Order this document
by EB422/D

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 preprogrammed 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_jmp      $00F2          XIRQ vector.
swi_jmp       $00F5          SWI vector.
cme_jmp       $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_jmp      $00F2          XIRQ vector.
swi_jmp       $00F5          SWI vector.
cme_jmp       $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

```
                            *Bootloader Firmware for MC68HC11ED0

                            ********************************************************
                            * November 11 1991
                            ********************************************************

                            * Equates for use with index offset = 0

0008                        PORTD    EQU         $08
0009                        DDRD     EQU         $09
0028                        SPCR     EQU         $28
002b                        BAUD     EQU         $2B
002c                        SCCR1    EQU         $2C
002d                        SCCR2    EQU         $2D
002e                        SCSR     EQU         $2E
002f                        SCDAT    EQU         $2F
003b                        PPROG    EQU         $3B
003f                        CONFIG   EQU         $3F

                            ********************************************************
                            * This bootstrap program allows the user to
                            * download a program of exactly 256 bytes.
                            * The program must start at $0100.
                            * Each byte of the program is received by the
                            * SCI, starting with the $0100 byte and working
                            * up to the $01FF.
                            *
                            * This bootloader is based on the bootloader for the
                            * MC68HC11A8. Modifications are to place the registers
                            * at $00XX and destination to $0100.
                            ********************************************************
bf80                                 ORG         $BF80
bf80                        BEGIN    EQU         *

                            * Init.stack
bf80   8e   00   ff                  LDS         #$00FF
                            * Put PORTD in wired-or mode
bf83   14   28   20                  BSET        SPCR        $20
                            * Init SCI, restart the divider chain, enable tx & rx
bf86   cc   a2   0c                  LDD         #$A20C
bf89   97   2d                       STAA        BAUD
bf8b   d7   2d                       STAB        SCCR2
                            * Send break as soon as start bit is detected
bf8d   14   2d   01                  BSET        SCCR2       $01
                            * Clear break as soon as start bit is detected
bf90   12   08   01 fc               BRSET       PORTD       $01 *
bf94   15   2d   01                  BCLR        SSCR2       $01 *      Clear break
                            * Wait for first character (users send $ff)
bf97   13   2e   20 fc               BRCLR       SCSR        $20 *    Wait for RDRF
bf9b   96   2f                       LDAA        SCDAT                   Read data
                            *** No jump to EEPROM or RAM here !!
                            * If data = $FF, then /16 is correct baud
bf9d   81   ff                       CMPA        #$FF
bf9f   27   03                       BEQ         BAUDOK
                            * else change to /104 (/13 & /8) 1200 @ 2 MHz
bfa1   14   2b   33                  BSET        BAUD        $33
                            * Then download 256 byte program
bfa4                        BAUDOK EQU *
bfa4   ce   01   00                  LDX         #$100                 Init pointer
                            * Read in program and put in RAM
bfa7                        BK2      EQU         *
```

```
bfa7   13   2e   20 fc              BRCLR       SCSR $20 *   $20
bfab   96   2f                      LDAA        SCDAT
bfad   a7   00                      STAA        $00,x
bfaf   97   2f                      STAA        SCDAT
bfb1   08                           INX
                          * Until the end is reached
bfb2   8c   02   00                 CPX         $#0200
bfb5   26   f0                      BNE         BK2
                          *****************************************************
                          * All start user’s program
                          *
bfb7                      STAR      EQU         *
bfb7   7e   01   00                 JMP         $0100

                                    END
```

Listing 8. MC68HC711EA9 Boatloader ROM Listing

```
                             ***********************************************************
                             *BOOTLOADER FIRMWARE FOR 68HC711EA9 - 23 Aug 91
                             ***********************************************************
                             * Features of this bootloader are. . .
                             *
                             * Auto baud select between 7812.5 and 1200 (8 MHz)
                             * 0 - 512 byte variable length download
                             * Jump to EEPROM at $B600 if 1st download byte = $00
                             * PROGRAM - Utility subroutine to program EPROM
                             * UPLOAD - Utility subroutine to dump memory to host
                             * Mask I.D. at $BFD4 = $7EA9
                             ***********************************************************
                             * Revision A -
                             *
                             * This bootloader based on the MC68HC711E9
                             * Bootloader Revision A with SCI register
                             * behavior from MC68HC711K4 Bootloader
                             * Revision B
                             *
                             * IMPORTANT VDDSYN must be tied low to allow the
                             * oscillator to work as a normal 68HC11.
