11E20 Talker Initialization Code

```

M68HC11 Absolute Assembler Version 2.70C:\talk7e20.ASC
1 A          0000          ORG          $0
2 A    0000    7EBE40    start    JMP          $BE40
3 A          00F4          ORG          $F4
4 A    00F4    7EBEA1          JMP          $BEA1
5 A                                     END
    
```

Listing 4. PCbug11 MC68HC(7)11E20 Talker Map File

Name of constant must not exceed 14 characters.

Value of constant must start in column 15 or higher

talker_start	\$BE40	Talker code start address. (TLKRSTART)
talker_idle	\$BE5B	Talker code idle loop address. (IDLE)
user_start	\$BE49	User's reset entry into talker code. (USERSTART)
xirq_ujmp	\$00F2	Address in talker code of user's XIRQ server address.
relocate_buf	\$00A0	Address to where user's code is relocated on break point.
xirq_srv	\$BE5E	Talker's XIRQ service address. (SCISRV)
swi-srv	\$BEDD	Talker's SWI service address for break points. (SWISRV)
swi_idle	\$BEE1	Talker's SWI idle loop. (SWIIDLE)
null_srv	\$BEA1	Talker RTI. (NULLSRV)
xirq_jump	\$00F2	XIRQ vector.
swi_jump	\$00F5	SWI vector.
cme_jump	\$00FE	COP clock monitor vector.

Listing 5. PCbug11 MC68HC(7)11E32 Talker Initialization Code

```

M68HC11 Absolute Assembler Version 2.70C:\talk7e32.ASC
1 A          0000          ORG          $0
2 A    0000    7BE40    start    JMP          $BE40
3 A          00F4          ORG          $F4
4 A    00F4    7EBE9B          JMP          $BE9B
5 A                                     END
    
```

Listing 6. PCbug11 MC68(7)11E32 Talker Map File

Name of constant must not exceed 14 characters.

Value of constant must start in column 15 or higher

talker_start	\$BE40	Talker code start address. (TLKRSTART)
talker_idle	\$BE5B	Talker code idle loop address. (IDLE)
user_start	\$BE49	User's reset entry into talker code. (USERSTART)
xirq_ujmp	\$00F2	Address in talker code of user's XIRQ server address.
relocate_buf	\$00A0	Address to where user's code is relocated on break point.
xirq_srv	\$BE5E	Talker's XIRQ service address. (SCISRV)
swi_srv	\$BED7	Talker's SWI service address for break points. (SWISRV)
swi_idle	\$BEDB	Talker's SWI idle loop. (SWIIDLE)
null_srv	\$BE9B	Talker RTI. (NULLSRV)
xirq_jump	\$00F2	XIRQ vector.
swi_jump	\$00F5	SWI vector.
cme_jump	\$00FE	COP clock monitor vector.

Boot ROM IDs

Across the M68HC11 Family, boot ROM IDs are available (with some exceptions) to allow users to identify the device in use and the revision of the boot ROM software. The ID is present in the boot ROM at one of two locations and in one of two formats. The format in use in general depends on the lineage and age of the device.

The first ID originated with the MC68HC11E9 and uses two bytes that contain this information:

- EPROM (711) or not
- Device numeric identifier, up to 15
- Device alphabetic identifier, one character or two if both are less than F

The second ID originated later, uses a different format, and contains this information:

- EPROM (711) or not
- Bootstrap ROM allows for enhanced security operation or not
- First digit of device identifier
- Second digit of device identifier or enhanced numeric identifier
- Numeric identifier up to 15 or enhanced numeric identifier up to 63

Table 3 gives details on how to decode the first format.

Table 3. M68HC11 Device ID First Format

Address	Content
\$BFD1	Revision of boot ROM, A upward
\$BFD2, \$BFD3	Mask set ID, \$0000 for EPROM, otherwise used by mask generation
\$BFD4, \$BFD5	2-byte device ID as follows: Bit 15–bit 12 is a 7 if EPROM, 0 if ROM. Bit 11–bit 8 is hex nibble of first ASCII letter. Bit 7–bit 4 is hex nibble of second ASCII letter. Bit 3–bit 0 is hex nibble of digit. Alternatively: Bit 11–bit 8 is digit.

Table 4 gives details on how to decode the second format.

Table 4. M68HC11 Device ID Second Format

Address	Content
\$BFBB	Revision of boot ROM, A upward
\$BFBC, \$BFBD	Mask set ID, \$0000 for EPROM, otherwise used by mask generation
\$BFBE, \$BFBF	2-byte device ID as follows: Bit 15 is 0 if the part is ROM or ROMless, 1 if the part is EPROM. Bit 14 is a 0 if unsecured. Bit 13–bit 9 are lower five bits of first ASCII letter. Bit 8–bit 4 are lower five bits of second ASCII letter. Bit 3–bit 0 are last digit of part number. Alternatively: If bit 8 – bit 6 are %111, then bit 5–bit 0 are last digit of number part.

Table 5 gives guidance on where to find the ID for many M68HC11 devices.

Table 5. M68HC11 Boot ROM Device IDs

Device	ID	Location
MC68HC11A0/1/8	None	N/A
MC68HC11D3	\$11D3	\$BFD4
MC68HC11E20	\$E9E9	\$BFD4
MC68HC11E32	\$4BE0	\$BFBE
MC68HC11E9	\$E9E9	\$BFD4
MC68HC11EA9	\$0EA9	\$BFD4
MC68HC11ED0	None	N/A
MC68HC11F1	\$F1F1	\$BFD4
MC68HC11K4	\$044B	\$BFD4
MC68HC11PA8	\$6018	\$BFBE
MC68HC11PH8	\$2088	\$BFBE
MC68HC711D3	\$71D3	\$BFD4
MC68HC711E20	\$CBF4	\$BFBE
MC68HC711E32	\$CBE0	\$BFBE
MC68HC711E9	\$71E9	\$BFD4
MC68HC711EA9	\$7EA9	\$BFD4
MC68HC711K4	\$744B	\$BFD4
MC68HC711PA8	\$E018	\$BFBE
Mc68HC711PH8	\$E088	\$BFBE

Conclusion

By using the features embedded in the bootstrap ROMs of the described MCUs, the user can enjoy enhanced bootstrap operation from the M68HC11 Family. [Table 6](#) describes the bootstrap features of several M68HC11 devices at the time of printing. Complete commented listings of the boot ROM programs in seven specific versions of the M68HC11 are contained in:

- [Listing 7. MC68HC11ED0 Bootloader ROM Listing](#)
- [Listing 8. MC68HC711EA9 Boatloader ROM Listing](#)
- [Listing 9. MC68HC711PH8 Bootloader ROM Listing](#)
- [Listing 10. MC68HC11PH8 Bootloader ROM Listing](#)
- [Listing 11. MC68HC711E20 Secured Bootloader ROM Listing](#)
- [Listing 12. MC68HC711E32 Secured Bootloader ROM Listing](#)
- [Listing 13. MC68HC11E32 Secured Bootloader ROM Listing](#)

Other versions can be found in [References](#) [1] and [2].

References

- [1] *M68HC11 Bootstrap Mode*, Motorola document order number AN1060/D
- [2] *M68HC11 Reference Manual*, Motorola document order number M68HC11RM/AD
- [3] *PCbug11 User's Manual*, Motorola document order number M68PCBUG11/D
- [4] *ROMed HC11E32 and HC11PH8 including Buffalo Monitor and PCbug11 Talker*, Motorola document order number EB419/D

Table 6. M68HC11 Boot ROM Features

MCU	Download (Bytes)	PROGRAM	UPLOAD	TALKER	Other
MC68HC11A0/1/8	Fixed, 256	No	No	No	None
MC68HC11D3	Variable, 512	No	No	No	None
MC68HC11E20	Variable, 768	No	No	No	None
MC68HC11E32	Variable, 2048	No	No	Yes	None
MC68HC11E9	Variable, 512	No	No	No	None
MC68HC11EA9	Variable 512	No	No	No	None
MC68HC11ED0	Fixed, 256	No	No	No	None
MC68HC11F1	Variable, 1024	No	No	No	None
MC68HC11K4	Variable, 768	No	No	No	None
MC68HC11PA8	Variable, 2048	No	No	Yes	PLL autostart
MC68HC11PH8	Variable, 2048	No	No	Yes	PLL autostart
MC68HC711D3	Variable, 512	Yes	Yes	No	None
MC68HC711E20	Variable, 768	No	No	Yes	Enhanced security
MC68HC711E32	Variable, 2048	No	No	Yes	Enhanced security
MC68HC711E9	Variable, 512	Yes	Yes	No	Enhanced security on request
MC68HC711EA9	Variable, 512	Yes	Yes	No	No
MC68HC711K4	Variable, 768	Yes	Yes	No	No
MC68HC711PA8	Variable, 2048	Yes	No	No	PLL autostart
Mc68HC711PH8	Variable, 2048	Yes	No	No	PLL autostart

Listing 7. MC68HC11ED0 Bootloader ROM Listing

```

0001                                *Bootloader Firmware for MC68HC11ED0
0002
0003                                *****
0004                                * November 11 1991
0005                                *****
0006
0007                                * Equates for use with index offset = 0
0008
0009    0008    PORTD    EQU    $08
0010    0009    DDRD    EQU    $09
0011    0028    SPCR    EQU    $28
0012    002b    BAUD    EQU    $2B
0013    002c    SCCR1   EQU    $2C
0014    002d    SCCR2   EQU    $2D
0015    002e    SCSR    EQU    $2E
0016    002f    SCDAT   EQU    $2F
0017    003b    PPROG   EQU    $3B
0018    003f    CONFIG  EQU    $3F
0019
0020                                *****
0021                                * This bootstrap program allows the user to
0022                                * download a program of exactly 256 bytes.
0023                                * The program must start at $0100.
0024                                * Each byte of the program is received by the
0025                                * SCI, starting with the $0100 byte and working
0026                                * up to the $01FF.
0027                                *
0028                                * This bootloader is based on the bootloader for the
0029                                * MC68HC11A8. Modifications are to place the registers
0030                                * at $00XX and destination to $0100.
0031                                *****
0032    bf80                                ORG    $BF80
0033    bf80                                BEGIN  EQU    *
0034
0035                                * Init.stack
0036    bf80    8e    00    ff                LDS    #$00FF
0037                                * Put PORTD in wired-or mode
0038    bf83    14    28    20                BSET   SPCR    $20
0039                                * Init SCI, restart the divider chain, enable tx & rx
0040    bf86    cc    a2    0c                LDD    #$A20C
0041    bf89    97    2d                    STAA   BAUD
0042    bf8b    d7    2d                    STAB   SCCR2
0043                                * Send break as soon as start bit is detected
0044    bf8d    14    2d    01                BSET   SCCR2    $01
0045                                * Clear break as soon as start bit is detected
0046    bf90    12    08    01    fc          BRSET  PORTD    $01 *
0047    bf94    15    2d    01    fc          BCLR  SSCR2    $01 *      Clear break
0048                                * Wait for first character (users send $ff)
0049    bf97    13    2e    20    fc          BRCLR  SCSR    $20 *      Wait for RDRF
0050    bf9b    96    2f                    LDAA   SCDAT      Read data
0051                                *** No jump to EEPROM or RAM here !!
0052                                * If data = $FF, then /16 is correct baud
0053    bf9d    81    ff                    CMPA   #$FF
0054    bf9f    27    03                    BEQ    BAUDOK
0055                                * else change to /104 (/13 & /8) 1200 @ 2 MHz
0056    bfa1    14    2b    33                BSET   BAUD    $33
0057                                * Then download 256 byte program
0058    bfa4                                BAUDOK EQU *
0059    bfa4    ce    01    00                LDX    #$100      Init pointer
0060                                * Read in program and put in RAM
0061    bfa7                                BK2    EQU    *

```

Freescale Semiconductor, Inc.

```

0062  bfa7  13  2e  20 fc          BRCLR      SCSR $20 * $20
0063  bfab  96  2f                LDAA      SCDAT
0064  bfad  a7  00                STAA     $00,x
0065  bfaf  97  2f                STAA     SCDAT
0066  bfb1  08                INX
0067                                * Until the end is reached
0068  bfb2  8c  02  00          CPX      $#0200
0069  bfb5  26  f0          BNE      BK2
0070                                *****
0071                                * All start user's program
0072                                *
0073  bfb7                STAR     EQU      *
0074  bfb7  7e  01  00          JMP      $0100
0075
0076                                END
  
```

Listing 8. MC68HC711EA9 Boatloader ROM Listing

```

0001                                *****
0002                                *BOOTLOADER FIRMWARE FOR 68HC711EA9 - 23 Aug 91
0003                                *****
0004                                * Features of this bootloader are. . .
0005                                *
0006                                * Auto baud select between 7812.5 and 1200 (8 MHz)
0007                                * 0 - 512 byte variable length download
0008                                * Jump to EEPROM at $B600 if 1st download byte = $00
0009                                * PROGRAM - Utility subroutine to program EPROM
0010                                * UPLOAD - Utility subroutine to dump memory to host
0011                                * Mask I.D. at $BFD4 = $7EA9
0012                                *****
0013                                * Revision A -
0014                                *
0015                                * This bootloader based on the MC68HC711E9
0016                                * Bootloader Revision A with SCI register
0017                                * behavior from MC68HC711K4 Bootloader
0018                                * Revision B
0019                                *
0020                                * IMPORTANT VDDSYN must be tied low to allow the
0021                                * oscillator to work as a normal 68HC11.
0022                                *
0023                                * This new version allows variable length download
0024                                * by quitting reception of characters when an idle
0025                                * of at least four character times occurs
0026                                *
0027                                *****
0028                                *
0029                                * EQUATES FOR USE WITH INDEX OFFSET = $1000
0030                                *
0031  0008          PORTD      EQU      $08
0032  000e          TCNT      EQU      $0E
0033  0016          TOC1      EQU      $16
0034  0023          TFLG1     EQU      $23
0035                                * BIT EQUATES FOR TFLG1
0036  0080          0CF1      EQU      $80
0037                                *
0038  0028          SCBD      EQU      $28          Baud Register
0039  002a          SCCR1     EQU      $2A
0040  002b          SCCR2     EQU      $2B
0041  002c          SCSR1     EQU      $2C
0042  002d          SCSR2     EQU      $2D
0043  002e          SCDRH     EQU      $2E
0044  002f          SCDRL     EQU      $2F
  
```

EB422

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0045 003b      PPROG      EQU      $3B
0046          * BIT EQUATES FOR PPROG
0047 0020      ELAT      EQU      $20
0048 0001      EPGM      EQU      $01
0049          *
0050
0051          * MEMORY CONFIGURATION EQUATES
0052          *
0053 b600      EEPMSTR    EQU      $B600      Start of EEPROM
0054 b7ff      EEPMEND    EQU      $B7FF      End of EEPROM
0055          *
0056 d000      EPRMSTR    EQU      $D000      Start of EPROM
0057 ffff      EPRMEND    EQU      $FFFF      End of EPROM
0058          *
0059 0000      RAMSTR     EQU      $0000
0060 01ff      RAMEND     EQU      $01FF
0061
0062          * DELAY CONSTANTS
0063          *
0064 0db0      DELAYS     EQU      3504      Delay at slow baud
0065 021b      DELAYF     EQU      539       Delay at fast baud
0066          *
0067 1068      PROGDEL    EQU      4200      2 ms prog delay
0068          *                               At 2.1 MHz
0069
0070          *****
0071 bf00      ORG$BF00
0072          *****
0073
0074          * Next two instructions provide a predictable place
0075          * to call PROGRAM and UPLOAD even if the routines
0076          * change size in future versions.
0077          *
0078 bf00 7e bf 13  PROGRAM    JMP      PRGROUT    EPROM prog utility
0079 bf03          UPLOAD     EQU      *          Upload utility
0080
0081          *****
0082          * UPLOAD - Utility subroutine to send data from
0083          * inside the MCU to the host via the SCI interface.
0084          * Prior to calling UPLOAD set baud rate, turn on SCI
0085          * and set Y=first address to upload.
0086          * Bootloader leaves baud set, SCI enabled, and
0087          * Y pointing at EPROM start ($D0000) so these default
0088          * values do not have to be changed typically.
0089          * Consecutive locations are sent via SCI in an
0090          * infinite loop. Reset stops the upload process.
0091          *****
0092 bf03 ce 10 00          LDX      #$1000      Point to registers
0093 bf06 18 a6 00          LDAA     0,Y          Read byte
0094 bf09 1f 2c 80 fc      BRCLR   SCSR1,X $80 *  Wait for TDRE
0095 bf0d a7 2f          STAA     SCDRL,X      Send it
0096 bf0f 18 08          INY
0097 bf11 20 f3          BRA      UPLOOP      Next...
0098

```



```

0099          *****
0100          * PROGRAM - Utility subroutine to program EPROM.
0101          * Prior to calling PROGRAM set baud rate, turn on SCI
0102          * set X=2ms prog delay constant, and set Y=first
0103          * address to program. SP must point to RAM.
0104          * Bootloader leaves baud set, SCI enabled, X=4200
0105          * and Y pointing at EPROM start ($D000) so these
0106          * default values don't have to be changed typically.
0107          * Delay constant in X should be equivalent to 2 ms
0108          * at 2.1 MHz X=4200; at 1 MHz X=2000.
0109          * At external voltage source is required for EPROM
0110          * programming.
0111          * This routine uses 2 bytes of stack space
0112          * Routine does not return. Reset to exit.
0113          *****
0114          bf13          PRGROUT      EQU          *
0115          bf13 3c          PSHX          Save prog dly
0116          bf14 cd 10 00    LDX          #$1000    Point to regs
0117
0118          * Send $FF to indicate ready for program data
0119
0120          bf171f 2c 80 fc          BRCLR      SCSR1,X $80 *    Wait for TDRE
0121          bf1b 86 ff          LDAA          #$FF
0122          bf1d a7 2f          STAA          SCDRL,X
0123
0124          bf1f          WAIT1      EQU          *
0125          bf1f 1f 2c 20 fc          BRCLR      SCSR1,X $20 *    Wait for RDRF
0126          bf23 e6 2f          LDAB          SCDRL,X          Get rx byte
0127          bf25 18 e1 00          CMPB          $0,Y          Already prog?
0128          bf28 27 1d          BEQ          DONEIT          Skip prog cyc
0129          bf2a 86 20          LDAA          #ELAT          EPROM in prog
0130          bf2c a7 3b          STAA          PPROG,X
0131          bf2e 18 e7 00          STAB          0,Y          Write data
0132          bf31 86 21          LDAA          #ELAT+EPGM
0133          bf33 a7 3b          STAA          PPROG,X          Prog start
0134          bf35 32          PULA          Get dly const
0135          bf36 33          PULB          into D-reg
0136          bf37 37          PSHB          But keep dly
0137          bf38 36          PSHA          on stack
0138          bf39 e3 0e          ADDD          TCNT,X          Delay + TCNT
0139          bf3b ed 16          STD          TOC1,X          2ms delay
0140          bf3d 86 80          LDAA          #OC1F
0141          bf3f a7 23          STAA          TFLG1,X          Clear flag
0142
0143          bf41 1f 23 80 fc          BRCLR      TFLG1,X OC1F *    Wait for dly
0144          bf45 6f 3b          CLR          PROG,X          Prog stop
0145
0146          *
0147          bf47          DONEIT      EQU          *
0148          bf47 1f 2c 80 fc          BRCLR      SCSR1,X $80 *    Wait for TDRE
0149          bf4b 18 a6 00          LDAA          $0,Y          Get EPROM &
0150          bf4e a7 2f          STAA          SCDRL,X          Xmit for verf
0151          bf50 18 08          INY          Next location
0152          bf52 20 cb          BRA          WAIT1          Back for next
0153
0154          * Loops indefinitely as long as more data sent.
0155
0156          *****
0157          * Main bootloader starts here
0158          *****
0159          * RESET vector bootloader starts here
0160
0161          bf54          BEGIN      EQU          *
0162          bf54 8e 01 ff          LDS          #RAMEND          Initialise SP
0163          bf57 ce 10 00          LDX          #$1000          Point at regs
0164          bf5a cc 00 20          LDD          #$0020          Init baud for

```

Engineering Bulletin