                             *
                             * This new version allows variable length download
                             * by quitting reception of characters when an idle
                             * of at least four character times occurs
                             *
                             ***********************************************************
                             *
                             * EQUATES FOR USE WITH INDEX OFFSET = $1000
                             *
0008                         PORTD      EQU       $08
000e                         TCNT       EQU       $0E
0016                         TOC1       EQU       $16
0023                         TFLG1      EQU       $23
                             * BIT EQUATES FOR TFLG1
0080                         0CF1       EQU       $80
                             *
0028                         SCBD       EQU       $28              Baud Register
002a                         SCCR1      EQU       $2A
002b                         SCCR2      EQU       $2B
002c                         SCSR1      EQU       $2C
002d                         SCSR2      EQU       $2D
002e                         SCDRH      EQU       $2E
002f                         SCDRL      EQU       $2F
```

```
003b                   PPROG       EQU        $3B
                       * BIT EQUATES FOR PPROG
0020                   ELAT        EQU        $20
0001                   EPGM        EQU        $01
                       *

                       * MEMORY CONFIGURATION EQUATES
                       *
b600                   EEPMSTR     EQU        $B600           Start of EEPROM
b7ff                   EEPMEND     EQU        $B7FF           End of EEPROM
                       *
d000                   EPRMSTR     EQU        $D000           Start of EPROM
ffff                   EPRMEND     EQU        $FFFF           End of EPROM
                       *
0000                   RAMSTR      EQU        $0000
01ff                   RAMEND      EQU        $01FF

                       * DELAY CONSTANTS
                       *
0db0                   DELAYS      EQU        3504            Delay at slow baud
021b                   DELAYF      EQU        539             Delay at fast baud
                       *
1068                   PROGDEL     EQU        4200            2 ms prog delay
                       *                                      At 2.1 MHz

                       *******************************************
bf00                               ORG$BF00
                       *******************************************

                       * Next two instructions provide a predictable place
                       * to call PROGRAM and UPLOAD even if the routines
                       * change size in future versions.
                       *
bf00 7e bf 13          PROGRAM     JMP        PRGROUT         EPROM prog utility
bf03                   UPLOAD      EQU        *               Upload utility

                       *********************************************
                       * UPLOAD - Utility subroutine to send data from
                       * inside the MCU to the host via the SCI interface.
                       * Prior to calling UPLOAD set baud rate, turn on SCI
                       * and set Y=first address to upload.
                       * Bootloader leaves baud set, SCI enabled, and
                       * Y pointing at EPROM start ($D0000) so these default
                       * values do not have to be changed typically.
                       * Consecutive locations are sent via SCI in an
                       * infinite loop. Reset stops the upload process.
                       *******************************************
bf03 ce 10 00                      LDX        #$1000          Point to registers
bf06 18 a6 00          UPLOOP      LDAA       0,Y             Read byte
bf09 1f 2c 80 fc                   BRCLR      SCSR1,X $80 *   Wait for TDRE
bf0d a7 2f                         STAA       SCDRL,X         Send it
bf0f 18 08                         INY
bf11 20 f3                         BRA        UPLOOP          Next...

```

```
                      *****************************************************
                      * PROGRAM - Utility subroutine to program EPROM.
                      * Prior to calling PROGRAM set baud rate, turn on SCI
                      * set X=2ms prog delay constant, and set Y=first
                      * address to program. SP must point to RAM.
                      * Bootloader leaves baud set, SCI enabled, X=4200
                      * and Y pointing at EPROM start ($D000) so these
                      * default values don’t have to be changed typically.
                      * Delay constant in X should be equivalent to 2 ms
                      * at 2.1 MHz X=4200; at 1 MHz X=2000.
                      * At external voltage source is required for EPROM
                      * programming.
                      * This routine uses 2 bytes of stack space
                      * Routine does not return. Reset to exit.
                      *****************************************************
bf13                  PRGROUT     EQU        *
bf13 3c                           PSHX                            Save prog dly
bf14 cd 10 00                     LDX        #$1000               Point to regs

                      * Send $FF to indicate ready for program data

bf171f 2c 80 fc                   BRCLR      SCSR1,X $80 *        Wait for TDRE
bf1b 86 ff                        LDAA       #$FF
bf1d a7 2f                        STAA       SCDRL,X

bf1f                  WAIT1       EQU        *
bf1f 1f 2c 20 fc                  BRCLR      SCSR1,X $20 *        Wait for RDRF
bf23 e6 2f                        LDAB       SCDRL,X              Get rx byte
bf25 18 e1 00                     CMPB       $0,Y                 Already prog?
bf28 27 1d                        BEQ        DONEIT               Skip prog cyc
bf2a 86 20                        LDAA       #ELAT                EPROM in prog
bf2c a7 3b                        STAA       PPROG,X
bf2e 18 e7 00                     STAB       0,Y                  Write data
bf31 86 21                        LDAA       #ELAT+EPGM
bf33 a7 3b                        STAA       PPROG,X              Prog start
bf35 32                           PULA                            Get dly const
bf36 33                           PULB                            into D-reg
bf37 37                           PSHB                            But keep dly
bf38 36                           PSHA                            on stack
bf39 e3 0e                        ADDD       TCNT,X               Delay + TCNT
bf3b ed 16                        STD        TOC1,X               2ms delay
bf3d 86 80                        LDAA       #OC1F
bf3f a7 23                        STAA       TFLG1,X              Clear flag

bf41 1f 23 80 fc                  BRCLR      TFLG1,X OC1F *       Wait for dly
bf45 6f 3b                        CLR        PROG,X               Prog stop
                      *
bf47                  DONEIT      EQU        *
bf47 1f 2c 80 fc                  BRCLR      SCSR1,X $80 *        Wait for TDRE
bf4b 18 a6 00                     LDAA       $0,Y                 Get EPROM &
bf4e a7 2f                        STAA       SCDRL,X              Xmit for verf
bf50 18 08                        INY                             Next location
bf52 20 cb                        BRA        WAIT1                Back for next
                      * Loops indefinitely as long as more data sent.

                      ***********************************************************
                      * Main bootloader starts here
                      ***********************************************************
                      * RESET vector bootloader starts here

bf54                  BEGIN       EQU        *
bf54 8e 01 ff                     LDS        #RAMEND              Initialise SP
bf57 ce 10 00                     LDX        #$1000               Point at regs
bf5a cc 00 20                     LDD        #$0020               Init baud for
```

```
bf5d ed 28                        STD        SCBD,X          7812 at 2MHz
bf5f cc 40 0c                     LDD        #$400C          Port D WOI &
bf62 ed 2a                        STD        SCCR1,X         Rx & Tx on
bf64 cc 02 1b                     LDD        #DELAYF         Fast baud dly
bf67 ed 16                        STD        TOC1,X          is default
                                             *
                       * Send BREAK to signal ready for download
bf69 1c 2b 01                     BSET       SCCR2,X $01     Send break
bf6c 1e 08 01 fc                  BRSET      PORTD,X $01 *   RxD pin low?
bf70 1d 2b 01                     BCLR       SCCR2,X $01     Clear break

bf73 1f 2c 20 fc                  BRCLR      SCSR1,X $20 *   Wait for RDRF
bf77 a6 2f                        LDAA       SCDRL,X         Read data
                       * Data will be $00 if BREAK OR $00 received
bf79 26 03                        BNE        NOTZERO         No JMP ! = 0
bf7b 7e b6 00                     JMP        EEPMSTR         JMP to EEPROM
bf7e                   NOTZERO    EQU        *
bf7e 81 ff                        CMPA       #$FF            $FF is $FF
bf80 27 09                        BEQ        BAUDOK          if baud OK
                       * Or else change to 1200 @ 2MHz
bf82 c6 d0                        LDAB       #$D0
bf84 e7 29                        STAB       SCBD+1,X        Baud to 1200
bf86 cc 0d b0                     LDD        #DELAYS         And slower...