```

0163 bf5d ed 28          STD          SCBD,X          7812 at 2MHz
0164 bf5f cc 40 0c      LDD          #$400C          Port D WOI &
0165 bf62 ed 2a          STD          SCCR1,X          Rx & Tx on
0166 bf64 cc 02 1b      LDD          #DELAYF          Fast baud dly
0167 bf67 ed 16          STD          TOC1,X          is default
0168
0169
0170 bf69 1c 2b 01      * Send BREAK to signal ready for download
0171 bf6c 1e 08 01 fc      BSET         SCCR2,X $01      Send break
0172 bf70 1d 2b 01      BRSET        PORTD,X $01 *     RxD pin low?
0173
0174 bf73 1f 2c 20 fc      BCLR         SCCR2,X $01      Clear break
0175 bf77 a6 2f
0176
0177 bf79 26 03          * Data will be $00 if BREAK OR $00 received
0178 bf7b 7e b6 00          BNE          NOTZERO          No JMP ! = 0
0179 bf7e                JMP          EEPMSTR          JMP to EEPROM
0180 bf7e 81 ff          NOTZERO      EQU          *
0181 bf80 27 09          CMPA         #$FF            $FF is $FF
0182
0183 bf82 c6 d0          * Or else change to 1200 @ 2MHz
0184 bf84 e7 29          LDAB         #$D0
0185 bf86 cc 0d b0          STAB         SCBD+1,X          Baud to 1200
0186 bf89 ed 16          LDD          #DELAYS          And slower...
0187 bf8b                STD          TOC1,X          delay const
0188 bf8b 18 ce 00 00      BAUDOK      EQU          *
0189
0190 bf8f                LDY         #RAMSTR          Start of RAM
0191 bf8f ec 16          WAI          EQU          *
0192 bf91                LDD         TOC1,X          D = Dly const
0193 bf91 1e 2c 20 07      WTLOOP      EQU          *          Exit loop if RDRF set
0194 bf95 8f                BRSET        SCSR1,X $20 NEWONE
0195 bf96 09                XGDX
0196 bf97 8f                DEX          X = Dly const
0197 bf98 26 f7                DEX          Dec count
0198 bf9a 20 0f                DEX          D = Dly const
0199
0200 bf9c                BNE          WTLOOP          Finished?
0201 bf9c a6 2f                BRA          STAR            Quit on t/o
0202 bf9e 18 a7 00          NEWONE      EQU          *
0203 bfa1 a7 2f                LDAA        SCDRL,X          Put rx data
0204 bfa3 18 08                LDAA        $00,Y            into next RAM
0205 bfa5 18 8c 02 00       LDAA        SCDRL,X          Tx for hshake
0206 bfa9 26 e4                INY          Next RAM loc
0207
0208 bfab                CPY         #RAMEND+1       Past end?
0209 bfab ec 10 68          STAR        EQU          *          No, get next
0210 bfae 18 ce d0 00       LDX         #PROGDEL        X = prog dly
0211 bfb2 7e 00 00          LDY         #EPRMSTR        Y=EPROM start
0212
0213
0214
0215
0216 bfb5 00 00 00 00 00 00 BSZ          $BFD1-*          To RAM start
0217
0218
0219
0220
0221 bfd1 41          *****
          * Block fill unused bytes with zeros
          *****
          * Boot ROM revision level in ASCII
          * (ORG$BFD1)
          FCC "A"

```

Freescale Semiconductor, Inc.

```

0222 *****
0223 * Mask set I.D. ($0000 FOR EPROM PARTS)
0224 * (ORG$BFD2)
0225 bfd2 00 00 FDB $0000
0226 *****
0227 * '711EA9 I.D. - Can be used to determine MCU type
0228 * (ORG$BFD4)
0229 bfd4 7e a9 FDB $7EA9
0230 *****
0231 * VECTORS - point to RAM for pseudo-vector JUMPS
0232 *****
0233 *****
0234 bfd6 00 c4 FDB $100-60 SCI
0235 bfd8 00 c7 FDB $100-57 SPI
0236 bfda 00 ca FDB $100-54 PULSE ACCUM INPUT EDGE
0237 bfdc 00 cd FDB $100-51 PULSE ACCUM OVERFLOW
0238 bfde 00 d0 FDB $100-48 TIMER OVERFLOW
0239 bfe0 00 d3 FDB $100-45 TIMER O/P COMPARE 5
0240 bfe2 00 d6 FDB $100-42 TIMER O/P COMPARE 4
0241 bfe4 00 d9 FDB $100-39 TIMER O/P COMPARE 3
0242 bfe6 00 dc FDB $100-36 TIMER O/P COMPARE 2
0243 bfe8 00 df FDB $100-33 TIMER O/P COMPARE 1
0244 bfef 00 e2 FDB $100-30 TIMER I/P CAPTURE 3
0245 bfec 00 e5 FDB $100-27 TIMER I/P CAPTURE 2
0246 bfee 00 e8 FDB $100-24 TIMER I/P CAPTURE 1
0247 bff0 00 eb FDB $100-21 REAL TIME INT
0248 bff2 00 ee FDB $100-18 IRQ
0249 bff4 00 f1 FDB $100-15 XIRQ
0250 bff6 00 f4 FDB $100-12 SWI
0251 bff8 00 f7 FDB $100-9 ILLEGAL OP-CODE
0252 bffa 00 fa FDB $100-6 COP FAIL
0253 bffc 00 fd FDB $100-3 CLOCK MONITOR
0254 bfef bf 54 FDB BEGIN RESET
0255 END

```

Listing 9. MC68HC711PH8 Bootloader ROM Listing

```

0001 *****
0002 * BOOTLOADER FIRMWARE FOR MC68HC711PH8 - 7 OCT 93
0003 *****
0004 * Features of this bootloader are...
0005 *
0006 * Auto baud select between 7812, 1200, 9600, 5208
0007 * and 3906 (E = 2 MHz).
0008 * 0 - 768 byte variable length download:
0009 * reception of characters quits when an idle of at
0010 * least four character times occurs. (Note: at 9600
0011 * baud rate this is almost five bit times and at
0012 * 5208 and 3906 rates the timeout is even longer).
0013 * Jump to EEPROM at $0D00 if first byte = $00.
0014 * PROGRAM - Utility subroutine to program EPROM.
0015 * Part I.D. at $BFBE is $E088. {7PH8}
0016 *****
0017 * Revision A (1 DEC 92) -
0018 *
0019 * Based on P2 Bootloader Rev B.
0020 *****
0021 * Revision B (7 OCT 93) -
0022 *
0023 * Security mode caused an overwrite of registers
0024 * with $FF. Fix implemented - (CPX)

```

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0025 *
0026 * Altered PLL startup behaviour as follows :
0027 * 1/ Changed programmed value read from PORT B to F
0028 * 2/ If $FF is found then use value $CC instead. This
0029 * winds a 38.4KHz crystal to 8MHz (i.e. * 208).
0030 * 3/ Default delay of 10ms has been adjusted for a
0031 * 38.4KHz crystal rather than 640KHz.
0032 *
0033 * INITIAL subroutine removed since it was useless and
0034 * PROGRAM enhanced to move the EPROM block iff the
0035 * destination address clashes with the boot ROM
0036 *****
0037
0038 * Equates (registers in direct space)
0039 *
0040 0004 PORTB EQU $04
0041 0005 PORTF EQU $05
0042 0008 PORTD EQU $08
0043 0009 DDRD EQU $09
0044 *
0045 000e TCNT EQU $0E
0046 0016 TOC1 EQU $16
0047 0023 TFLG1 EQU $23
0048 002e PLLCR EQU $2E
0049 002f SYNCR EQU $2F
0050 * Bit equates for TFLG1
0051 0080 OC1F EQU $80
0052 *
0053 002b EPROG EQU $2B
0054 * Bit equates for EPROG
0055 0020 ELAT EQU $20
0056 0001 EPGM EQU $01
0057 *
0058 0035 BPROT EQU $35
0059 0037 INIT2 EQU $37
0060 003b PPROG EQU $3B
0061 003d INIT EQU $3D
0062 003e TEST1 EQU $3E
0063 003f CONFIG EQU $3F
0064 *
0065 0070 SCBD EQU $70
0066 0072 SCCR1 EQU $72
0067 0073 SCCR2 EQU $73
0068 0074 SCSR1 EQU $74
0069 0075 SCSR2 EQU $75
0070 0076 SCDRH EQU $76
0071 0077 SCDRL EQU $77
0072
0073 * Memory configuration equates
0074 *
0075 0d00 EEPMSTR EQU $0D00 Start of EEPROM
0076 0fff EEPMEND EQU $0FFF End of EEPROM
0077 *
0078 4000 EPRMSTR EQU $4000 Start of EPROM
0079 ffff EPRMEND EQU $FFFF End of EPROM
0080 *
0081 0080 RAMSTR EQU $0080 Start of RAM
0082 087f RAMEND EQU $087F End of RAM
0083
0084 be40 BOOTSTR EQU $BE40 Start of boot ROM
0085 bfff BOOTEND EQU $BFFF End of boot ROM
0086 00b0 BOOTH1 EQU $B0 Hi BR address
0087 0080 BOOTLO EQU $80 Shifted BR address
0088

```

```

0089          * Delay constants
0090          *
0091 15ab      DELAYS      EQU      5547      Dly at slow baud rate
0092 0356      DELAYF     EQU      854       Dly at fast baud rates
0093          *
0094 1068      PROGDEL   EQU      4200      2 mSec prog delay
0095          *                                     at 2.1MHz
0096          *
0097          *****
0098 be40          ORG      $BE40
0099          *****
0100
0101          *****
0102          * REVISION B - When the EPROM destination address
0103          * is within the range of the boot ROM, then move the
0104          * EPROM down to $0000 and alter the destination
0105          * address to match.
0106          *
0107          * PROGRAM - Utility subroutine to program EPROM.
0108          * Prior to calling PROGRAM set baud rate, turn on SCI
0109          * set X=2ms prog delay constant, and set Y=first
0110          * address to program. SP must point to RAM.
0111          * Bootloader leaves baud set, and SCI enabled so
0112          * default values do not have to be changed typically.
0113          * Delay constant in X should be equivalent to 2 ms
0114          * at 2.1 MHz X=4200; at 1 MHz X=2000, at 4MHz X=8000.
0115          * An external voltage source is required for EPROM
0116          * programming.
0117          * This routine uses 4 bytes of stack space.
0118          * Routine does not return. Reset to exit.
0119          *****
0120 be40      PROGOUT   EQU      *
0121
0122          * Send $FF to indicate ready for program data
0123
0124 be40 13 74 80 fc      BRCLR   SCSR1 $80 * Wait for TDRE
0125 be44 86 ff          LDAA    #$FF
0126 be46 97 77          STAA    SCDRL
0127
0128          * WAIT FOR A BYTE
0129 be48      WAIT1     EQU      *
0130 be48 13 74 20 fc      BRCLR   SCSR1 $20 * Wait for RDRF
0131 be4c d6 77          LDAB    SCDRL   Get received byte
0132          * REVISION B - Check if boot ROM is in the way
0133 be4e 18 8c be 40      CPY     #BOOTSTR   Is Y < BOOTSTR
0134 be52 25 22          BLO     NOADJ
0135 be54 18 8c bf ff      CPY     #BOOTEND   Is Y > BOOTEND
0136 be58 22 1c          BHI     NOADJ
0137
0138          * REVISION B - Move Boot ROM and adjust Y
0139 be5a      ADJ      EQU      *           Adjust Y & move EPROM
0140 be5a 18 8f          XGDY
0141 be5c 84 0f          ANDA    #$0F      Mask off hi nibble
0142 be5e 8b 80          ADDA    #BOOTLO   Replace with hi nibble
0143 be60 18 8f          XGDY
0144 be62 86 0f          LDAA    #$0F      EPROM on
0145 be64 97 3f          STAA    CONFIG   at shifted addr
0146 be66 8d 14          BSR     PROG     Program memory
0147 be68 18 8f          XGDY
0148 be6a 84 0f          ANDA    #$0F      Mask off hi nibble
0149 be6c 8b b0          ADDA    #BOOTHI   Replace with hi nibble
0150 be6e 18 8f          XGDY
0151 be70 86 8f          LDAA    #$8F      EPROM at standard addr
0152 be72 97 3f          STAA    CONFIG

```

EB422

Engineering Bulletin

```

0153 be74 20 02          BRA      CARRYON      Find next address
0154
0155 be76              NOADJ      EQU      *          No need to adjust Y
0156 be76 8d 04        BSR      PROG
0157
0158 be78              CARRYON EQU *
0159 be78 18 08        INY
0160 be7a 20 cc        BRA      WAIT1       Point to next location
0161                          Back to top for next
0162          * Loops indefinitely as long as more data sent.
0163
0164          * REVISION B - Now a subroutine
0165 be7c              PROG      EQU      *
0166 be7c 18 e1 00      CMPB    $0,Y         See if already prog
0167 be7f 27 1d        BEQ     DONEIT       If so, skip prog cycle
0168 be81 86 20        LDAA   #ELAT        Put EPROM in prog mode
0169 be83 97 2b        STAA   EPROG
0170 be85 18 e7 00    STAB   0,Y          Write data
0171 be88 86 21        LDAA   #ELAT+EPGM
0172 be8a 97 2b        STAA   EPROG       Turn on prog voltage
0173 be8c 3c          PSHX          Save delay on stack
0174 be8d 32          PULA          Put delay into D-reg
0175 be8e 33          PULB
0176 be8f d3 0e      ADDD   TCNT        Dly const+present TCNT
0177 be91 dd 16      STD    TOC1        Schedule OC1 =prog dly
0178 be93 86 80      LDAA   #OC1F
0179 be95 97 23      STAA   TFLG1       Clear any prev flag
0180
0181 be97 13 23 80 fc  BRCLR  TFLG1 OC1F * Wait for delay to expire
0182 be9b 7f 00 2b    CLR    EPROG       Turn off prog voltage
0183
0184 be9e              DONEIT   EQU      *
0185 be9e 13 74 80 fc  BRCLR  SCSR1 $80 * Wait for TDRE
0186 bea2 18 a6 00    LDAA   $0,Y        Read from EPROM and...
0187 bea5 97 77      STAA   SCDRL       Xmit for verify
0188 bea7 39          RTS            Go back for more
0189
0190          *****
0191          *
0192          * REVISION B - Load from PORTF, change $FF -> $CC
0193          * Now check to see if the PLL is active:
0194          * If VDDSYN pin is low the PLL is inactive and the
0195          * MCU will continue using the EXTAL frequency
0196          * This is detected since the PLLON bit is forced to
0197          * zero.
0198          * If VDDSYN pin is high then the MCU will switch
0199          * to the new higher frequency. After a delay of
0200          * 10ms (for a crystal of 38.4KHz).
0201          * The value to be stored in the SYNRR register is
0202          * loaded from port F.
0203          * IMPORTANT NOTE: IF SOME PINS ON PORTF ARE
0204          * SHORTED TO GROUND, BE SURE
0205          * TO REMOVE SHORTS BEFORE
0206          * SETTING MDA BIT FOR EXPANDED
0207          * BUSES
0208          * Procedure for PLL is:
0209          * 1/ If PLLON=0 then continue bootloader
0210          * 2/ Load value from port F - ACCB
0211          * 3/ If ACCB=$FF then ACCB:=$CC
0212          * 4/ Reset BCS=0
0213          * 5/ Reset PLLON=0
0214          * 6/ Store ACCB in SYNRR
0215          * 7/ Set PLLON=1
0216          * 8/ If PLLON=0 then continue bootloader

```

```

0217          * 9/ If PLLON=1 then wait 10ms (@ 38.4KHz)
0218          * 10/ Set BCS=1 and MCS=1
0219          * 11/ Continue bootloader
0220          *****
0221 bea8      PLLSTRT EQU *
0222 bea8 96 2e          LDAA    PLLCR      Get control values
0223 beaa 2a 26          BPL     DONE     Not active carry on
0224 beac d6 05          LDAB    PORTF    Get SYNCR value
0225 beae c1 ff          CMPB   #$FFF    Check for default
0226 beb0 26 02          BNE    NOTFF    Not $FFF so carry on
0227 beb2 c6 cc          LDAB    #$CC     Load value for 8MHz
0228 beb4          NOTFF    EQU     *
0229 beb4 84 bf          ANDA   #$BF     Reset BCS=0
0230 beb6 97 2e          STAA   PLLCR
0231 beb8 84 3f          ANDA   #$3F     Reset PLLON=0
0232 beba 97 2e          STAA   PLLCR
0233 bebc d7 2f          STAB   SYNCR    New value for SYNCR
0234 bebe 8a 80          ORAA   #$80     Set PLLON=1
0235 bec0 97 2e          STAA   PLLCR
0236 bec2 96 2e          LDAA   PLLCR    Check if PLL is active
0237 bec4 2a 0c          BPL    DONE     Not active carry on
0238 bec6 18 ce 00 0e    LDY    #14      Dly for 10ms @ 38.4KHz
0239 beca 18 09          DELLP  DEY
0240 becc 26 fc          BNE    DELLP
0241 bece 8a 44          ORAA   #$44
0242 bed0 97 2e          STAA   PLLCR    BCS/MCS for high speed
0243 bed2 39          DONE   RTS
0244
0245          *****
0246          * EEPROM ERASE DELAY
0247 bed3          ERASE   EQU     *
0248 bed3 18 ce 0b b8    LDY    #3000
0249 bed7          BK1     EQU     *
0250 bed7 18 09          DEY
0251 bed9 26 fc          BNE    BK1
0252 bedb 7f 00 3b      CLR    PPROG    Clear
0253 bede 39          RTS
0254
0255          *****
0256          * Block fill unused bytes with zero
0257
0258 bedf 00 00 00 00 00 00 00          BSZ    $BF00-*
0259          00 00 00 00 00 00
0260          00 00 00 00 00 00
0261          00 00 00 00 00 00
0262          00 00 00 00 00 00
0263          00 00 00
0264          *****
0265          * Next instruction provides a predictable place
0266          * to call PROGRAM even if the routine changes in
0267          * size in future versions. Note that the "UPLOAD"
0268          * routine did not fit on this part.
0269          *
0270          PROGRAM JMP  PROGOUT Program utility
0271          *****
0272          * Main bootloader starts here
0273          *****
0274          * RESET vector points to here

```

Freescale Semiconductor, Inc.