bf89 ed 16                        STD        TOC1,X          delay const
bf8b                   BAUDOK     EQU        *
bf8b 18 ce 00 00                  LDY        #RAMSTR         Start of RAM

bf8f                   WAI        EQU        *
bf8f ec 16                        LDD        TOC1,X          D = Dly const
bf91                   WTLOOP     EQU        *               Exit loop if RDRF set
bf91 1e 2c 20 07                  BRSET      SCSR1,X $20 NEWONE
bf95 8f                           XGDX                       X = Dly const
bf96 09                           DEX                        Dec count
bf97 8f                           XGDX                       D = Dly const
bf98 26 f7                        BNE        WTLOOP          Finished?
bf9a 20 0f                        BRA        STAR            Quit on t/o

bf9c                   NEWONE     EQU        *
bf9c a6 2f                        LDAA       SCDRL,X         Put rx data
bf9e 18 a7 00                     STAA       $00,Y           into next RAM
bfa1 a7 2f                        STAA       SCDRL,X         Tx for hshake
bfa3 18 08                        INY                        Next RAM loc
bfa5 18 8c 02 00                  CPY        #RAMEND+1       Past end?
bfa9 26 e4                        BNE        WAIT            No, get next

bfab                   STAR       EQU        *
bfab ec 10 68                     LDX        #PROGDEL        X = prog dly
bfae 18 ce d0 00                  LDY        #EPRMSTR        Y=EPROM start
bfb2 7e 00 00                     JMP        RAMSTR          To RAM start

                       ***********************************************************
                       * Block fill unused bytes with zeros

bfb5 00 00 00 00 00 00            BSZ        $BFD1-*
00 00 00 00 00 00
00 00 00 00 00 00
00 00 00 00 00 00
00 00 00 00

                       ***********************************************************
                       * Boot ROM revision level in ASCII
                       *          (ORG$BFD1)
bfd1 41                           FCC "A"
```

```
                       ************************************************************
                       * Mask set I.D. ($0000 FOR EPROM PARTS)
                       *           (ORG$BFD2)
bfd2 00 00                         FDB       $0000
                       ************************************************************
                       * ’711EA9 I.D. - Can be used to determine MCU type
                       *           (ORG$BFD4)
bfd4 7e a9                         FDB       $7EA9

                       ************************************************************
                       * VECTORS - point to RAM for pseudo-vector JUMPs

bfd6 00 c4                         FDB       $100-60          SCI
bfd8 00 c7                         FDB       $100-57          SPI
bfda 00 ca                         FDB       $100-54          PULSE ACCUM INPUT EDGE
bfdc 00 cd                         FDB       $100-51          PULSE ACCUM OVERFLOW
bfde 00 d0                         FDB       $100-48          TIMER OVERFLOW
bfe0 00 d3                         FDB       $100-45          TIMER O/P COMPARE 5
bfe2 00 d6                         FDB       $100-42          TIMER O/P COMPARE 4
bfe4 00 d9                         FDB       $100-39          TIMER O/P COMPARE 3
bfe6 00 dc                         FDB       $100-36          TIMER O/P COMPARE 2
bfe8 00 df                         FDB       $100-33          TIMER O/P COMPARE 1
bfea 00 e2                         FDB       $100-30          TIMER I/P CAPTURE 3
bfec 00 e5                         FDB       $100-27          TIMER I/P CAPTURE 2
bfee 00 e8                         FDB       $100-24          TIMER I/P CAPTURE 1
bff0 00 eb                         FDB       $100-21          REAL TIME INT
bff2 00 ee                         FDB       $100-18          IRQ
bff4 00 f1                         FDB       $100-15          XIRQ
bff6 00 f4                         FDB       $100-12          SWI
bff8 00 f7                         FDB       $100-9           ILLEGAL OP-CODE
bffa 00 fa                         FDB       $100-6           COP FAIL
bffc 00 fd                         FDB       $100-3           CLOCK MONITOR
bffe bf 54                         FDB       BEGIN            RESET
                                   END
```

Listing 9. MC68HC711PH8 Bootloader ROM Listing

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

```
                         *
                         * Altered PLL startup behaviour as follows :
                         * 1/ Changed programmed value read from PORT B to F
                         * 2/ If $FF is found then use value $CC instead. This
                         *    winds a 38.4KHz crystal to 8MHz (i.e. * 208).
                         * 3/ Default delay of 10ms has been adjusted for a
                         *    38.4KHz crystal rather than 640KHz.
                         *
                         * INITIAL subroutine removed since it was useless and
                         * PROGRAM enhanced to move the EPROM block iff the
                         * destination address clashes with the boot ROM
                         ****************************************************************

                         * Equates (registers in direct space)
                         *
0004                     PORTB         EQU     $04
0005                     PORTF         EQU     $05
0008                     PORTD         EQU     $08
0009                     DDRD          EQU     $09
                         *
000e                     TCNT          EQU     $0E
0016                     TOC1          EQU     $16
0023                     TFLG1         EQU     $23
002e                     PLLCR         EQU     $2E
002f                     SYNR          EQU     $2F
                         * Bit equates for TFLG1
0080                     OC1F          EQU     $80
                         *
002b                     EPROG         EQU     $2B
                         * Bit equates for EPROG
0020                     ELAT          EQU     $20
0001                     EPGM          EQU     $01
                         *
0035                     BPROT         EQU     $35
0037                     INIT2         EQU     $37
003b                     PPROG         EQU     $3B
003d                     INIT          EQU     $3D
003e                     TEST1         EQU     $3E
003f                     CONFIG        EQU     $3F
                         *
0070                     SCBD          EQU     $70
0072                     SCCR1         EQU     $72
0073                     SCCR2         EQU     $73
0074                     SCSR1         EQU     $74
0075                     SCSR2         EQU     $75
0076                     SCDRH         EQU     $76
0077                     SCDRL         EQU     $77

                         * Memory configuration equates
                         *
0d00                     EEPMSTR       EQU     $0D00       Start of EEPROM
0fff                     EEPMEND       EQU     $0FFF       End of EEPROM
                         *
4000                     EPRMSTR       EQU     $4000       Start of EPROM
ffff                     EPRMEND       EQU     $FFFF       End of EPROM
                         *
0080                     RAMSTR        EQU     $0080       Start of RAM
087f                     RAMEND        EQU     $087F       End of RAM

be40                     BOOTSTR       EQU     $BE40       Start of boot ROM
bfff                     BOOTEND       EQU     $BFFF       End of boot ROM
00b0                     BOOTHI        EQU     $B0         Hi BR address
0080                     BOOTLO        EQU     $80         Shifted BR address

```

```
                         * Delay constants
                         *
15ab                     DELAYS       EQU     5547        Dly at slow baud rate
0356                     DELAYF       EQU     854         Dly at fast baud rates
                         *
1068                     PROGDEL EQU  4200    2 mSec prog delay
                         *                                at 2.1MHz
                         *
                         ***************************************************************
be40                                  ORG     $BE40
                         ***************************************************************

                         ***************************************************************
                         * REVISION B - When the EPROM destination address
                         * is within the range of the boot ROM, then move the
                         * EPROM down to $0000 and alter the destination
                         * address to match.
                         *
                         * PROGRAM - Utility subroutine to program EPROM.
                         * Prior to calling PROGRAM set baud rate, turn on SCI
                         * set X=2ms prog delay constant, and set Y=first
                         * address to program. SP must point to RAM.
                         * Bootloader leaves baud set, and SCI enabled so
                         * default values do not have to be changed typically.
                         * Delay constant in X should be equivalent to 2 ms
                         * at 2.1 MHz X=4200; at 1 MHz X=2000, at 4MHz X=8000.
                         * An external voltage source is required for EPROM
                         * programming.
                         * This routine uses 4 bytes of stack space.
                         * Routine does not return. Reset to exit.