Engineering Bulletin

```

0275 bf03          BEGIN          EQU          *
0276 bf03 8e 08 7f          LDS          #RAMEND      Initialize stack ptrntr
0277 bf06 8d a0          BSR          PLLSTRT      Turn on PLL
0278
0279 bf08          CONTINU EQU    *
0280 bf08 cc 00 1a          LDD          #$001A      Initialize baud for...
0281 bf0b dd 70          STD          SCBD        9600 baud at 2 MHz
0282 bf0d cc 40 0c          LDD          #$400C      Put SCI in WOI mode...
0283 bf10 dd 72          STD          SCCR1      Enable Xmtr and Rcvr
0284          * Test the security bit
0285 bf12 12 3f 08 3c          BRSET       CONFIG $08 NOSEC
0286          *****
0287          * WE ARE IN SECURITY MODE
0288          * OUTPUT $FF ON TRANSMITTER
0289 bf16          AGAIN          EQU          *
0290 bf16 96 74          LDAA        SCSR1
0291 bf18 86 ff          LDAA        #$FF
0292 bf1a 97 77          STAA        SCDRL      Transmit $FF
0293          * ACCA NOW IS SET FOR $FF
0294
0295          * ERASE EEPROM
0296 bf1c 7f 00 35          CLR          BPROT      Turn off Block Protect
0297 bf1f c6 06          LDAB        #$06      EELAT and BULK ERASE
0298 bf21 d7 3b          STAB        PPROG      Set EELAT=1 & ERASE=1
0299 bf23 f7 0d 00          STAB        EEPROMSTR  Save in any EEPROM loc
0300 bf26 5c          INCB
0301 bf27 d7 3b          STAB        PPROG      to begin programming
0302 bf29 8d a8          BSR          ERASE      Wait 10ms (@ 2MHz)
0303          * If EEPROM is not enabled then we can't check its
0304          * erased - ACCA is still $FF
0305 bf2b 13 3f 01 0d          BRCLR      CONFIG $01 NOEE
0306 bf2f ce 0d 00          LDX          #EEPSTR
0307          * Check the EEPROM is erased
0308 bf32          LOOP          EQU          *
0309 bf32 a1 00          CMPA        0,X
0310 bf34 26 e0          BNE          AGAIN      EEPROM not erased
0311 bf36 08          INX
0312 bf37 8c 10 00          CPX          #EEPSTR+1  All checked
0313 bf3a 26 f6          BNE          LOOP
0314 bf3c          NOEE          EQU          *
0315          *****
0316          * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES WHICH
0317          * ARE USED BY THE STACK AND RAMSTR WHICH IS LEFT
0318          * INTACT. ACCA IS STILL $FF
0319 bf3c          ERAM          EQU          *
0320 bf3c ce 08 7d          LDX          #RAMEND-2
0321 bf3f          LOPl          EQU          *
0322 bf3f a7 00          STAA        0,X
0323 bf41 09          DEX
0324 bf42 8c 00 7f          CPX          #RAMSTR-1  *** REVISION B
0325 bf45 26 f8          BNE          LOPl
0326          *****
0327          * NOW ERASE CONFIG REGISTER
0328 bf47          ECONFG        EQU          *
0329 bf47 5a          DECB
0330 bf48 d7 3b          STAB        PPROG      B still = $06
0331 bf4a d7 3f          STAB        CONFIG     BULK ERASE CONFIG
0332 bf4c 5c          INCB
0333 bf4d d7 3b          STAB        PPROG
0334 bf4f bd be d3          JSR          ERASE
0335
0336          *****
0337          * NON-SECURITY AND SECURITY MODES MEET HERE
0338          *

```



```

0339 bf52          NOSEC      EQU      *
0340 bf52 cc 03 56      LDD      #DELAYF      Dly for fast bd rates
0341 bf55 dd 16          STD      TOC1          Set as default delay
0342                * Send BREAK to signal ready for download
0343 bf57 14 73 01      BSET     SCCR2 $01     Set send break bit
0344 bf5a 12 08 01 fc    BRSET    PORTD $01 *  RxD pin low?
0345 bf5e 15 73 01      BCLR     SCCR2 $01     Clear send break bit
0346
0347 bf61 13 74 20 fc    BRCLR    SCSR1 $20 *  Wait for RDRF
0348 bf65 96 77          LDAA     SCDRL          Read data
0349                * Data will be $00 if BREAK or $00 received
0350 bf67 26 03          BNE      NOTZERO      Bypass jump if not $00
0351 bf69 7e 0d 00      JMP      EEPROMSTR     Jump to EEPROM if $00
0352 bf6c          NOTZERO    EQU      *
0353                * Check div by 26 (9600 baud at 2 MHz)
0354 bf6c 81 f0          CMPA     #$F0          $F0 seen as $F0...
0355 bf6e 27 1d          BEQ      BAUDOK        if baud was correct
0356                * Check div by 208 (1200 baud at 2 MHz)
0357 bf70 c6 d0          LDAB     #$D0          Init B for this rate
0358 bf72 81 80          CMPA     #$80          $FF seen as $80...
0359 bf74 27 10          BEQ      SLOBAUD       if baud was correct
0360                * Check div by 64 (3906 baud at 2 MHz)
0361                * (equals: 8192 baud at 4.2 MHz)
0362 bf76 c6 40          LDAB     #$40          Init B for this rate
0363 bf78 85 20          BITA     #$20          $FD has bit 5 clear...
0364 bf7a 27 0a          BEQ      SLOBAUD       if baud was correct
0365                * Change to div by 32 (7812 baud at 2 MHz)
0366                * (equals: 8192 baud at 2.1 MHz)
0367 bf7c c6 20          LDAB     #$20          Init B for this rate
0368 bf7e d7 71          STAB     SCBD+1
0369 bf80 85 08          BITA     #$08          $FF has bit 3 set...
0370 bf82 26 09          BNE      BAUDOK        if baud was correct
0371                * Change to div by 36 (7777 baud at 640KHz x 14)
0372                *
0373 bf84 c6 24          LDAB     #$24          By default
0374
0375 bf86          SLOBAUD    EQU      *
0376 bf86 d7 71          STAB     SCBD+1        Store baudrate
0377 bf88 cc 15 ab      LDD      #DELAYS      Switch to slower...
0378 bf8b dd 16          STD      TOC1          delay constant
0379 bf8d          BAUDOK    EQU      *
0380 bf8d 18 ce 00 80    LDY      #RAMSTR       Point to start of RAM
0381
0382 bf91          WAIT      EQU      *
0383 bf91 de 16          LDX      TOC1          Move delay const to X
0384 bf93          WTLOOP   EQU      *
0385 bf93 12 74 20 05    BRSET    SCSR1 $20 NEWONE Exit if RDRF set
0386 bf97 09          DEX      Decrement count
0387 bf98 26 f9          BNE      WTLOOP        Loop if not timed out
0388 bf9a 20 0f          BRA      STAR          Quit on timeout
0389
0390 bf9c          NEWONE   EQU      *
0391 bf9c 96 77          LDAA     SCDRL          Get received data
0392 bf9e 18 a7 00      STAA     $00,Y          Store to next RAM location
0393 bfa1 97 77          STAA     SCDRL          Transmit for h/shake
0394 bfa3 18 08          INY      Point to next RAM loc
0395 bfa5 18 8c 08 80    CPY      #RAMEND+1     See if past end
0396 bfa9 26 e6          BNE      WAIT          If not, get another
0397
0398 bfab          STAR      EQU      *
0399 bfab 7e 00 80      JMP      RAMSTR         ** To start of RAM **
    
```

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0400 *****
0401 * Block fill unused bytes with zero
0402
0403 bfae 00 00 00 00 00 00          BSZ      $BFBB-*
      00 00 00 00 00 00
      00

0404 *****
0405 * Boot ROM revision level in ASCII
0406
0407 *      (ORG $BFBB)
0408 bffb 42          FCC      "B"
0409 *****
0410 * Mask set I.D. ($0000 for EPROM parts)
0411 *      (ORG$BFBC)
0412 bfbf 00 00          FDB      $0000
0413 *****
0414 * 711PH8 I.D. - can be used to determine MCU type
0415 * Bit 15 is a 0 if the part is ROM (or ROMless,
0416 * 1 -> EPROM)
0417 * Bit 14 is a 0 if unsecured
0418 * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
0419 * Bits 8 - Bit 4 are lower 5 bits of 2nd ASCII letter
0420 * Bits 3 - Bit 0 are last digit of part number
0421 * (note: $50 = P , $48 = H in ASCII)
0422 *      (ORG$BFBE)
0423 bfbe e0 88          FDB      %1110000010001000
0424 *****
0425 * VECTORS - point to RAM for pseudo-vector JUMPs
0426
0427 bfc0 00 00          FDB      $0000      reserved
0428 bfc2 00 00          FDB      $0000      reserved
0429 bfc4 00 00          FDB      $0000      reserved
0430 bfc6 00 00          FDB      $0000      reserved
0431 bfc8 00 00          FDB      $0000      reserved
0432 bfca 00 00          FDB      $0000      reserved
0433 bfcc 00 b5          FDB      $100-75      BIT SYNC & LOCK
0434 bfce 00 b8          FDB      $100-72      PORTH WOI
0435 bfd0 00 bb          FDB      $100-69      8 BIT TIMERS
0436 bfd2 00 be          FDB      $100-66      SCI2
0437 bfd4 00 c1          FDB      $100-63      SPI2
0438 bfd6 00 c4          FDB      $100-60      SCI1
0439 bfd8 00 c7          FDB      $100-57      SPI1
0440 bfda 00 ca          FDB      $100-54      PULSE ACCUM INPUT EDGE
0441 bfdc 00 cd          FDB      $100-51      PULSE ACCUM OVERFLOW
0442 bfde 00 d0          FDB      $100-48      TIMER OVERFLOW
0443 bfe0 00 d3          FDB      $100-45      TIMER OUTPUT COMPARE 5
0444 bfe2 00 d6          FDB      $100-42      TIMER OUTPUT COMPARE 4
0445 bfe4 00 d9          FDB      $100-39      TIMER OUTPUT COMPARE 3
0446 bfe6 00 dc          FDB      $100-36      TIMER OUTPUT COMPARE 2
0447 bfe8 00 df          FDB      $100-33      TIMER OUTPUT COMPARE 1
0448 bfea 00 e2          FDB      $100-30      TIMER INPUT CAPTURE 3
0449 bfec 00 e5          FDB      $100-27      TIMER INPUT CAPTURE 2
0450 bfef 00 e8          FDB      $100-24      TIMER INPUT CAPTURE 1
0451 bff0 00 eb          FDB      $100-21      REAL TIME INT
0452 bff2 00 ee          FDB      $100-18      IRQ
0453 bff4 00 f1          FDB      $100-15      XIRQ
0454 bff6 00 f4          FDB      $100-12      SWI
0455 bff8 00 f7          FDB      $100-9      ILLEGAL OP-CODE
0456 bffa 00 fa          FDB      $100-6      COP FAIL
0457 bffc 00 fd          FDB      $100-3      CLOCK MONITOR
0458 bffe bf 03          FDB      BEGIN      RESET
0459          END
0460

```

Listing 10. MC68HC11PH8 Bootloader ROM Listing

```

0001 *****
0002 * SECURED BOOTLOADER FOR MC68HC11PH8 - 26 AUG 93
0003 *****
0004 * Features of this bootloader are...
0005 *
0006 * Auto baud select between 7812, 1200, 9600, 5208
0007 * and 3906 (E = 2 MHz).
0008 * 0 - 768 byte variable length download:
0009 * reception of characters quits when an idle of at
0010 * least four character times occurs. (Note: at 9600
0011 * baud rate this is almost five bit times and at
0012 * 5208 and 3906 rates the timeout is even longer).
0013 * Jump to EEPROM at $0D00 if first download byte = $00.
0014 * Part I.D. at $BFBE is $2088. {PH8}
0015 *****
0016 * Revision A (12 MAY 93) -
0017 *
0018 * Based on 7PH8 Bootloader Rev A.
0019 *****
0020 * Revision B (26 AUG 93) -
0021 *
0022 * Security mode caused an overwrite of registers
0023 * with $FF. Fix implemented - (CPX)
0024 *
0025 * Altered PLL startup behaviour as follows :
0026 * 1/ Changed programmed value read from PORT B to PORT F
0027 * 2/ If $FF is found then use value $CC instead. This
0028 * winds a 38.4KHz crystal to 8MHz (i.e. * 208).
0029 * 3/ Default delay of 10ms has been adjusted to be for a
0030 * 38.4KHz crystal rather than 640KHz.
0031 *
0032 *****
0033
0034 * Equates (registers in direct space)
0035 *
0036 0004 PORTB EQU $04
0037 0005 PORTF EQU $05
0038 0008 PORTD EQU $08
0039 0009 DDRD EQU $09
0040 *
0041 000e TCNT EQU $0E
0042 0016 TOC1 EQU $16
0043 0023 TFLG1 EQU $23
0044 002e PLLCR EQU $2E
0045 002f SYNRR EQU $2F
0046 * Bit equates for TFLG1
0047 0080 OC1F EQU $80
0048 *
0049 *
0050 0035 BPROT EQU $35
0051 0037 INIT2 EQU $37
0052 003b PPROG EQU $3B
0053 003d INIT EQU $3D
0054 003e TEST1 EQU $3E
0055 003f CONFIG EQU $3F
0056 *
0057 0070 SCBD EQU $70
0058 0072 SCCR1 EQU $72
0059 0073 SCCR2 EQU $73
0060 0074 SCSR1 EQU $74
0061 0075 SCSR2 EQU $75
0062 0076 SCDRH EQU $76
0063 0077 SCDRL EQU $77
0064
0065 * Memory configuration equates
0066 *
0067 0d00 EEPROMSTR EQU $0D00 Start of EEPROM
0068 0fff EEPROMEND EQU $0FFF End of EEPROM
0069 *
0070 0080 RAMSTR EQU $0080 Start of RAM
0071 087f RAMEND EQU $087F End of RAM

```

Freescale Semiconductor, Inc.

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0072
0073      * Delay constants
0074      *
0075 15ab DELAYS EQU 5547 Delay at slow baud rate
0076 0356 DELAYF EQU 854 Delay at fast baud rates
0077      *
0078 1068 PROGDEL EQU 4200 2 mSec programming delay
0079      * at 2.1MHz
0080      *
0081 bed5          ORG $BED5
0082      *
0083      *
0084      * REVISION B - Load from PORTF, change $FF -> $CC
0085      * Now check to see if the PLL is active:
0086      * If VDDSYN pin is low the PLL is inactive and the
0087      * MCU will continue using the EXTAL frequency
0088      * This is detected since the PLLON bit is forced to
0089      * zero.
0090      * If VDDSYN pin is high then the MCU will switch
0091      * to the new higher frequency. After a delay of
0092      * 10ms (for a crystal of 38.4KHz).
0093      * The value to be stored in the SYNRR register is
0094      * loaded from port F.
0095      * IMPORTANT NOTE: IF SOME PINS ON PORTF ARE
0096      * SHORTED TO GROUND, BE SURE
0097      * TO REMOVE SHORTS BEFORE
0098      * SETTING MDA BIT FOR EXPANDED
0099      * BUSES
0100      * Procedure for PLL is:
0101      * 1/ If PLLON=0 then continue bootloader
0102      * 2/ Load value from port F - ACCB
0103      * 3/ If ACCB=$FF then ACCB:=$CC
0104      * 4/ Reset BCS=0
0105      * 5/ Reset PLLON=0
0106      * 6/ Store ACCB in SYNRR
0107      * 7/ Set PLLON=1
0108      * 8/ If PLLON=0 then continue bootloader
0109      * 9/ If PLLON=1 then wait 10ms (@ 38.4KHz)
0110      * 10/ Set BCS=1 and MCS=1
0111      * 11/ Continue bootloader
0112      *
0113 bed5 PLLSTRT EQU *
0114 bed5 96 2e LDAA PLLCR Get control values
0115 bed7 2a 26 BPL DONE Not active carry on
0116 bed9 d6 05 LDAB PORTF Get SYNRR value
0117 bedb c1 ff CMPB #$FF Check for default
0118 bedd 26 02 BNE NOTFF Not $FF so carry on
0119 bedf c6 cc LDAB $CC Load value for 8MHz
0120 bee1 NOTFF EQU *
0121 bee1 84 bf ANDA #$BF Reset BCS=0
0122 bee3 97 2e STAA PLLCR
0123 bee5 84 3f ANDA #$3F Reset PLLON=0
0124 bee7 97 2e STAA PLLCR
0125 bee9 d7 2f STAB SYNRR New value for SYNRR
0126 beeb 8a 80 ORAA #$80 Set PLLON=1
0127 beed 97 2e STAA PLLCR
0128 beef 96 2e LDAA PLLCR Now check if PLL is active
0129 bef1 2a 0c BPL DONE Not active carry on
0130 bef3 18 ce 00 0e LDY #14 Delay for 10ms
0131 bef7 18 09 DELLP DEY
0132 bef9 26 fc BNE DELLP
0133 befb 8a 44 ORAA #$44
0134 befd 97 2e STAA PLLCR Set BCS/MCS bit for high speed
0135 beff 39 DONE RTS
0136
0137      *
0138 bf00          ORG $BF00
0139      *
0140      *
0141      *
0142      * Main bootloader starts here
0143      *
0144      * RESET vector points to here
0145 bf00 BEGIN EQU *

```

```

0146 bf00 8e 08 7f          LDS      #RAMEND          Initialize stack pntr
0147 bf03 8d d0            BSR      PLLSTRT          Turn on PLL
0148
0149 bf05                  CONTINU  EQU      *
0150 bf05 cc 00 1a          LDD      #$001A          Initialize baud for...
0151 bf08 dd 70            STD      SCBD             9600 baud at 2 MHz
0152 bf0a cc 40 0c          LDD      #$400C          Put SCI in wire-OR mode...
0153 bf0d dd 72            STD      SCCR1           Enable Xmtr and Rcvr
0154
0155 bf0f 12 3f 08 3d      * Test the security bit
                        BRSET   CONFIG $08 NOSEC
0156 *****
0157 * WE ARE IN SECURITY MODE
0158 * OUTPUT $FF ON TRANSMITTER
0159 bf13                  AGAIN   EQU      *
0160 bf13 96 74            LDAA    SCSR1
0161 bf15 86 ff            LDAA    #$FF
0162 bf17 97 77            STAA    SCDRL           Transmit $FF
0163
0164 * ACCA NOW IS SET FOR $FF
0165
0166 * ERASE EEPROM
0166 bf19 7f 00 35          CLR      BPROT           Turn off Block Protect
0167 bf1c c6 06            LDAB    #$06           For EELAT and BULK ERASE
0168 bf1e d7 3b          STAB    PPROG           Set EELAT=1 & ERASE=1
0169 bf20 f7 0d 00          STAB    EEPROMSTR       Store in any EEPROM location
0170 bf23 5c              INCB
                        SET     EEPROMMSTR
0171 bf24 d7 3b          STAB    PPROG           Set EEPROM bit
0172 bf26 bd bf ac          JSR     ERASE           to begin programming
                        Wait 10ms (@ 2MHz)
0173
0174 * If EEPROM is not enabled then we can't check its
                        * erased - ACCA is still $FF
0175 bf29 13 3f 01 0d      BRCLR   CONFIG $01 NOEE
0176 bf2d ce 0d 00          LDX     #EEPSTR
0177 * Check the EEPROM is erased
0178 bf30                  LOOP   EQU      *
0179 bf30 a1 00            CMPA    0,X
0180 bf32 26 df            BNE     AGAIN           EEPROM not erased
0181 bf34 08              INX
0182 bf35 8c 10 00          CPX     #EEPSTR+1       All checked
0183 bf38 26 f6            BNE     LOOP
0184 bf3a                  NOEE   EQU      *
0185 *****
0186 * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES WHICH
0187 * ARE USED BY THE STACK AND RAMSTR WHICH IS LEFT
0188 * INTACT. ACCA IS STILL $FF
0189 bf3a                  ERAM   EQU      *
0190 bf3a ce 08 7d          LDX     #RAMEND-2
0191 bf3d                  LOPI  EQU      *
0192 bf3d a7 00            STAA    0,X
0193 bf3f 09              DEX
0194 bf40 8c 00 7f          CPX     #RAMSTR-1**** REVISION B
0195 bf43 26 f8            BNE     LOPI
0196 *****
0197 * NOW ERASE CONFIG REGISTER
0198 bf45                  ECONF EQU      *
0199 bf45 5a              DECB
0200 bf46 d7 3b          STAB    PPROG           B still = $06
0201 bf48 d7 3f          STAB    CONFIG         BULK ERASE CONFIG
0202 bf4a 5c              INCB
0203 bf4b d7 3b          STAB    PPROG
0204 bf4d bd bf ac          JSR     ERASE
0205
0206 *****
0207 * NON-SECURITY AND SECURITY MODES MEET HERE
0208 *
0209 bf50                  NOSEC  EQU      *
0210 bf50 cc 03 56          LDD      #DELAYF         Delay for fast baud rates
0211 bf53 dd 16            STD      TOC1           Set as default delay
0212
0213 * Send BREAK to signal ready for download
0213 bf55 14 73 01          BSET    SCCR2 $01       Set send break bit
0214 bf58 12 08 01 fc          BRSET   PORTD $01 *     Wait for RxD pin to go low
0215 bf5c 15 73 01          BCLR    SCCR2 $01       Clear send break bit
0216
0217 bf5f 13 74 20 fc          BRCLR   SCSR1 $20 *     Wait for RDRF
0218 bf63 96 77            LDAA    SCDRL           Read data
0219
                        * Data will be $00 if BREAK or $00 received

```

EB422

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0220 bf65 26 03          BNE      NOTZERO      Bypass jump if not $00
0221 bf67 7e 0d 00     JMP      EEPMSTR      Jump to EEPROM if $00
0222 bf6a              NOTZERO  EQU      *
0223                  * Check div by 26 (9600 baud at 2 MHz)
0224 bf6a 81 f0          CMPA    #$F0          $F0 will be seen as $F0...
0225 bf6c 27 1d          BEQ     BAUDOK        if baud was correct
0226                  * Check div by 208 (1200 baud at 2 MHz)
0227 bf6e c6 d0          LDAB   #$D0          Initialize B for this rate
0228 bf70 81 80          CMPA    #$80          $FF will be seen as $80...
0229 bf72 27 10          BEQ     SLOBAUD       if baud was correct
0230                  * Check div by 64 (3906 baud at 2 MHz)
0231                  * (equals: 8192 baud at 4.2 MHz)
0232 bf74 c6 40          LDAB   #$40          Initialize B for this rate
0233 bf76 85 20          BITA   #$20          $FD shows as bit 5 clear...
0234 bf78 27 0a          BEQ     SLOBAUD       if baud was correct
0235                  * Change to div by 32 (7812 baud at 2 MHz)
0236                  * (equals: 8192 baud at 2.1 MHz)
0237 bf7a c6 20          LDAB   #$20          Initialize B for this rate
0238 bf7c d7 71          STAB   SCBD+1
0239 bf7e 85 08          BITA   #$08          $FF shows as bit 3 set...
0240 bf80 26 09          BNE    BAUDOK        if baud was correct
0241                  * Change to div by 36 (7777 baud at 640KHz x 14)
0242                  *
0243 bf82 c6 24          LDAB   #$24          By default
0244
0245 bf84              SLOBAUD  EQU      *
0246 bf84 d7 71          STAB   SCBD+1        Store baudrate
0247 bf86 cc 15 ab       LDD    #DELAYS       Switch to slower...
0248 bf89 dd 16          STD    TOC1          delay constant
0249 bf8b              BAUDOK   EQU      *
0250 bf8b 18 ce 00 80    LDY    #RAMSTR       Point to start of RAM
0251
0252 bf8f              WAIT     EQU      *
0253 bf8f de 16          LDX    TOC1          Move delay constant to X
0254 bf91              WTLOOP   EQU      *
0255 bf91 12 74 20 05   BRSET  SCSR1 $20 NEWONE Exit loop if RDRF set
0256 bf95 09            DEX                    Decrement count
0257 bf96 26 f9         BNE    WTLOOP        Loop if not timed out
0258 bf98 20 0f         BRA    STAR           Quit download on timeout
0259
0260 bf9a              NEWONE   EQU      *
0261 bf9a 96 77          LDAA   SCDRL         Get received data
0262 bf9c 18 a7 00       STAA   $00,Y         Store to next RAM location
0263 bf9f 97 77          STAA   SCDRL         Transmit it for handshake
0264 bfa1 18 08          INY                    Point to next RAM location
0265 bfa3 18 8c 08 80   CPY    #RAMEND+1     See if past end
0266 bfa7 26 e6          BNE    WAIT          If not, get another
0267
0268 bfa9              STAR     EQU      *
0269 bfa9 7e 00 80       JMP    RAMSTR        ** Exit to start of RAM **
0270 *****
0271                  * EEPROM ERASEDELAY
0272 bfac              ERASE    EQU      *
0273 bfac 18 ce 0b b8    LDY    #3000
0274 bfb0              BK1     EQU      *
0275 bfb0 18 09          DEY                    Clear
0276 bfb2 26 fc          BNE    BK1
0277 bfb4 7f 00 3b       CLR    PPROG
0278 bfb7 39            RTS
0279
0280 *****
0281                  * Block fill unused bytes with zero
0282
0283 bfb8 00 00 00       BSZ    $BFBB-*
0284
0285 *****
0286                  * Boot ROM revision level in ASCII
0287                  * (ORG$BFBB)
0288 bfbb 42              FCC     "B"
0289 *****
0290                  * Mask set I.D. ($0000 for EPROM parts)
0291                  * (ORG$BFBC)
0292 bfbc 00 00          FDB    $0000
0293 *****

```

```

0294 * 11PH8 I.D. - can be used to determine MCU type
0295 * Bit 15 is a 0 if the part is ROM (or ROMless,
0296 * 1 -> EPROM)
0297 * Bit 14 is a 0 if unsecured
0298 * Bits 13 - Bit 9 are lower 5 bits of first ASCII letter
0299 * Bits 8 - Bit 4 are lower 5 bits of second ASCII letter
0300 * Bits 3 - Bit 0 are last digit of part number
0301 * (note: $50 = P , $48 = H in ASCII)
0302 * (ORG$BFBE)
0303 bfbe 20 88 FDB %0010000010001000
0304 *****
0305 * VECTORS - point to RAM for pseudo-vector JUMPs
0306
0307 bfc0 00 00 FDB $0000 reserved
0308 bfc2 00 00 FDB $0000 reserved
0309 bfc4 00 00 FDB $0000 reserved
0310 bfc6 00 00 FDB $0000 reserved
0311 bfc8 00 00 FDB $0000 reserved
0312 bfca 00 00 FDB $0000 reserved
0313 bfcc 00 b5 FDB $100-75 BIT SYNC & LOCK
0314 bfce 00 b8 FDB $100-72 PORTH WOI
0315 bfd0 00 bb FDB $100-69 8 BIT TIMERS
0316 bfd2 00 be FDB $100-66 SCI2
0317 bfd4 00 c1 FDB $100-63 SPI2
0318 bfd6 00 c4 FDB $100-60 SCI1
0319 bfd8 00 c7 FDB $100-57 SPI1
0320 bfda 00 ca FDB $100-54 PULSE ACCUM INPUT EDGE
0321 bfdc 00 cd FDB $100-51 PULSE ACCUM OVERFLOW
0322 bfde 00 d0 FDB $100-48 TIMER OVERFLOW
0323 bfe0 00 d3 FDB $100-45 TIMER OUTPUT COMPARE 5
0324 bfe2 00 d6 FDB $100-42 TIMER OUTPUT COMPARE 4
0325 bfe4 00 d9 FDB $100-39 TIMER OUTPUT COMPARE 3
0326 bfe6 00 dc FDB $100-36 TIMER OUTPUT COMPARE 2
0327 bfe8 00 df FDB $100-33 TIMER OUTPUT COMPARE 1
0328 bfea 00 e2 FDB $100-30 TIMER INPUT CAPTURE 3
0329 bfec 00 e5 FDB $100-27 TIMER INPUT CAPTURE 2
0330 bfee 00 e8 FDB $100-24 TIMER INPUT CAPTURE 1
0331 bff0 00 eb FDB $100-21 REAL TIME INT
0332 bff2 00 ee FDB $100-18 IRQ
0333 bff4 00 f1 FDB $100-15 XIRQ
0334 bff6 00 f4 FDB $100-12 SWI
0335 bff8 00 f7 FDB $100-9 ILLEGAL OP-CODE
0336 bffa 00 fa FDB $100-6 COP FAIL
0337 bffc 00 fd FDB $100-3 CLOCK MONITOR
0338 bfbe bf 00 FDB BEGIN RESET
0339
0340 ***** TBRPH8.ASM 12/5/93 *****
0341 * Motorola Copyright 1993 *
0342 * MCU resident, Interrupt driven Communication routines for 68HC11 *
0343 * monitor.Provides low level memory and stack read/write operations. *
0344 * *
0345 * This talker DOES NOT use XIRQ *
0346 * ----- *
0347 * *
0348 * N.B. TBRPH8 is designed to work with the 68HC11PH8 or other *
0349 * compatible MCU types. This version of the TALKER is designed to *
0350 * execute from MC68HC11PH8 Boot ROM. *
0351 * To initiate communication with TBRPH8, the standard bootloader *
0352 * must be used to initialise the redirected vector table and then *
0353 * cause a jump to USER START *
0354 * This talker does NOT support SWI handling (trace and break) *
0355 * *
0356 * CONSTANTS
0357 be40 TALKBASE equ $BE40Start of RAM
0358 00c4 BOOTVECT equ $00C4Start of bootstrap vector jump table.
0359 087f STACK equ $087FAt end of this talker
0360 *
0361 00c4 JSCI equ $00C4SCI interrupt service
0362 00f1 JXIRQ equ $00F1XIRQ interrupt service
0363 00f4 JSWI equ $00F4SWI interrupt service
0364 00f7 JILLOP equ $00F7Illegal opcode service
0365 00fa JCOPI equ $00FACOP timeout reset service
0366 007e JMPEXT equ $7E Mnemonic for jump extended instruction
0367 004a BRKCODE equ $4A Break point signal code to host.

```


Engineering Bulletin

```

0368 004a      BRKACK    equ $4A  Break point acknowledge code fromhost.
0369          *
0370          * REGISTERS
0371 0020      RDRF      equ $20  Masks for checking
0372 0080      TDRE      equ $80  status of SCI
0373          *
0374          * PROGRAM
0375 be40      org          TALKBASE
0376          *
0377 be40      TLKRSTART EQU          *          Initialise SCI interrupt
0378 be40 86 7e LDAA      #JMPEXT
0379 be42 97 c4 STAA      JSCI
0380 be44 ce be 61 LDX      #SCISRV
0381 be47 df c5 STX      JSCI+1
0382 be49      USERSTART EQU          *
0383 be49 8e 08 7f LDS      #STACK          Initialise stack
0384 be4c cc 00 0d LDD      #13
0385 be4f dd 70 STD      SCBD          Initialise SCI to 19200 baud (SCBDH=0)
0386 be51 86 00 LDAA      #0
0387 be53 97 72 STAA      SCCR1          No LOOPS,WOMS,parity. Idle line wake up+8 bits
0388 be55 97 76 STAA      SCDRH          No data in bit 8
0389 be57 c6 2c LDAB      #S2C
0390 be59 d7 73 STAB      SCCR2          and enable SCI Rx interrupt & Tx.
0391 be5b 86 40 LDAA      #S40          Enable STOP, and I bit interrupts, disable XIRQ.
0392 be5d 06 TAP
0393          *
0394 be5e 7e be 5e IDLE     JMP      IDLE          Now hang around for SCI interrupt from host.
0395          *
0396          *A RESET from host changes above jump destination to start of user code.
0397          *
0398 be61      SCISRV    EQU          *          On detecting interrupt,
0399 be61 13 74 20 fc BRCLR   SCSR1    #RDRF SCISRV
0400          *
0401 be65      RXSRV    EQU          *          Talker code processes received data.
0402 be65 96 77 LDAA      SCDRL          Read command byte, & echo it as acknowledge
0403 be67 43 COMA          inverted
0404 be68 8d 3b BSR      OUTSCI          to host.
0405 be6a 2a 40 BPL      INH1          If command bit 7 set, then process inherent command
0406 be6c 8d 2c BSR      INSCI          else read byte count from host into ACCB.(0=256)
0407 be6e 8f XGDX          Save command and byte count.
0408 be6f 8d 29 BSR      INSCI          Read high address byte
0409 be71 17 TBA          into ACCA
0410 be72 8d 26 BSR      INSCI          then low address byte into ACCB
0411 be74 8f XGDX          Restore command in ACCA,count in ACCB,address in X
0412 be75 81 fe CMPA      #SFE
0413 be77 26 0d BNE      RXSRV1          If command is not memory read then RXSRV1
0414          *
0415 be79      TREADMEM EQU          *          REPEAT
0416 be79 a6 00 LDAA      ,X          read required address
0417 be7b 8d 28 BSR      OUTSCI          send it to host
0418 be7d 17 TBA          (save byte count)
0419 be7e 8d 1a BSR      INSCI          and wait for acknowledge
0420 be80 16 TAB          (restore byte count)
0421 be81 08 INX          Increment address
0422 be82 5a DECB          Decrement byte count
0423 be83 26 f4 BNE      TREADMEM          UNTIL all done
0424 be85 3b RTI
0425          *
0426 be86      RXSRV1   EQU          *
0427 be86 81 be CMPA      #SBE
0428 be88 26 0f BNE      RXSRVEX          If command is memory write then
0429          *
0430 be8a 17 TBA          move byte count to ACCA
0431 be8b      TWRTMEM  EQU          *          REPEAT
0432 be8b 8d 0d BSR      INSCI          Read next byte from host into ACCB,
0433 be8d e7 00 STAB      ,X          and store at required address.
0434 be8f 8d 13 BSR      EPRG          Dummy jump - used as hook for EPROM prog
0435 be91 e6 00 LDAB      ,X          Read stored byte and
0436 be93 d7 77 STAB      SCDRL          echo it back to host,
0437 be95 08 INX          Increment address
0438 be96 4a DECA          Decrement byte count
0439 be97 26 f2 BNE      TWRTMEM          UNTIL all done
0440 be99      RXSRVEX  EQU          *
0441 be99 3b NULLSRV   RTI          and return

```

EB422


```

0442          *
0443 be9a          INSCI      EQU          *
0444 be9a 12 74 0a a2 BRSET    SCSR1     #$0A TLKRSTART   Restart talker if break detected
0445 be9e 13 74 20 f8          BRCLR    SCSR1 #RDRF INSCI  Loop to INSCI if no character received
0446 bea2 d6 77          LDAB     SCDRL    then read data received from host
0447 bea4 39          EPRG     RTS        and return with data in ACCB
0448          *
0449 bea5          OUTSCI     EQU          *          Only register Y modified.
0450 bea5 13 74 80 fc OUTSCI1  BRCLR    SCSR1 #$80 OUTSCI1 Loop until set before storing next byte
0451 bea9 97 77          STAA    SCDRL    Important - Updates CCR!
0452 beab 39          RTS
0453          *
0454 beac          INH1       EQU          *
0455 beac 81 7e          CMPA    #$7E          If command is read MCU registers then INH1A
0456 beae 26 0c          BNE     INH2          else jump to INH2
0457          *
0458 beb0 30          INH1A   TSX          Move stack pointer to X
0459 beb1 8f          XGDX          then to ACCD
0460 beb2 8d f1          BSR     OUTSCI     send stack pointer to host (high byte first)
0461 beb4 17          TBA
0462 beb5 8d ee          BSR     OUTSCI     then low byte
0463 beb7 30          TSX          Restore X (=stack pointer)
0464 beb8 c6 09          LDAB    #9          then return 9 bytes on stack
0465 beba 20 bd          BRA     TREADMEM   i.e. CCR,ACCB,ACCA,IXH,IXL,IYH,IYL,PCH,PCL
0466          *
0467 bebc          INH2       EQU          *
0468 bebc 81 3e          CMPA    #$3E          If command is write MCU registers then don't jump
0469 bebe 26 d9          BNE    RXSRVEX     else quit processing
0470          *
0471 bec0 8d d8          BSR     INSCI     get stack pointer from host (High byte first)
0472 bec2 17          TBA
0473 bec3 8d d5          BSR     INSCI
0474 bec5 8f          XGDX          Move to X reg
0475 bec6 35          TXS          and copy to stack pointer
0476 bec7 86 09          LDAA    #9          Then put next 9 bytes from host on to stack
0477 bec9 20 c0          BRA     TWRITMEM
0478          *
0479          *****
0480          * Block fill unused bytes with zero
0481
0482 becb 00 00 00 00 00 00 00 00  BSZ     $BED5-*
0483          00 00 00 00
0484          *****
0485
0486          END
0487

```

Listing 11. MC68HC711E20 Secured Bootloader ROM Listing