                         ***************************************************************
be40                     PROGOUT EQU *

                         * Send $FF to indicate ready for program data

be40 13 74 80 fc                      BRCLR   SCSR1 $80 * Wait for TDRE
be44 86 ff                            LDAA    #$FF
be46 97 77                            STAA    SCDRL

                         * WAIT FOR A BYTE
be48                     WAIT1        EQU     *
be48 13 74 20 fc                      BRCLR   SCSR1 $20 * Wait for RDRF
be4c d6 77                            LDAB    SCDRL       Get received byte
                         * REVISION B - Check if boot ROM is in the way
be4e 18 8c be 40                      CPY     #BOOTSTR    Is Y < BOOTSTR
be52 25 22                            BLO     NOADJ
be54 18 8c bf ff                      CPY     #BOOTEND    Is Y > BOOTEND
be58 22 1c                            BHI     NOADJ

                         * REVISION B - Move Boot ROM and adjust Y
be5a                     ADJ          EQU     *           Adjust Y & move EPROM
be5a 18 8f                            XGDY
be5c 84 0f                            ANDA    #$0F        Mask off hi nibble
be5e 8b 80                            ADDA    #BOOTLO     Replace with hi nibble
be60 18 8f                            XGDY
be62 86 0f                            LDAA    #$0F        EPROM on
be64 97 3f                            STAA    CONFIG       at shifted addr
be66 8d 14                            BSR     PROG        Program memory
be68 18 8f                            XGDY
be6a 84 0f                            ANDA    #$0F        Mask off hi nibble
be6c 8b b0                            ADDA    #BOOTHI     Replace with hi nibble
be6e 18 8f                            XGDY
be70 86 8f                            LDAA    #$8F        EPROM at standard addr
be72 97 3f                            STAA    CONFIG
```

```
be74 20 02                             BRA     CARRYON     Find next address

be76                     NOADJ         EQU     *           No need to adjust Y
be76 8d 04                             BSR     PROG

be78                     CARRYON EQU *
be78 18 08                             INY                 Point to next location
be7a 20 cc                             BRA     WAIT1       Back to top for next

                         * Loops indefinitely as long as more data sent.

                         * REVISION B - Now a subroutine
be7c                     PROG          EQU     *
be7c 18 e1 00                          CMPB    $0,Y        See if already prog
be7f 27 1d                             BEQ     DONEIT      If so, skip prog cycle
be81 86 20                             LDAA    #ELAT       Put EPROM in prog mode
be83 97 2b                             STAA    EPROG
be85 18 e7 00                          STAB    0,Y         Write data
be88 86 21                             LDAA    #ELAT+EPGM
be8a 97 2b                             STAA    EPROG       Turn on prog voltage
be8c 3c                                PSHX                Save delay on stack
be8d 32                                PULA                Put delay into D-reg
be8e 33                                PULB
be8f d3 0e                             ADDD    TCNT        Dly const+present TCNT
be91 dd 16                             STD     TOC1        Schedule OC1 =prog dly
be93 86 80                             LDAA    #OC1F
be95 97 23                             STAA    TFLG1       Clear any prev flag

be97 13 23 80 fc                       BRCLR   TFLG1 OC1F * Wait for delay to expire
be9b 7f 00 2b                          CLR     EPROG       Turn off prog voltage

be9e                     DONEIT        EQU     *
be9e 13 74 80 fc                       BRCLR   SCSR1 $80 *  Wait for TDRE
bea2 18 a6 00                          LDAA    $0,Y        Read from EPROM and...
bea5 97 77                             STAA    SCDRL       Xmit for verify
bea7 39                                RTS                 Go back for more

                         ********
                         *
                         * REVISION B - Load from PORTF, change $FF -> $CC
                         * Now check to see if the PLL is active:
                         * If VDDSYN pin is low the PLL is inactive and the
                         * MCU will continue using the EXTAL frequency
                         * This is detected since the PLLON bit is forced to
                         * zero.
                         * If VDDSYN pin is high then the MCU will switch
                         * to the new higher frequency. After a delay of
                         * 10ms (for a crystal of 38.4KHz).
                         * The value to be stored in the SYNR register is
                         * loaded from port F.