```

0001          *****
0002          * FIRMWARE FOR SECURED 68HC711E20 - 11 December 1992
0003          *
0004          *****
0005          * SECURED EPROM VERSION
0006          *
0007          * BASED ON SECURE 711E9 BOOTLOADER 09 MAY 92
0008          * EXTENDED TO CHECK BOTH ROM BLOCKS OF E20
0009          *
0010          * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
0011          * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
0012          * OF AT LEAST FOUR WORD TIMES OCCURS
0013          *
0014          * EQUATES FOR USE WITH INDEX OFFSET = $1000
0015          *
0016          *
0017 0008          PORTD      EQU          $08
0018 0009          DDRD       EQU          $09

```

EB422

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0019 0016          TOC1          EQU          $16          [STORAGE (POOR STYLE)]
0020 0028          SPCR          EQU          $28          (FOR DWOM BIT)
0021 002b          BAUD          EQU          $2B
0022 002c          SCCR1         EQU          $2C
0023 002d          SCCR2         EQU          $2D
0024 002e          SCSR          EQU          $2E
0025 002f          SCDAT         EQU          $2F
0026 0035          BPROT         EQU          $35
0027 003b          PPROG         EQU          $3B
0028 003e          TEST1         EQU          $3E
0029 003f          CONFIG        EQU          $3F
0030
0031          * MORE EQUATES
0032          *
0033 bf00          EEPSTR          EQU          $B600         START OF EEPROM
0034 bf7f          EEPEND          EQU          $B7FF         END OF EEPROM
0035          *
0036 d000          EP1STR          EQU          $D000         START OF EPROM 1
0037 ffff          EP1END          EQU          $FFFF         END OF EPROM 1
0038 9000          EP2STR          EQU          $9000         START OF EPROM 2
0039 afff          EP2END          EQU          $AFFF         END OF EPROM 2
0040          *
0041 0000          RAMSTR          EQU          $0000
0042 02ff          RAMEND          EQU          $02FF
0043          *
0044 odb0          DELAYS          EQU          3504         DELAY AT SLOW BAUD
0045 021b          DELAYF          EQU          539          DELAY AT FAST BAUD
0046          *
0047          *****
0048          * THIS BOOTSTRAP PROGRAM ALLOWS THE USER TO
0049          * DOWNLOAD A PROGRAM OF 0 - 768 BYTES.
0050          * THE PROGRAM MUST START AT $0000.
0051          * EACH BYTE OF THE PROGRAM IS RECEIVED BY THE SCI.
0052          * THE FIRST BYTE ESTABLISHES BAUD RATE.
0053          * THEN THE PROGRAM IS DOWNLOADED STARTING WITH
0054          * THE $0000 BYTE AND WORKING UP TOWARD THE $01FF
0055          * A DELAY OF FOUR WORD TIMES (AT EITHER BAUD RATE)
0056          * CAUSES THE RECEPTION OF CHARACTERS TO STOP AND
0057          * A JUMP TO $0000.
0058          *
0059          * THE TRANSMITTER WILL BE USED FOR THE PURPOSE
0060          * OF COMMUNICATION TO THE OUTSIDE WORLD.
0061          *
0062          *****
0063 bf00          ORG          $BF00
0064          *
0065 bf00          BEGIN          EQU          *
0066          * INIT STACK
0067 bf00 8e 02 ff          LDS          #RAMEND
0068          * INIT X REG FOR INDEXED ACCESS TO REGISTERS
0069 bf03 ce 10 00          LDX          #$1000
0070          *
0071          * PUT PORT D IN WIRE OR MODE
0072 bf06 1c 28 20          BSET          SPCR,X $20
0073          * INIT SCI AND RESTART BAUD DIVIDER CHAIN
0074 bf09 cc a2 0c          LDD          #$A20C          DIV BY 16
0075 bf0c a7 2b          STAA          BAUD,X
0076          * RECEIVER & TRANSMITTER ENABLED
0077 bf0e e7 2d          STAB          SCCR2,X
0078          *
0079          * TEST THE SECURITY BIT
0080 bf10 1e 3f 08 5e          BRSET          CONFIG,X $08 NOSEC
0081          *****
0082          * WE ARE IN SECURITY MODE
0083          *
0084          * OUTPUT $FF ON TRANSMITTER
0085 bf14          AGAIN          EQU          *
0086 bf14 a6 2e          LDAA          SCSR,X

```

```

0087 bf16 86 ff          LDAA      #$FF
0088 bf18 a7 2f          STAA      SCDAT,X
0089                    * ACCA NOW IS SET FOR $FF
0090                    *
0091                    * ERASE EEPROM:
0092                    * TURN OFF BLOCK PROTECT
0093 bf1a 6f 35          CLR       BPROT,X
0094                    * SET ERASE AND EELAT BITS BEFORE USING "ERASE"
0095 bf1c 54              LSRB          CHANGE $0C TO $06
0096 bf1d e7 3b          STAB      PPROG,X
0097 bf1f f7 b6 00       STAB      EEPSTR      WRITE EEPROM LOCATION
0098 bf22 8d 34          BSR       ERASE
0099                    * ACCB IS NOW SET FOR $06
0100                    *
0101                    *
0102                    * ERASE CYCLE IS COMPLETE
0103                    *
0104                    * IF THE EEPROM IS NOT ENABLED,
0105                    * WE CAN'T CHECK THAT THE EEPROM IS ERASED
0106 bf24 1f 3f 01 11    BRCLR    CONFIG,X $01 NOEE
0107                    * EEPROM IS ON,
0108                    * NOW CHECK THAT THE EEPROM IS ERASED
0109 bf28 18 ce b6 00     LDY      #EEPSTR
0110 bf2c          LOOP    EQU      *
0111 bf2c 18 a1 00       CMPA     0,Y          (A = $FF)
0112                    * ANY UNERASED BYTE SENDS US BACK TO ERASE AGAIN
0113 bf2f 26 e3          BNE     AGAIN
0114 bf31 18 08          INY
0115 bf33 18 8c b8 00   CPY     #EEPEND+1
0116 bf37 26 f3          BNE     LOOP
0117 bf39          NOEE   EQU      *
0118                    *
0119                    * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES
0120                    * WHICH ARE USED BY THE STACK & $0000 WHICH IS
0121                    * LEFT INTACT
0122                    *
0123 bf39          ERAM   EQU      *
0124 bf39 3c          PSHX
0125 bf3a ce 02 fd     LDX     #RAMEND-2
0126 bf3d          LOP1  EQU      *
0127 bf3d a7 00       STAA    $00,X
0128 bf3f 09          DEX
0129 bf40 26 fb       BNE     LOP1
0130                    *
0131                    * DO NOT SEPARATE RAM AND EPROM ROUTINES WITHOUT <<<<<
0132                    * FIXING THE STACK (PULX)
0133                    *
0134                    * CONFIRM THAT EPROM IS ERASED
0135                    *
0136 bf42 ce d0 00     LDX     #EP1STR
0137 bf45 8d 1f       LOP2   BSR     TSTLP
0138 bf47 26 fc       BNE     LOP2
0139 bf49 ce 90 00    LOP2  LDX     #EP2STR
0140 bf4c a1 00       LOP3  CMPA    0,X          (A=$FF)
0141 bf4e 8d 16       BSR     TSTLP
0142 bf50 8c b0 00    CPX     #EP2END+1
0143 bf53 26 f7       BNE     LOP3
0144 bf55 38          PULX          <<<<<
0145 bf56 20 14       BRA     ECONFG
0146                    *
0147                    *
0148                    * BOOTLOADER SUBROUTINES
0149                    *
0150                    * EEPROM ERASE SUBROUTINE
0151                    *
0152                    * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
0153                    * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
0154                    * IS TO BE ERASED

```

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0155      *
0156      * ENTRY X=$1000
0157      * EXIT X=$1000, Y=$0000
0158      *
0159 bf58      ERASE          EQU          *
0160      * SET EEPROM BIT
0161 bf58 1c 3b 01      BSET          PPROG,X $01
0162      * 10 MILLISEC DELAY @ 2.1 MHZ
0163 bf5b 18 ce 0b b8      LDY          #3000
0164 bf5f 18 09      BK1          DEY
0165 bf61 26 fc      BNE          BK1
0166      * TURN OFF ERASE AND EELAT BITS
0167 bf63 6f 3b      CLR          PPROG,X
0168 bf65 39      RTS
0169      *
0170      * EPROM TEST SUBROUTINE
0171      *
0172 bf66 a1 00      TSTLP          CMPA          0,X          (A=$FF)
0173 bf68 26 fe      BNE          *
0174 bf6a 08      TSTOK          INX
0175 bf6b 39      RTS
0176
0177      *****
0178      * NOW ERASE CONFIG REGISTER
0179      *
0180 bf6c      ECONFG          EQU          *
0181      * SET ERASE AND EELAT BITS
0182 bf6c e7 3b      STAB          PPROG,X          (B STILL = $06)
0183      * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
0184 bf6e e7 3f      STAB          CONFIG,X
0185 bf70 8d e6      BSR          ERASE
0186      *****
0187      * ERASE CYCLE IS COMPLETE
0188      *
0189      *****
0190      * NON-SECURITY AND SECURITY MODES MEET HERE
0191      *
0192 bf72      NOSEC          EQU          *
0193      *****
0194      * SET UP DELAY FOR FASTEST BAUD RATE
0195 bf72 cc 02 1b      LDD          #DELAYF
0196 bf75 ed 16      STD          TOC1,X
0197      *****
0198      * SEND BREAK TO SIGNAL START OF DOWNLOAD
0199 bf77 1c 2d 01      BSET          SCCR2,X $01
0200      * CLEAR BREAK AS SOON AS START BIT IS DETECTED
0201 bf7a 1e 08 01 fc      BRSET        PORTD,X $01 *
0202 bf7e 1d 2d 01      BCLR         SCCR2,X $01      CLEAR BREAK
0203      * WAIT FOR FIRST CHARACTER (USERS SEND $FF)
0204 bf81 1f 2e 20 fc      BRCLR        SCSR,X $20 *      WAIT FOR RDRF
0205 bf85 a6 2f      LDAA         SCDAT,X          READ DATA
0206      * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
0207 bf87 26 03      BNE          NOTZERO
0208 bf89 7e b6 00      JMP          EEPSTR
0209 bf8c      NOTZERO          EQU          *
0210      * IF DATA = $FF, THEN /16 IS CORRECT BAUD
0211 bf8c 81 ff      CMPA         #$FF
0212 bf8e 27 08      BEQ          BAUDOK
0213      * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
0214 bf90 1c 2b 33      BSET         BAUD,X $33
0215      * SET UP DELAY FOR SLOWER BAUD RATE
0216 bf93 cc 0d b0      LDD          #DELAYS
0217 bf96 ed 16      STD          TOC1,X
0218      *
0219 bf98      BAUDOK          EQU          *
0220 bf98 18 ce 00 00      LDY          #RAMSTR          PNTR TO START OF RAM
0221      *
0222      * TIME EACH BYTE

```

```

0223 bf9c ec 16      WAIT          LDD          TOC1,X          PUT DELAY TIME IN ACCD
0224                *
0225 bf9e 1e 2e 20 07 WTLOOP          BRSET          SCSR,X $20 NEWONE
0226 bfa2 8f                XGDY          DELAY INTO X
0227 bfa3 09                DEX           DECREMENT DELAY
0228 bfa4 8f                XGDY          RETURN DELAY TO ACCD
0229 bfa5 26 f7          BNE           WTLOOP
0230 bfa7 20 0f          BRA           STAR
0231                * DID NOT TIME OUT
0232 bfa9                NEWONE       EQU           *
0233                * READ IN BYTE AND PUT INTO RAM
0234 bfa9 a6 2f          LDAA         SCDAT,X
0235 bfab 18 a7 00          STAA        $00,Y
0236 bfae a7 2f          STAA         SCDAT,X          HANDSHAKE
0237 bfb0 18 08          INY
0238 bfb2 18 8c 03 00      CPY          #RAMEND+1
0239 bfb6 26 e4          BNE          WAIT
0240                *****
0241                * START USER'S PROGRAM
0242                *
0243 bfb8                STAR          EQU           *
0244 bfb8 7e 00 00          JMP          RAMSTR
0245                *****
0246                * FILL UNUSED BYTES WITH ZERO
0247                *
0248 bfb8                BSZ          $BFBB-*
0249                *
0250                *****
0251                * REVISION LEVEL IN ASCII
0252                * (ORG          $BFB)
0253 bfb8 41                FCC          "A"
0254                *****
0255                * MASK I.D. ($0000 FOR EPROM PARTS)
0256                * (ORG          $BFB)
0257 bfb8 00 00          FDB          $0000
0258                *****
0259                * 711E20 I.D. - can be used to determine MCU type
0260                * Bit 15 is a 0 if the part is ROM (or ROMless,
0261                * 1 -> EPROM)
0262                * Bit 14 is a 0 if unsecured
0263                * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
0264                * Bits 8 - Bit 5 are $F => last five bits are number
0265                * Bits 4 - Bit 0 are last digit of part number
0266                * (note: $45 = E in ASCII)
0267                * (ORG$BFB)
0268 bfb8 cb f4          FDB          %1100101111110100
0269                *****
0270                * VECTORS          ($BDC0)
0271                *
0272 bfc0 00 00          FDB          $0000          reserved
0273 bfc2 00 00          FDB          $0000          reserved
0274 bfc4 00 00          FDB          $0000          reserved
0275 bfc6 00 00          FDB          $0000          reserved
0276 bfc8 00 00          FDB          $0000          reserved
0277 bfca 00 00          FDB          $0000          reserved
0278 bfcc 00 00          FDB          $0000          reserved
0279 bfce 00 00          FDB          $0000          reserved
0280 bfd0 00 00          FDB          $0000          reserved
0281 bfd2 00 00          FDB          $0000          reserved
0282 bfd4 00 00          FDB          $0000          reserved
0283 bfd6 00 c4          FDB          $100-60        SCI
0284 bfd8 00 c7          FDB          $100-57        SPI
0285 bfda 00 ca          FDB          $100-54        PULSE ACCUM INPUT EDGE
0286 bfdc 00 cd          FDB          $100-51        PULSE ACCUM OVERFLOW
0287 bfde 00 d0          FDB          $100-48        TIMER OVERFLOW
0288 bfe0 00 d3          FDB          $100-45        TIMER OUTPUT COMPARE 5
0289 bfe2 00 d6          FDB          $100-42        TIMER OUTPUT COMPARE 4
0290 bfe4 00 d9          FDB          $100-39        TIMER OUTPUT COMPARE 3

```

Engineering Bulletin

0291	bfe6	00	dc	FDB	\$100-36	TIMER OUTPUT COMPARE 2
0292	bfe8	00	df	FDB	\$100-33	TIMER OUTPUT COMPARE 1
0293	bfea	00	e2	FDB	\$100-30	TIMER INPUT CAPTURE 3
0294	bfec	00	e5	FDB	\$100-27	TIMER INPUT CAPTURE 2
0295	bfee	00	e8	FDB	\$100-24	TIMER INPUT CAPTURE 1
0296	bff0	00	eb	FDB	\$100-21	REAL TIME INT
0297	bff2	00	ee	FDB	\$100-18	IRQ
0298	bff4	00	f1	FDB	\$100-15	XIRQ
0299	bff6	00	f4	FDB	\$100-12	SWI
0300	bff8	00	f7	FDB	\$100-9	ILLEGAL OP-CODE
0301	bffa	00	fa	FDB	\$100-6	COP FAIL
0302	bffc	00	fd	FDB	\$100-3	CLOCK MONITOR
0303	bffe	bf	00	FDB	BEGIN	RESET

```

0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358

***** TBRE20.ASC 31/8/92 *****
* Motorola Copyright 1988,1990,1992 *
* MCU resident, Interrupt driven Communication routines for 68HC11 *
* monitor. Provides low level memory and stack read/write operations.*
*
* This talker DOES NOT uses XIRQ
* -----
*
* N.B. TBRE20 is designed to work with the 68HC11E20 or other
* compatible MCU types. This version of the TALKER is designed
* to execute from MC68HC711E20 Boot ROM.
* To initiate communication with TBRE20, the standard bootloader*
* must be used to initialise the redirected vector table and
* then cause a jump to USERSTART
*
* CONSTANTS
0322 be40 TALKBASE equ $BE40
0323 00c4 BOOTVECT equ $00C4 Start of boot vectors
0324 02ff STACK equ $02FF User may alter this
0325 1000 REGBASE equ $1000
*
0327 00c4 JSCI equ $00C4
0328 00f1 JXIRQ equ $00F1
0329 00f4 JSWI equ $00F4
0330 00f7 JILLOP equ $00F7
0331 00fa JCOP equ $00FA
0332 007e JMPEXT equ $7E Mnemonic for jump ext
0333 004a BRKCODE equ $4A Break point code
0334 004a BRKACK equ $4A Break point ack
*
* REGISTERS
0337 002f SCDR equ $2F
*
0339 0020 RDRF equ $20
0340 0080 TDRE equ $80
0341 0008 OR equ $08
0342 0002 FE equ $02
*
* PROGRAM
0345 be40 org TALKBASE
*
0347 be40 TLKRSTART EQU * Initialise SCI int
0348 be40 86 7e LDAA #JMPEXT
0349 be42 97 c4 STAA JSCI
0350 be44 ce be 5e LDX #SCISRV
0351 be47 df c5 STX JSCI+1
0352 be49 USERSTART EQU *
0353 be49 8e 02 ff LDS #STACK
0354 be4c ce 10 00 LDX #REGBASE
0355 be4f 6f 2c CLR SCCR1,X
0356 be51 cc 30 2c LDD #$302C Init SCI to 9600 baud,
0357 be54 a7 2b STAA BAUD,X no parity, no int
0358 be56 e7 2d STAB SCCR2,X & enable SCI tx & rx.

```

Freescale Semiconductor, Inc.

```

0359 be58 86 40          LDAA      #$40          Enable STOP, and I bit
0360 be5a 06            TAP                    int, disable XIRQ.
0361                    *                    Now hang around for
0362 be5b 7e be 5b      IDLE      JMP      IDLE      SCI int from host.
0363                    * A RESET from host changes above jump destination to
0364                    * start of user code.
0365 be5e                SCISRV      EQU      *                    On detecting int,
0366 be5e b6 10 2e      LDAA      SCSR+REGBASE    assume rx caused it.
0367 be61 84 20          ANDA      #RDRF
0368 be63 27 f9          BEQ      SCISRV          otherwise loop here
0369                    *
0370 be65                RXSRV      EQU      *                    Process received data.
0371 be65 b6 10 2f      LDAA      SCDR+REGBASE    Read cmd byte, & tx it
0372 be68 43            COMA                    inverted
0373 be69 8d 46          BSR      OUTSCI         as ack to host.
0374 be6b 2a 51          BPL      INH1           Bit 7 set? => inherent
0375 be6d 8d 33          BSR      INSCI          else B = byte count
0376 be6f 8f            XGDX                    Save cmd and byte cnt.
0377 be70 8d 30          BSR      INSCI          Read high address byte
0378 be72 17            TBA                    into ACCA
0379 be73 8d 2d          BSR      INSCI          then B = low addr byte
0380 be75 8f            XGDX                    A=cmd; B=cnt; X=addr
0381 be76 81 fe          CMPA      #$FE
0382 be78 26 0d          BNE      RXSRV1        If cmd is mem read ...
0383                    *
0384 be7a                TREADMEM    EQU      *                    REPEAT
0385 be7a a6 00          LDAA      ,X            read required address
0386 be7c 8d 33          BSR      OUTSCI        send it to host
0387 be7e 17            TBA                    (save byte count)
0388 be7f 8d 21          BSR      INSCI         and wait for ack
0389 be81 16            TAB                    (restore byte count)
0390 be82 08            INX                    Increment address
0391 be83 5a            DECB                    Decrement byte count
0392 be84 26 f4          BNE      TREADMEM UNTIL all done
0393 be86 3b            RTI                    & return
0394                    *
0395 be87                RXSRV1     EQU      *
0396 be87 81 be          CMPA      #$BE
0397 be89 26 16          BNE      RXSRVEX If cmd is mem write ..
0398                    *
0399 be8b 17            TBA                    move byte count to A
0400 be8c                TWRTMEM    EQU      *                    REPEAT
0401 be8c 8d 14          BSR      INSCI         Read next byte into B
0402 be8e e7 00          STAB      ,X            and store at req addr
0403 be90 18 ce 00 01    LDY      #0001          Set up wait loop
0404 be94 18 09          DEY                    WAITPOLL
0405 be96 26 fc          BNE      WAITPOLL
0406 be98 e6 00          LDAB      ,X            Read stored byte and
0407 be9a f7 10 2f      STAB      SCDR+REGBASE  echo it back to host,
0408 be9d 08            INX                    Decrement byte count
0409 be9e 4a            DECA                    UNTIL all done
0410 be9f 26 eb          BNE      TWRTMEM        and return
0411 bea1                RXSRVEX    EQU      *
0412 bea1 3b            NULLSRV    RTI
0413                    *
0414 bea2                INSCI      EQU      *
0415 bea2 f6 10 2e      LDAB      SCSR+REGBASE    Wait for RDRF=1
0416 bea5 c5 00          BITB      #(FE+OR)      If break detected then
0417 bea7 26 97          BNE      TLKRSTART      restart talker.
0418 bea9 c4 20          ANDB      #RDRF
0419 beab 27 f5          BEQ      INSCI
0420 bead f6 10 2f      LDAB      SCDR+REGBASE    read data received
0421 beb0 39            RTS                    & return B = data
0422                    *
0423 beb1                OUTSCI     EQU      *                    Only Y modified.
0424 beb1 18 8f          XGDY                    Enter with A = tx data
0425 beb3 b6 10 2e      OUTSCI1    LDAA      SCSR+REGBASE
0426 beb6 2a fb          BPL      OUTSCI1        MS bit is TDRE flag
    
```

EB422

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0427 beb8 18 8f          XGDY
0428 beba b7 10 2f      STAA          SCDR+REGBASE      Important: Updates CCR
0429 bebd 39            RTS
0430
0431 bebe                *
                        INH1          EQU          *
0432 bebe 81 7e          CMPA          #$7E          If cmd is read regs ..
0433 bec0 26 0c          BNE          INH2
0434
0435 bec2 30            *
                        INH1A        TSX
0436 bec3 8f            XGDX          Move SP to X
0437 bec4 8d eb          BSR          OUTSCI        then to ACCD
0438 bec6 17            TBA          send SP high byte 1st
0439 bec7 8d e8          BSR          OUTSCI        then low byte
0440 bec9 30            TSX          Restore X (=SP)
0441 beca c6 09          LDAB         #9           Tx 9 bytes on stack
0442 becc 20 ac          BRA          TREADMEM     i.e. CCR,ACCB,ACCA,IXH
0443                    *
                        INH2          EQU          *
0444 bece                *
                        INH2          EQU          *
0445 bece 81 3e          CMPA          #$3E          If cmd is write regs..
0446 bed0 26 12          BNE          SWISRV1
0447                    *
0448 bed2 8d ce          BSR          INSCI        get SP High byte first
0449 bed4 17            TBA
0450 bed5 8d cb          BSR          INSCI
0451 bed7 8f            XGDX          Move to X reg
0452 bed8 35            TXS          and copy to SP
0453 bed9 86 09          LDAA         #9           Then 9 bytes to stack
0454 bedb 20 af          BRA          TWRITMEM
0455                    *
0456 bedd                SWISRV          EQU          *           Breakpoints by SWI
0457 bedd 86 4a          LDAA         #BRKCODE     Force host to proc BR
0458 bedf 8d d0          BSR OUTSCI by sending it BREAK
0459 bee1 0e            CLI
0460 bee2 20 fd          BRA SWIIDLE then wait for response
0461                    *
0462 bee4                SWISRV1        EQU          *
0463 bee4 81 4a          CMPA #BRKACK If host acknowledges
0464 bee6 26 b9          BNE RXSRVEX
0465 bee8 30            TSX          move SP to SWI stack &
0466 bee9 c6 09          LDAB #9
0467 beeb 3a            ABX          Send user code
0468 beec 35            TXS          breakpoint return
0469 beed ec 07          LDD 7,X      address to host
0470 beef 8d c0          BSR OUTSCI (high byte first)
0471 bef1 17            TBA
0472 bef2 8d bd          BSR OUTSCI (low byte next)
0473 bef4 cc be e1      LDD #SWIIDLE force idle loop on
0474 bef7 ed 07          STD 7,X      return from breakpoint
0475 bef9 20 c7          BRA INH1A   but first return all
0476                    *
                        *
0477                    *
0478                    * FILL UNUSED BYTES WITH ZERO
0479                    *
0480 befb 00 00 00 00 00 BSZ          $BF00-*
0481                    *
0482                    *
0483                    *
0484                    *
0485                    *
0486                    *
                        END

```


Listing 12. MC68HC711E32 Secured Bootloader ROM Listing

```

0001 *****
0002 * FIRMWARE FOR SECURED 68HC711E32 - 16 July 1993
0003 *
0004 *****
0005 * SECURED EPROM VERSION
0006 *
0007 * Part I.D. at $BFBE is $CBE0. {7E32}
0008 *****
0009 * REVISION A
0010 * BASED ON SECURE 711E20 BOOTLOADER 11 DECEMBER 92
0011 * EXTENDED TO CHECK BOTH LARGER ROM BLOCKS OF E32
0012 *
0013 *****
0014 * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
0015 * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
0016 * OF AT LEAST FOUR WORD TIMES OCCURS
0017 *
0018 * EQUATES FOR USE WITH INDEX OFFSET = $1000
0019 *
0020 *
0021 0008 PORTD EQU $08
0022 0009 DDRD EQU $09
0023 0016 TOC1 EQU $16 [STORAGE (POOR STYLE)]
0024 0028 SPCR EQU $28 (FOR DWOM BIT)
0025 002b BAUD EQU $2B
0026 002c SCCR1 EQU $2C
0027 002d SCCR2 EQU $2D
0028 002e SCSR EQU $2E
0029 002f SCDAT EQU $2F
0030 0035 BPROT EQU $35
0031 003b PPROG EQU $3B
0032 003e TEST1 EQU $3E
0033 003f CONFIG EQU $3F
0034 *
0035 * MORE EQUATES
0036 *
0037 b600 EEPSTR EQU $B600 START OF EEPROM
0038 b7ff EEPEND EQU $B7FF END OF EEPROM
0039 *
0040 c000 EP1STR EQU $C000 START OF EPROM 1
0041 ffff EP1END EQU $FFFF END OF EPROM 1
0042 7000 EP2STR EQU $7000 START OF EPROM 2
0043 afff EP2END EQU $AFFF END OF EPROM 2
0044 *
0045 0000 RAMSTR EQU $0000
0046 07ff RAMEND EQU $07FF
0047 *
0048 0db0 DELAYS EQU 3504 DELAY AT SLOW BAUD
0049 021b DELAYF EQU 539 DELAY AT FAST BAUD
0050 *
0051 *****
0052 * THIS BOOTSTRAP PROGRAM ALLOWS THE USER TO
0053 * DOWNLOAD A PROGRAM OF 0 - 2048 BYTES.
0054 * THE PROGRAM MUST START AT $0000.
0055 * EACH BYTE OF THE PROGRAM IS RECEIVED BY THE SCI.
0056 * THE FIRST BYTE ESTABLISHES BAUD RATE.
0057 * THEN THE PROGRAM IS DOWNLOADED STARTING WITH
0058 * THE $0000 BYTE AND WORKING UP TOWARD THE $03FF
0059 * A DELAY OF FOUR WORD TIMES (AT EITHER BAUD RATE)
0060 * CAUSES THE RECEPTION OF CHARACTERS TO STOP AND
0061 * A JUMP TO $0000.
0062 *
0063 * THE TRANSMITTER WILL BE USED FOR THE PURPOSE
0064 * OF COMMUNICATION TO THE OUTSIDE WORLD.
0065 *
0066 *****

```

Freescale Semiconductor, Inc.

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0067 bf00                                ORG    $BF00
0068                                     *
0069 bf00                                BEGIN  EQU    *
0070                                     * INIT STACK
0071 bf00 8e 07 ff                        LDS    #RAMEND
0072                                     * INIT X REG FOR INDEXED ACCESS TO REGISTERS
0073 bf03 ce 10 00                        LDX    #$1000
0074                                     *****
0075                                     * PUT PORT D IN WIRE OR MODE
0076 bf06 1c 28 20                        BSET   SPCR,X $20
0077                                     * INIT SCI AND RESTART BAUD DIVIDER CHAIN
0078 bf09 cc a2 0c                        LDD    #A20C                                DIV BY 16
0079 bf0c a7 2b                            STAA   BAUD,X
0080                                     * RECEIVER & TRANSMITTER ENABLED
0081 bf0e e7 2d                            STAB   SCCR2,X
0082                                     *****
0083                                     * TEST THE SECURITY BIT
0084 bf10 1e 3f 08 5e                      BRSET  CONFIG,X $08 NOSEC
0085                                     *****
0086                                     * WE ARE IN SECURITY MODE
0087                                     *
0088                                     * OUTPUT $FF ON TRANSMITTER
0089 bf14                                AGAIN  EQU    *
0090 bf14 a6 2e                            LDAA   SCSR,X
0091 bf16 86 ff                            LDAA   #$FF
0092 bf18 a7 2f                            STAA   SCDAT,X
0093                                     * ACCA NOW IS SET FOR $FF
0094                                     *
0095                                     * ERASE EEPROM:
0096                                     * TURN OFF BLOCK PROTECT
0097 bf1a 6f 35                            CLR    BPROT,X
0098                                     * SET ERASE AND EELAT BITS BEFORE USING "ERASE"
0099 bf1c 54                                LSRB                                     CHANGE $0C TO $06
0100 bf1d e7 3b                            STAB   PPROG,X
0101 bf1f f7 b6 00                         STAB   EEPROM                                WRITE EEPROM LOCATION
0102 bf22 8d 34                            BSR    ERASE
0103                                     * ACCB IS NOW SET FOR $06
0104                                     *
0105                                     *****
0106                                     * ERASE CYCLE IS COMPLETE
0107                                     *
0108                                     * IF THE EEPROM IS NOT ENABLED,
0109                                     * WE CAN'T CHECK THAT THE EEPROM IS ERASED
0110 bf24 1f 3f 01 11                      BRCLR  CONFIG,X $01 NOEE
0111                                     * EEPROM IS ON,
0112                                     * NOW CHECK THAT THE EEPROM IS ERASED
0113 bf28 18 ce b6 00                        LDY    #EEPSTR
0114 bf2c                                LOOP   EQU    *
0115 bf2c 18 a1 00                        CMPA   0,Y                                (A = $FF)
0116                                     * ANY UNERASED BYTE SENDS US BACK TO ERASE AGAIN
0117 bf2f 26 e3                            BNE    AGAIN
0118 bf31 18 08                            INY
0119 bf33 18 8c b8 00                      CPY    #EEPEND+1
0120 bf37 26 f3                            BNE    LOOP
0121 bf39                                NOEE   EQU    *
0122                                     *****
0123                                     * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES
0124                                     * WHICH ARE USED BY THE STACK & $0000 WHICH IS
0125                                     * LEFT INTACT
0126                                     *
0127 bf39                                ERAM   EQU    *
0128 bf39 3c                                PSHX
0129 bf3a ce 07 fd                        LDX    #RAMEND-2
0130 bf3d                                LOPl   EQU    *
0131 bf3d a7 00                            STAA   $00,X
0132 bf3f 09                                DEX
0133 bf40 26 fb                            BNE    LOPl
0134                                     *                                PULX                                <<<<

```

```

0135          * DO NOT SEPARATE RAM AND EPROM ROUTINES WITHOUT
0136          * FIXING THE STACK (PULX)
0137          *****
0138          * CONFIRM THAT EPROM IS ERASED
0139          *
0140          bf42 ce c0 00          LDX      #EP1STR
0141          bf45 8d 1f          LOP2    BSR      TSTLP
0142          bf47 26 fc          BNE     LOP2
0143          bf49 ce 70 00          LDX      #EP2STR
0144          bf4c a1 00          LOP3    CMPA    0,X          (A=$FF)
0145          bf4e 8d 16          BSR     TSTLP
0146          bf50 8c b0 00          CPX     #EP2END+1
0147          bf53 26 f7          BNE     LOP3
0148          bf55 38          PULX           <<<<
0149          bf56 20 14          BRA     ECONFG
0150
0151          *****
0152          * BOOTLOADER SUBROUTINES
0153          *
0154          * EEPROM ERASE SUBROUTINE
0155          *
0156          * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
0157          * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
0158          * IS TO BE ERASED
0159          *
0160          * ENTRY X=$1000
0161          * EXIT X=$1000, Y=$0000
0162          *
0163          bf58          ERASE    EQU     *
0164          * SET EEPGM BIT
0165          bf58 1c 3b 01          BSET    PPROG,X $01
0166          * 10 MILLISEC DELAY @ 2.1 MHZ
0167          bf5b 18 ce 0b b8          LDY     #3000
0168          bf5f 18 09          BK1     DEY
0169          bf61 26 fc          BNE     BK1
0170          * TURN OFF ERASE AND EELAT BITS
0171          bf63 6f 3b          CLR     PPROG,X
0172          bf65 39          RTS
0173          *
0174          * EPROM TEST SUBROUTINE
0175          *
0176          bf66 a1 00          TSTLP   CMPA    0,X          (A=$FF)
0177          bf68 26 fe          BNE     *          (Loop here forever)
0178          bf6a 08          TSTOK   INX
0179          bf6b 39          RTS
0180
0181          *****
0182          * NOW ERASE CONFIG REGISTER
0183          *
0184          bf6c          ECONFG    EQU     *
0185          * SET ERASE AND EELAT BITS
0186          bf6c e7 3b          STAB    PPROG,X          (B STILL = $06)
0187          * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
0188          bf6e e7 3f          STAB    CONFIG,X
0189          bf70 8d e6          BSR     ERASE
0190          *****
0191          * ERASE CYCLE IS COMPLETE
0192          *
0193          *****
0194          * NON-SECURITY AND SECURITY MODES MEET HERE
0195          *
0196          bf72          NOSEC     EQU     *
0197          *****
0198          * SET UP DELAY FOR FASTEST BAUD RATE
0199          bf72 cc 02 1b          LDD     #DELAYF
0200          bf75 ed 16          STD     TOC1,X
0201          *****
0202          * SEND BREAK TO SIGNAL START OF DOWNLOAD
0203          bf77 1c 2d 01          BSET    SCCR2,X $01

```

EB422

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0204                                     * CLEAR BREAK AS SOON AS START BIT IS DETECTED
0205 bf7a 1e 08 01 fc                     BRSET  PORTD,X $01 *
0206 bf7e 1d 2d 01                       BCLR   SCCR2,X $01          CLEAR BREAK
0207                                     * WAIT FOR FIRST CHARACTER (USERS SEND $$$)
0208 bf81 1f 2e 20 fc                     BRCLR  SCSR,X $20 *        WAIT FOR RDRF
0209 bf85 a6 2f                           LDAA   SCDAT,X            READ DATA
0210                                     * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
0211 bf87 26 03                           BNE    NOTZERO
0212 bf89 7e b6 00                         JMP    EEPSTR
0213 bf8c                                     NOTZERO EQU *
0214                                     * IF DATA = $FF, THEN /16 IS CORRECT BAUD
0215 bf8c 81 ff                           CMPA   #$FF
0216 bf8e 27 08                           BEQ    BAUDOK
0217                                     * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
0218 bf90 1c 2b 33                         BSET   BAUD,X $33
0219                                     * SET UP DELAY FOR SLOWER BAUD RATE
0220 bf93 cc 0d b0                         LDD   #DELAYS
0221 bf96 ed 16                             STD   TOC1,X
0222                                     *
0223 bf98 BAUDOK EQU *
0224 bf98 18 ce 00 00                       LDY   #RAMSTR            PNTR TO START OF RAM
0225                                     *
0226                                     * TIME EACH BYTE
0227 bf9c ec 16                             WAIT  LDD   TOC1,X        PUT DELAY TIME IN ACCD
0228                                     *
0229 bf9e 1e 2e 20 07                       WTLOOP BRSET  SCSR,X $20 NEWONE
0230 bfa2 8f                                XGDX                                     DELAY INTO X
0231 bfa3 09                                DEX                                     DECREMENT DELAY
0232 bfa4 8f                                XGDX                                     RETURN DELAY TO ACCD
0233 bfa5 26 f7                             BNE    WTLOOP
0234 bfa7 20 0f                             BRA    STAR
0235                                     * DID NOT TIME OUT
0236 bfa9 NEWONE EQU *
0237                                     * READ IN BYTE AND PUT INTO RAM
0238 bfa9 a6 2f                             LDAA  SCDAT,X
0239 bfab 18 a7 00                         STAA  $00,Y
0240 bfae a7 2f                             STAA  SCDAT,X            HANDSHAKE
0241 bfb0 18 08                             INY
0242 bfb2 18 8c 08 00                       CPY   #RAMEND+1
0243 bfb6 26 e4                             BNE   WAIT
0244                                     *****
0245                                     * START USER'S PROGRAM
0246                                     *
0247 bfb8 STAR EQU *
0248 bfb8 7e 00 00                           JMP   RAMSTR
0249                                     *****
0250                                     * FILL UNUSED BYTES WITH ZERO
0251                                     *
0252 bbbb BSZ $BFBB-*
0253                                     *
0254                                     *****
0255                                     * REVISION LEVEL IN ASCII
0256                                     * (ORG $BFBB)
0257 bbbb 41 FCC "A"
0258                                     *****
0259                                     * MASK I.D. ($0000 FOR EPROM PARTS)
0260                                     * (ORG $BFBC)
0261 bbbc 00 00 FDB $0000
0262                                     *****
0263                                     * 711E32 I.D. - can be used to determine MCU type
0264                                     * Bit 15 is a 0 if the part is ROM (or ROMless,
0265                                     * 1 -> EPROM)
0266                                     * Bit 14 is a 0 if unsecured
0267                                     * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
0268                                     * Bits 8 - Bit 6 are $7 => last six bits are number
0269                                     * Bits 5 - Bit 0 are last digit of part number
0270                                     * (note: $45 = E in ASCII)
0271                                     *
0272                                     * 15 14 13 9 8 6 5 0

```

```

0273          * i.e. 1 1 00101 111 100000 (CBE0)
0274          *      EP SEC 'E' nul 32
0275          *
0276          *      (ORG$BFBE)
0277 bfbe cb e0          FDB      %1100101111100000
0278          *****
0279          * VECTORS      ($BDC0)
0280          *
0281 bf00 00 00          FDB      $0000          reserved
0282 bf02 00 00          FDB      $0000          reserved
0283 bf04 00 00          FDB      $0000          reserved
0284 bf06 00 00          FDB      $0000          reserved
0285 bf08 00 00          FDB      $0000          reserved
0286 bf0a 00 00          FDB      $0000          reserved
0287 bf0c 00 00          FDB      $0000          reserved
0288 bf0e 00 00          FDB      $0000          reserved
0289 bf10 00 00          FDB      $0000          reserved
0290 bf12 00 00          FDB      $0000          reserved
0291 bf14 00 00          FDB      $0000          reserved
0292 bf16 00 c4          FDB      $100-60          SCI
0293 bf18 00 c7          FDB      $100-57          SPI
0294 bf1a 00 ca          FDB      $100-54          PULSE ACCUM INPUT EDGE
0295 bf1c 00 cd          FDB      $100-51          PULSE ACCUM OVERFLOW
0296 bf1e 00 d0          FDB      $100-48          TIMER OVERFLOW
0297 bf20 00 d3          FDB      $100-45          TIMER OUTPUT COMPARE 5
0298 bf22 00 d6          FDB      $100-42          TIMER OUTPUT COMPARE 4
0299 bf24 00 d9          FDB      $100-39          TIMER OUTPUT COMPARE 3
0300 bf26 00 dc          FDB      $100-36          TIMER OUTPUT COMPARE 2
0301 bf28 00 df          FDB      $100-33          TIMER OUTPUT COMPARE 1
0302 bf2a 00 e2          FDB      $100-30          TIMER INPUT CAPTURE 3
0303 bf2c 00 e5          FDB      $100-27          TIMER INPUT CAPTURE 2
0304 bf2e 00 e8          FDB      $100-24          TIMER INPUT CAPTURE 1
0305 bf30 00 eb          FDB      $100-21          REAL TIME INT
0306 bf32 00 ee          FDB      $100-18          IRQ
0307 bf34 00 f1          FDB      $100-15          XIRQ
0308 bf36 00 f4          FDB      $100-12          SWI
0309 bf38 00 f7          FDB      $100-9          ILLEGAL OP-CODE
0310 bf3a 00 fa          FDB      $100-6          COP FAIL
0311 bf3c 00 fd          FDB      $100-3          CLOCK MONITOR
0312 bf3e bf 00          FDB      BEGIN          RESET
0313
0314
0315          ***** TBRE32.ASC 30/6/93 *****
0316          * Motorola Copyright 1988,1990,1992,1993          *
0317          * MCU resident, Interrupt driven Communication routines for 68HC11          *
0318          * monitor.Provides low level memory & stack read/write operations          *
0319          *          *
0320          *      This talker DOES NOT use XIRQ          *
0321          *      -----          *
0322          *          *
0323          * N.B. TBRE32 is designed to work with the 68HC11E32 or other          *
0324          * compatible MCU types. This version of the TALKER is designed to          *
0325          * execute from MC68HC711E32 Boot ROM.          *
0326          * To initiate communication with TBRE32, the standard bootloader          *
0327          * must be used to initialise the redirected vector table and the          *
0328          * cause a jump to USERSTART          *
0329          *          *
0330          * CONSTANTS          *
0331 be40          TALKBASE          equ          $BE40
0332 00c4          BOOTVECT          equ          $00C4          Start of boot vectors
0333 03ff          STACK          equ          $03FF          User may alter this
0334 1000          REGBASE          equ          $1000
0335          *
0336 00c4          JSCI          equ          $00C4
0337 00f1          JXIRQ          equ          $00F1
0338 00f4          JSWI          equ          $00F4
0339 00f7          JILLOP          equ          $00F7
0340 00fa          JCOP          equ          $00FA
0341 007e          JMPEXT          equ          $7E          Mnemonic for jump ext

```

Freescale Semiconductor, Inc.

Engineering Bulletin

```

0342 004a          BRKCODE      equ    $4A          Break point code
0343 004a          BRKACK      equ    $4A          Break point ack
0344              *
0345              * REGISTERS
0346 002f          SCDR        equ    $2F
0347              *
0348 0020          RDRF        equ    $20
0349 0080          TDRE        equ    $80
0350 0008          OR         equ    $08
0351 0002          FE         equ    $02
0352              *
0353              * PROGRAM
0354 be40          org      TALKBASE
0355              *
0356 be40          TLKRSTART  EQU    *          Initialise SCI int
0357 be40 86 7e          LDAA      #JMPEXT
0358 be42 97 c4          STAA      JSCI
0359 be44 ce be 5e          LDX      #SCISRV
0360 be47 df c5          STX      JSCI+1
0361 be49          USERSTART EQU    *
0362 be49 8e 03 ff          LDS      #STACK
0363 be4c ce 10 00          LDX      #REGBASE
0364 be4f 6f 2c          CLR      SCCR1,X
0365 be51 cc 30 2c          LDD      #$302C          Init SCI to 9600 baud,
0366 be54 a7 2b          STAA      BAUD,X          no parity, no int
0367 be56 e7 2d          STAB     SCCR2,X          & enable SCI tx & rx.
0368 be58 86 40          LDAA      #$40          Enable STOP, and I bit
0369 be5a 06          TAP
0370              *
0371 be5b 7e be 5b          IDLE     JMP      IDLE          Now hang around for
0372              * A RESET from host changes above jump destination to
0373              * start of user code.
0374 be5e          SCISRV    EQU    *          On detecting int
0375 be5e b6 10 2e          LDAA      SCSR+REGBASE    assume rx caused it.
0376 be61 84 20          ANDA     #RDRF
0377 be63 27 f9          BEQ      SCISRV          otherwise loop here
0378              *
0379 be65          RXSRV   EQU    *          Process received data.
0380 be65 b6 10 2f          LDAA      SCDR+REGBASE    Read cmd byte, & tx it
0381 be68 43          COMA
0382 be69 8d 40          BSR      OUTSCI          as ack to host.
0383 be6b 2a 4b          BPL      INH1          Bit 7 set? => inherent
0384 be6d 8d 2d          BSR      INSCI          else B = byte count
0385 be6f 8f          XGDX
0386 be70 8d 2a          BSR      INSCI          Save cmd and byte cnt.
0387 be72 17          TBA          Read high address byte
0388 be73 8d 27          BSR      INSCI          into ACCA
0389 be75 8f          XGDX          then B = low addr byte
0390 be76 81 fe          CMPA     #$FE          A=cmd; B=cnt; X=addr
0391 be78 26 0d          BNE      RXSRV1          If cmd is mem read ...
0392              *
0393 be7a          TREADMEM EQU    *          REPEAT
0394 be7a a6 00          LDAA      ,X          read required address
0395 be7c 8d 2d          BSR      OUTSCI          send it to host
0396 be7e 17          TBA          (save byte count)
0397 be7f 8d 1b          BSR      INSCI          and wait for acknowledge
0398 be81 16          TAB          (restore byte count)
0399 be82 08          INX          Increment address
0400 be83 5a          DECB     Decrement byte count
0401 be84 26 f4          BNE      TREADMEM          UNTIL all done
0402 be86 3b          RTI          & return
0403              *
0404 be87          RXSRV1  EQU    *
0405 be87 81 be          CMPA     #$BE
0406 be89 26 10          BNE      RXSRVEX          If cmd is mem write...
0407              *
0408 be8b 17          TBA          move byte count to A
0409 be8c          TWRTMEM EQU    *          REPEAT
0410 be8c 8d 0e          BSR      INSCI          Read next byte into B

```

0411	be8e	e7	00		STAB	,X		and store at req addr
0412	be90	8d	18		BSR	EPRG		
0413	be92	e6	00		LDAB	,X		Read stored byte and
0414	be94	f7	10	2f	STAB	SCDR+REGBASE		echo it back to host,
0415	be97	08			INX			
0416	be98	4a			DECA			Decrement byte count
0417	be99	26	f1		BNE	TWRITMEM		UNTIL all done
0418	be9b				RXSRVEX	EQU	*	and return
0419	be9b	3b			NULLSRV	RTI		
0420					*			
0421	be9c				INSCI	EQU	*	
0422	be9c	f6	10	2e	LDAB	SCSR+REGBASE		Wait for RDRF=1
0423	be9f	c5	00		BITB	\$(FE+OR)		If break detected then
0424	bea1	26	9d		BNE	TLKRSTART		restart talker.
0425	bea3	c4	20		ANDB	#RDRF		
0426	bea5	27	f5		BEQ	INSCI		
0427	bea7	f6	10	2f	LDAB	SCDR+REGBASE		read data received
0428	beaa	39			EPRG	RTS		& return B = data
0429					*			
0430	beab				OUTSCI	EQU	*	Only Y modified.
0431	beab	18	8f		XGDY			Enter with A = tx data
0432	bead	b6	10	2e	OUTSCI1	LDAA	SCSR+REGBASE	
0433	beb0	2a	fb		BPL	OUTSCI1		MS bit is TDRE flag
0434	beb2	18	8f		XGDY			
0435	beb4	b7	10	2f	STAA	SCDR+REGBASE		Important: Updates CCR
0436	beb7	39			RTS			
0437					*			
0438	beb8				INH1	EQU	*	
0439	beb8	81	7e		CMPA	#\$7E		If cmd is read regs ..
0440	beba	26	0c		BNE	INH2		
0441					*			
0442	becb	30			INH1A	TSX		Move SP to X
0443	becd	8f			XGDY			then to ACCD
0444	bebe	8d	eb		BSR	OUTSCI		send SP high byte 1st
0445	bec0	17			TBA			
0446	bec1	8d	e8		BSR	OUTSCI		then low byte
0447	bec3	30			TSX			Restore X (=SP)
0448	bec4	c6	09		LDAB	#9		Tx 9 bytes on stack
0449	bec6	20	b2		BRA	TREADMEM		i.e.CCR,ACCB,ACCA,IXH
0450					*			,IXL,IYH,IYL,PCH,PCL
0451	bec8				INH2	EQU	*	
0452	bec8	81	3e		CMPA	#\$3E		If cmd is write regs..
0453	beca	26	12		BNE	SWISRV1		
0454					*			
0455	becc	8d	ce		BSR	INSCI		Get SP High byte first
0456	becd	17			TBA			
0457	becf	8d	cb		BSR	INSCI		
0458	bed1	8f			XGDY			Move to X reg
0459	bed2	35			TXS			and copy to SP
0460	bed3	86	09		LDAA	#9		Then 9 bytes to stack
0461	bed5	20	b5		BRA	TWRITMEM		
0462					*			
0463	bed7				SWISRV	EQU	*	Breakpoints by SWI
0464	bed7	86	4a		LDAA	#BRKCODE		Force host to proc BR
0465	bed9	8d	d0		BSR	OUTSCI		by sending it BREAK
0466	bedb	0e			SWIIDLE	CLI		
0467	bedc	20	fd		BRA	SWIIDLE		then wait for response
0468					*			
0469	bede				SWISRV1	EQU	*	
0470	bede	81	4a		CMPA	#BRKACK		If host acknowledges
0471	bee0	26	b9		BNE	RXSRVEX		
0472	bee2	30			TSX			move SP to SWI stack &
0473	bee3	c6	09		LDAB	#9		
0474	bee5	3a			ABX			Send user code
0475	bee6	35			TXS			breakpoint return
0476	bee7	ec	07		LDD	7,X		address to host
0477	bee9	8d	c0		BSR	OUTSCI		(high byte first)
0478	beeb	17			TBA			
0479	beec	8d	bd		BSR	OUTSCI		(low byte next)

EB422

Engineering Bulletin

```

0480 beee cc be db          LDD    #SWIIDLE          force idle loop on
0481 bef1 ed 07            STD    7,X              return from breakpoint
0482 bef3 20 c7            BRA    INH1A           but first return all
0483                        *                               MCU registers to host
0484                        *                               *****
0485                        * FILL UNUSED BYTES WITH ZERO
0486                        *
0487 bef5 00 00 00 00 00  BSZ    $BF00-*
      00 00 00 00 00

0488                        *
0489                        *                               *****
0490
0491                        END
0492
0493

```

Listing 13. MC68HC11E32 Secured Bootloader ROM Listing

```

0001                        *****
0002                        * FIRMWARE FOR SECURED 68HC11E32 - 24 SEPT 1993
0003                        *
0004                        *                               *****
0005                        * SECURED EPROM VERSION
0006                        *
0007                        * Part I.D. at $BFBE is $4BE0. {E32}
0008                        *                               *****
0009                        * REVISION A 24 SEPT 1993
0010                        * BASED ON SECURE 711E32 BOOTLOADER 16 JULY 93
0011                        * REMOVED CHECK ON EPROM
0012                        *
0013                        *                               *****
0014                        * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
0015                        * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
0016                        * OF AT LEAST FOUR WORD TIMES OCCURS
0017                        *
0018                        * EQUATES FOR USE WITH INDEX OFFSET = $1000
0019                        *
0020                        *
0021 0008      PORTD      EQU    $08
0022 0009      DDRD       EQU    $09
0023 0016      TOC1       EQU    $16          [STORAGE (POOR STYLE)]
0024 0028      SPCR       EQU    $28          (FOR DWOM BIT)
0025 002b      BAUD      EQU    $2B
0026 002c      SCCR1     EQU    $2C
0027 002d      SCCR2     EQU    $2D
0028 002e      SCSR      EQU    $2E
0029 002f      SCDAT     EQU    $2F
0030 0035      BPROT     EQU    $35
0031 003b      PPROG     EQU    $3B
0032 003e      TEST1    EQU    $3E
0033 003f      CONFIG    EQU    $3F
0034                        *
0035                        * MORE EQUATES
0036                        *
0037 b600      EEPSTR     EQU    $B600      START OF EEPROM
0038 b7ff      EEPEND     EQU    $B7FF      END OF EEPROM
0039                        *
0040 0000      RAMSTR     EQU    $0000
0041 07ff      RAMEND     EQU    $07FF
0042                        *
0043 0db0      DELAYS     EQU    3504      DELAY AT SLOW BAUD
0044 021b      DELAYF     EQU    539      DELAY AT FAST BAUD
0045                        *

```

Freescale Semiconductor, Inc.


```

0046          *****
0047          * THIS BOOTSTRAP PROGRAM ALLOWS THE USER TO
0048          * DOWNLOAD A PROGRAM OF 0 - 2048 BYTES.
0049          * THE PROGRAM MUST START AT $0000.
0050          * EACH BYTE OF THE PROGRAM IS RECEIVED BY THE SCI.
0051          * THE FIRST BYTE ESTABLISHES BAUD RATE.
0052          * THEN THE PROGRAM IS DOWNLOADED STARTING WITH
0053          * THE $0000 BYTE AND WORKING UP TOWARD THE END OF RAM
0054          * A DELAY OF FOUR WORD TIMES (AT EITHER BAUD RATE)
0055          * CAUSES THE RECEPTION OF CHARACTERS TO STOP AND
0056          * A JUMP TO $0000.
0057          *
0058          * THE TRANSMITTER WILL BE USED FOR THE PURPOSE
0059          * OF COMMUNICATION TO THE OUTSIDE WORLD.
0060          *
0061          *****
0062 bf00          ORG      $BF00
0063          *
0064 bf00          BEGIN    EQU      *
0065          * INIT STACK
0066 bf00 8e 07 ff          LDS      #RAMEND
0067          * INIT X REG FOR INDEXED ACCESS TO REGISTERS
0068 bf03 ce 10 00          LDX      #$1000
0069          *****
0070          * PUT PORT D IN WIRE OR MODE
0071 bf06 1c 28 20          BSET   SPCR,X $20
0072          * INIT SCI AND RESTART BAUD DIVIDER CHAIN
0073 bf09 cc a2 0c          LDD      #$A20C          DIV BY 16
0074 bf0c a7 2b          STAA   BAUD,X
0075          * RECEIVER & TRANSMITTER ENABLED
0076 bf0e e7 2d          STAB   SCCR2,X
0077          *****
0078          * TEST THE SECURITY BIT
0079 bf10 1e 3f 08 45          BRSET  CONFIG,X $08 NOSEC
0080          *****
0081          * WE ARE IN SECURITY MODE
0082          *
0083          * OUTPUT $FF ON TRANSMITTER
0084 bf14          AGAIN    EQU      *
0085 bf14 a6 2e          LDAA   SCSR,X
0086 bf16 86 ff          LDAA   #$FF
0087 bf18 a7 2f          STAA   SCDAT,X
0088          * ACCA NOW IS SET FOR $FF
0089          *
0090          * ERASE EEPROM:
0091          * TURN OFF BLOCK PROTECT
0092 bf1a 6f 35          CLR    BPROT,X
0093          * SET ERASE AND EELAT BITS BEFORE USING "ERASE"
0094 bf1c 54          LSRB          CHANGE $0C TO $06
0095 bf1d e7 3b          STAB   PPROG,X
0096 bf1f f7 b6 00          STAB   EEPSTR          WRITE EEPROM LOCATION
0097 bf22 8d 21          BSR    ERASE
0098          * ACCB IS NOW SET FOR $06
0099          *
0100          *****
0101          * ERASE CYCLE IS COMPLETE
0102          *
0103          * IF THE EEPROM IS NOT ENABLED,
0104          * WE CAN'T CHECK THAT THE EEPROM IS ERASED
0105 bf24 1f 3f 01 11          BRCLR  CONFIG,X $01 NOEE
0106          * EEPROM IS ON,
0107          * NOW CHECK THAT THE EEPROM IS ERASED
0108 bf28 18 ce b6 00          LDY    #EEPSTR
0109 bf2c          LOOP    EQU      *
0110 bf2c 18 a1 00          CMPA   0,Y          (A = $FF)
0111          * ANY UNERASED BYTE SENDS US BACK TO ERASE AGAIN
0112 bf2f 26 e3          BNE    AGAIN
0113 bf31 18 08          INY
0114 bf33 18 8c b8 00          CPY    #EEPEND+1

```

EB422

Engineering Bulletin

```

0115 bf37 26 f3          BNE    LOOP
0116 bf39                NOEE    EQU    *
0117                    *****
0118                    * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES
0119                    * WHICH ARE USED BY THE STACK & $0000 WHICH IS
0120                    * LEFT INTACT
0121                    *
0122 bf39                ERAM    EQU    *
0123 bf39 3c            PSHX
0124 bf3a ce 07 fd      LDX    #RAMEND-2
0125 bf3d                LOPl    EQU    *
0126 bf3d a7 00        STAA   $00,X
0127 bf3f 09            DEX
0128 bf40 26 fb        BNE    LOPl
0129 bf42 38            PULX
0130 bf43 20 0e        BRA    ECONFG
0131
0132                    *****
0133                    * BOOTLOADER SUBROUTINES
0134                    *
0135                    * EEPROM ERASE SUBROUTINE
0136                    *
0137                    * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
0138                    * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
0139                    * IS TO BE ERASED
0140                    *
0141                    * ENTRY X=$1000
0142                    * EXIT X=$1000, Y=$0000
0143                    *
0144 bf45                ERASE    EQU    *
0145                    * SET EEGM BIT
0146 bf45 1c 3b 01      BSET   PPROG,X $01
0147                    * 10 MILLISEC DELAY @ 2.1 MHZ
0148 bf48 18 ce 0b b8  LDY    #3000
0149 bf4c 18 09        BK1    DEY
0150 bf4e 26 fc        BNE    BK1
0151                    * TURN OFF ERASE AND EELAT BITS
0152 bf50 6f 3b        CLR    PPROG,X
0153 bf52 39            RTS
0154
0155                    *****
0156                    * NOW ERASE CONFIG REGISTER
0157                    *
0158 bf53                ECONFG   EQU    *
0159                    * SET ERASE AND EELAT BITS
0160 bf53 e7 3b        STAB   PPROG,X          (B STILL = $06)
0161                    * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
0162 bf55 e7 3f        STAB   CONFIG,X
0163 bf57 8d ec        BSR    ERASE
0164                    *****
0165                    * ERASE CYCLE IS COMPLETE
0166                    *
0167                    *****
0168                    * NON-SECURITY AND SECURITY MODES MEET HERE
0169                    *
0170 bf59                NOSEC    EQU    *
0171                    *****
0172                    * SET UP DELAY FOR FASTEST BAUD RATE
0173 bf59 cc 02 1b      LDD    #DELAYF
0174 bf5c ed 16        STD    TOC1,X
0175                    *****
0176                    * SEND BREAK TO SIGNAL START OF DOWNLOAD
0177 bf5e 1c 2d 01      BSET   SCCR2,X $01
0178                    * CLEAR BREAK AS SOON AS START BIT IS DETECTED
0179 bf61 1e 08 01 fc  BRSET  PORTD,X $01 *
0180 bf65 1d 2d 01      BCLR  SCCR2,X $01          CLEAR BREAK
0181                    * WAIT FOR FIRST CHARACTER (USERS SEND $FF)
0182 bf68 1f 2e 20 fc  BRCLR  SCSR,X $20 *          WAIT FOR RDRF
0183 bf6c a6 2f        LDAA   SCDAT,X          READ DATA

```

```

0184          * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
0185 bf6e 26 03          BNE    NOTZERO
0186 bf70 7e b6 00      JMP    EEPSTR
0187 bf73          NOTZERO EQU    *
0188          * IF DATA = $FF, THEN /16 IS CORRECT BAUD
0189 bf73 81 ff          CMPA   #$FF
0190 bf75 27 08          BEQ    BAUDOK
0191          * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
0192 bf77 1c 2b 33      BSET   BAUD,X $33
0193          * SET UP DELAY FOR SLOWER BAUD RATE
0194 bf7a cc 0d b0      LDD    #DELAYS
0195 bf7d ed 16          STD    TOC1,X
0196          *
0197 bf7f          BAUDOK   EQU    *
0198 bf7f 18 ce 00 00    LDY    #RAMSTR          PNTR TO START OF RAM
0199          *
0200          * TIME EACH BYTE
0201 bf83 ec 16          WAIT    LDD    TOC1,X          PUT DELAY TIME IN ACCD
0202          *
0203 bf85 1e 2e 20 07    WTLOOP  BRSET   SCSR,X $20 NEWONE
0204 bf89 8f          XGDX          DELAY INTO X
0205 bf8a 09          DEX          DECREMENT DELAY
0206 bf8b 8f          XGDX          RETURN DELAY TO ACCD
0207 bf8c 26 f7          BNE    WTLOOP
0208 bf8e 20 0f          BRA    STAR
0209          * DID NOT TIME OUT
0210 bf90          NEWONE  EQU    *
0211          * READ IN BYTE AND PUT INTO RAM
0212 bf90 a6 2f          LDAA   SCDAT,X
0213 bf92 18 a7 00      STAA  $00,Y
0214 bf95 a7 2f          STAA  SCDAT,X          HANDSHAKE
0215 bf97 18 08          INY
0216 bf99 18 8c 08 00  CPY   #RAMEND+1
0217 bf9d 26 e4          BNE    WAIT
0218          *****
0219          * START USER'S PROGRAM
0220          *
0221 bf9f          STAR    EQU    *
0222 bf9f 7e 00 00      JMP    RAMSTR
0223          *****
0224          * FILL UNUSED BYTES WITH ZERO
0225          *
0226 bfa2 00 00 00 00 00 00 00 00          BSZ    $BFBF-*
0227          00 00 00 00 00 00
0228          00 00 00 00 00 00
0229          00 00 00 00 00 00
0230          00
0231          *
0232          *****
0233          * REVISION LEVEL IN ASCII
0234          * (ORG $BFB)
0235 bfbb 41          FCC    "A"
0236          *****
0237          * MASK I.D. ($0000 FOR EPROM PARTS)
0238          * (ORG $BFB)
0239          * FDB $0000
0240          *****
0241          * 11E32 I.D. - can be used to determine MCU type
0242          * Bit 15 is a 0 if the part is ROM (or ROMless,
0243          * 1 -> EPROM)
0244          * Bit 14 is a 0 if unsecured
0245          * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
0246          * Bits 8 - Bit 6 are $7 => last six bits are number
0247          * Bits 5 - Bit 0 are last digit of part number
0248          * (note: $45 = E in ASCII)
0249          *
0250          *      15 14 13   9 8 6 5   0
0251          * i.e. 0 1 00101 111 100000          (4BE0)
0252          *      ROM SEC 'E' nul 32

```

Engineering Bulletin

Freescale Semiconductor, Inc.

```

0249          *
0250          *          (ORG   $BFB)
0251 bfbe 4b e0          FDB   %0100101111100000
0252          *
0253          * VECTORS ($BDC0)
0254          *
0255 bfc0 00 00          FDB   $0000          reserved
0256 bfc2 00 00          FDB   $0000          reserved
0257 bfc4 00 00          FDB   $0000          reserved
0258 bfc6 00 00          FDB   $0000          reserved
0259 bfc8 00 00          FDB   $0000          reserved
0260 bfca 00 00          FDB   $0000          reserved
0261 bfcc 00 00          FDB   $0000          reserved
0262 bfce 00 00          FDB   $0000          reserved
0263 bfd0 00 00          FDB   $0000          reserved
0264 bfd2 00 00          FDB   $0000          reserved
0265 bfd4 00 00          FDB   $0000          reserved
0266 bfd6 00 c4          FDB   $100-60          SCI
0267 bfd8 00 c7          FDB   $100-57          SPI
0268 bfda 00 ca          FDB   $100-54          PULSE ACCUM INPUT EDGE
0269 bfdc 00 cd          FDB   $100-51          PULSE ACCUM OVERFLOW
0270 bfde 00 d0          FDB   $100-48          TIMER OVERFLOW
0271 bfe0 00 d3          FDB   $100-45          TIMER OUTPUT COMPARE 5
0272 bfe2 00 d6          FDB   $100-42          TIMER OUTPUT COMPARE 4
0273 bfe4 00 d9          FDB   $100-39          TIMER OUTPUT COMPARE 3
0274 bfe6 00 dc          FDB   $100-36          TIMER OUTPUT COMPARE 2
0275 bfe8 00 df          FDB   $100-33          TIMER OUTPUT COMPARE 1
0276 bfea 00 e2          FDB   $100-30          TIMER INPUT CAPTURE 3
0277 bfec 00 e5          FDB   $100-27          TIMER INPUT CAPTURE 2
0278 bfef 00 e8          FDB   $100-24          TIMER INPUT CAPTURE 1
0279 bff0 00 eb          FDB   $100-21          REAL TIME INT
0280 bff2 00 ee          FDB   $100-18          IRQ
0281 bff4 00 f1          FDB   $100-15          XIRQ
0282 bff6 00 f4          FDB   $100-12          SWI
0283 bff8 00 f7          FDB   $100-9          ILLEGAL OP-CODE
0284 bffa 00 fa          FDB   $100-6          COP FAIL
0285 bffc 00 fd          FDB   $100-3          CLOCK MONITOR
0286 bffe bf 00          FDB   BEGIN          RESET
0287
0288
0289          *
0290          * ***** TBRE32.ASC 30/6/93 *****
0291          * Motorola Copyright 1988,1990,1992,1993
0292          * MCU resident,InterruptdrivenCommunicationroutines for 68HC11 mon-
0293          * itor.Provides low level memory and stack read/write operations.
0294          *
0295          * This talker DOES NOT use XIRQ
0296          * -----
0297          *
0298          * N.B.TBRE32 is designed to work with the 68HC11E32 or other
0299          * compatible MCU types.This version of the TALKER is designed to
0300          * execute from MC68HC11E32 Boot ROM.
0301          * To initiate communication with TBRE32,the standard bootloader
0302          * must be used to initialise the redirected vector table and then
0303          * cause a jump to USERSTART
0304          *
0305          * CONSTANTS
0306 be40          TALKBASE          equ   $BE40
0307 00c4          BOOTVECT          equ   $00C4          Start of boot vectors
0308 03ff          STACK            equ   $03FF          User may alter this
0309 1000          REGBASE          equ   $1000
0310          *
0311 00c4          JSCI            equ   $00C4
0312 00f1          JXIRQ          equ   $00F1
0313 00f4          JSWI          equ   $00F4
0314 00f7          JILLOP        equ   $00F7
0315 00fa          JCOP          equ   $00FA
0316 007e          JMPEXT        equ   $7E          Mnemonic for jump ext
0317 004a          BRKCODE        equ   $4A          Break point code
0318 004a          BRKACK         equ   $4A          Break point ack

```

```

0318          *
0319          * REGISTERS
0320 002f      SCDR          equ    $2F
0321          *
0322 0020      RDRF          equ    $20
0323 0080      TDRE          equ    $80
0324 0008      OR            equ    $08
0325 0002      FE            equ    $02
0326          *
0327          * PROGRAM
0328 be40          org      TALKBASE
0329          *
0330 be40      TLKRSTART EQU *      Initialise SCI int
0331 be40 86 7e          LDA    #JMPEXT
0332 be42 97 c4          STAA   JSCI
0333 be44 ce be 5e      LD    #SCISRV
0334 be47 df c5          STX    JSCI+1
0335 be49      USERSTART EQU*
0336 be49 8e 03 ff      LDS    #STACK
0337 be4c ce 10 00      LDX    #REGBASE
0338 be4f 6f 2c          CLR    SCCR1,X
0339 be51 cc 30 2c      LDD    #$302C          Init SCI to 9600 baud,
0340 be54 a7 2b          STAA   BAUD,X          no parity, no int
0341 be56 e7 2d          STAB   SCCR2,X          & enable SCI tx & rx.
0342 be58 86 40          LDA    #$40          Enable STOP, and I bit
0343 be5a 06            TAP           int, disable XIRQ.
0344          *          Now hang around for
0345 be5b 7e be 5b      IDLE    JMP    IDLE          SCI int from host.
0346          * A RESET from host changes above jump destination to
0347          * start of user code.
0348 be5e          SCISRV    EQU    *          On detecting int
0349 be5e b6 10 2e      LDA    SCSR+REGBASE      assume rx caused it.
0350 be61 84 20          ANDA   #RDRF
0351 be63 27 f9          BEQ    SCISRV          otherwise loop here
0352          *
0353 be65          RXSRV    EQU    *          Process received data.
0354 be65 b6 10 2f      LDA    SCDR+REGBASE      Read cmd byte, & tx it
0355 be68 43            COMA          inverted
0356 be69 8d 40          BSR    OUTSCI          as ack to host.
0357 be6b 2a 4b          BPL    INH1          Bit 7 set? => inherent
0358 be6d 8d 2d          BSR    INSCI          else B = byte count
0359 be6f 8f            XGDX          Save cmd and byte cnt.
0360 be70 8d 2a          BSR    INSCI          Read high address byte
0361 be72 17            TBA          into ACCA
0362 be73 8d 27          BSR    INSCI          then B = low addr byte
0363 be75 8f            XGDX          A=cmd; B=cnt; X=addr
0364 be76 81 fe          CMPA   #$FE
0365 be78 26 0d          BNE    RXSRV1          If cmd is mem read ...
0366          *
0367 be7a          TREADMEM EQU    *          REPEAT
0368 be7a a6 00          LDA    ,X          read required address
0369 be7c 8d 2d          BSR    OUTSCI          send it to host
0370 be7e 17            TBA          (save byte count)
0371 be7f 8d 1b          BSR    INSCI          and wait for acknowledge
0372 be81 16            TAB          (restore byte count)
0373 be82 08            INX          Increment address
0374 be83 5a            DECB          Decrement byte count
0375 be84 26 f4          BNE    TREADMEM          UNTIL all done
0376 be86 3b            RTI          & return
0377          *
0378 be87          RXSRV1    EQU    *
0379 be87 81 be          CMPA   #$BE
0380 be89 26 10          BNE    RXSRVEX          If cmd is mem write...
0381          *
0382 be8b 17            TBA          move byte count to A
0383 be8c          TWRTMEM    EQU    *          REPEAT
0384 be8c 8d 0e          BSR    INSCI          Read next byte into B
0385 be8e e7 00          STAB   ,X          and store at req addr
0386 be90 8d 18          BSR    EPRG
    
```

Engineering Bulletin


0387	be92	e6	00		LDAB	,X		Read stored byte and
0388	be94	f7	10	2f	STAB	SCDR+REGBASE		echo it back to host,
0389	be97	08			INX			
0390	be98	4a			DECA			Decrement byte count
0391	be99	26	f1		BNE	TWRITMEM		UNTIL all done
0392	be9b				RXSRVEX	EQU	*	and return
0393	be9b	3b			NULLSRV	RTI		
0394					*			
0395	be9c				INSCI	EQU	*	
0396	be9c	f6	10	2e	LDAB	SCSR+REGBASE		Wait for RDRF=1
0397	be9f	c5	00		BITB	\$(FE+OR)		If break detected then
0398	bea1	26	9d		BNE	TLKRSTART		restart talker.
0399	bea3	c4	20		ANDB	#RDRF		
0400	bea5	27	f5		BEQ	INSCI		
0401	bea7	f6	10	2f	LDAB	SCDR+REGBASE		read data received
0402	beaa	39			EPRG	RTS		& return B = data
0403					*			
0404	beab				OUTSCI	EQU	*	Only Y modified.
0405	beab	18	8f		XGDY			Enter with A = tx data
0406	bead	b6	10	2e	OUTSCII1	LDAA	SCSR+REGBASE	
0407	beb0	2a	fb		BPL	OUTSCII1		MS bit is TDRE flag
0408	beb2	18	8f		XGDY			
0409	beb4	b7	10	2f	STAA	SCDR+REGBASE		Important: Updates CCR
0410	beb7	39			RTS			
0411					*			
0412	beb8				INH1	EQU	*	
0413	beb8	81	7e		CMPA	#\$7E		If cmd is read regs ..
0414	beba	26	0c		BNE	INH2		
0415					*			
0416	becb	30			INH1A	TSX		Move SP to X
0417	becd	8f			XGDY			then to ACCD
0418	bebe	8d	eb		BSR	OUTSCI		send SP high byte 1st
0419	bec0	17			TBA			
0420	bec1	8d	e8		BSR	OUTSCI		then low byte
0421	bec3	30			TSX			Restore X (=SP)
0422	bec4	c6	09		LDAB	#9		Tx 9 bytes on stack
0423	bec6	20	b2		BRA	TREADMEM		i.e. CCR,ACCB,ACCA,IXH
0424					*			, IXL,IYH,IYL,PCH,PCL
0425	bec8				INH2	EQU	*	
0426	bec8	81	3e		CMPA	#\$3E		If cmd is write regs..
0427	beca	26	12		BNE	SWISRV1		
0428					*			
0429	becc	8d	ce		BSR	INSCI		Get SP High byte first
0430	bece	17			TBA			
0431	becf	8d	cb		BSR	INSCI		
0432	bed1	8f			XGDY			Move to X reg
0433	bed2	35			TXS			and copy to SP
0434	bed3	86	09		LDAA	#9		Then 9 bytes to stack
0435	bed5	20	b5		BRA	TWRITMEM		
0436					*			
0437	bed7				SWISRV	EQU	*	Breakpoints by SWI
0438	bed7	86	4a		LDAA	#BRKCODE		Force host to proc BR
0439	bed9	8d	d0		BSR	OUTSCI		by sending it BREAK
0440	bedb	0e			SWIIDLE	CLI		
0441	bedc	20	fd		BRA	SWIIDLE		then wait for response
0442					*	40		
0443	bede				SWISRV1	EQU	*	
0444	bede	81	4a		CMPA	#BRKACK		If host acknowledges
0445	bee0	26	b9		BNE	RXSRVEX		
0446	bee2	30			TSX			move SP to SWI stack &
0447	bee3	c6	09		LDAB	#9		
0448	bee5	3a			ABX			Send user code
0449	bee6	35			TXS			breakpoint return
0450	bee7	ec	07		LDD	7,X		address to host
0451	bee9	8d	c0		BSR	OUTSCI		(high byte first)
0452	beeb	17			TBA			
0453	beec	8d	bd		BSR	OUTSCI		(low byte next)
0454	beee	cc	be	db	LDD	#SWIIDLE		force idle loop on



```

0455 bef1 ed 07          STD    7,X          return from breakpoint
0456 bef3 20 c7          BRA    INH1A        but first return all
0457                    *          MCU registers to host
0458                    *          *****
0459                    *          * FILL UNUSED BYTES WITH ZERO
0460                    *          *
0461 bef5 00 00 00 00 00  BSZ    $BF00-*
           00 00 00 00 00
0462                    *
0463                    *          *****
0464
0465                    END
0466
0467

```

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, 1-303-675-2140
or 1-800-441-2447. Customer Focus Center, 1-800-521-6274

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

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

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

© Motorola, Ltd., 1995, 2000

EB422/D

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