                         * IMPORTANT NOTE: IF SOME PINS ON PORTF ARE
                         *                 SHORTED TO GROUND, BE SURE
                         *                 TO REMOVE SHORTS BEFORE
                         *                 SETTING MDA BIT FOR EXPANDED
                         *                 BUSES
                         * Procedure for PLL is:
                         *  1/ If PLLON=0 then continue bootloader
                         *  2/ Load value from port F - ACCB
                         *  3/ If ACCB=$FF then ACCB:=$CC
                         *  4/ Reset BCS=0
                         *  5/ Reset PLLON=0
                         *  6/ Store ACCB in SYNR
                         *  7/ Set PLLON=1
                         *  8/ If PLLON=0 then continue bootloader
                         *  9/ If PLLON=1 then wait 10ms (@ 38.4KHz)
                         * 10/ Set BCS=1 and MCS=1
                         * 11/ Continue bootloader
                         ********
bea8                     PLLSTRT EQU *
bea8 96 2e                             LDAA    PLLCR       Get control values
beaa 2a 26                             BPL     DONE        Not active carry on
beac d6 05                             LDAB    PORTF       Get SYNR value
beae c1 ff                             CMPB    #$FF        Check for default
beb0 26 02                             BNE     NOTFF       Not $FF so carry on
beb2 c6 cc                             LDAB    #$CC        Load value for 8MHz
beb4                     NOTFF         EQU     *
beb4 84 bf                             ANDA    #$BF        Reset BCS=0
beb6 97 2e                             STAA    PLLCR
beb8 84 3f                             ANDA    #$3F        Reset PLLON=0
beba 97 2e                             STAA    PLLCR
bebc d7 2f                             STAB    SYNR        New value for SYNR
bebe 8a 80                             ORAA    #$80        Set PLLON=1
bec0 97 2e                             STAA    PLLCR
bec2 96 2e                             LDAA    PLLCR       Check if PLL is active
bec4 2a 0c                             BPL     DONE        Not active carry on
bec6 18 ce 00 0e                       LDY     #14         Dly for 10ms @ 38.4KHz
beca 18 09               DELLP         DEY
becc 26 fc                             BNE     DELLP
bece 8a 44                             ORAA    #$44
bed0 97 2e                             STAA    PLLCR       BCS/MCS for high speed
bed2 39                  DONE          RTS

                         ***************************************************************
                         * EEPROM ERASE DELAY
bed3                     ERASE         EQU     *
bed3 18 ce 0b b8                       LDY     #3000
bed7                     BK1           EQU     *
bed7 18 09                             DEY
bed9 26 fc                             BNE     BK1
bedb 7f 00 3b                          CLR     PPROG       Clear
bede 39                                RTS

                         ***************************************************************
                         * Block fill unused bytes with zero

bedf 00 00 00 00 00 00                 BSZ     $BF00-*
00 00 00 00 00 00
00 00 00 00 00 00
00 00 00 00 00 00
00 00 00 00 00 00
00 00 00

                         ***************************************************************
bf00                                   ORG     $BF00
                         ***************************************************************

                         * Next instruction provides a predictable place
                         * to call PROGRAM even if the routine changes in
                         * size in future versions. Note that the "UPLOAD"
                         * routine did not fit on this part.
                         *
bf00 7e be 40            PROGRAM JMP   PROGOUT Program utility

                         ***************************************************************
                         * Main bootloader starts here
                         ***************************************************************
                         * RESET vector points to here
```

```
0275 bf03                     BEGIN         EQU     *
0276 bf03 8e 08 7f                          LDS     #RAMEND     Initialize stack pntr
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    EEPMSTR     Save in any EEPROM loc
0300 bf26 5c                                INCB                Set EEPGM bit
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     #EEPMSTR
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     #EEPMEND+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                     LOP1          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     LOP1
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                          *
```

```
bf52                     NOSEC         EQU *
bf52 cc 03 56                          LDD     #DELAYF     Dly for fast bd rates
bf55 dd 16                             STD     TOC1        Set as default delay
                         * Send BREAK to signal ready for download
bf57 14 73 01                          BSET    SCCR2 $01   Set send break bit
bf5a 12 08 01 fc                       BRSET   PORTD $01 * RxD pin low?
bf5e 15 73 01                          BCLR    SCCR2 $01   Clear send break bit

bf61 13 74 20 fc                       BRCLR   SCSR1 $20 * Wait for RDRF
bf65 96 77                             LDAA    SCDRL       Read data
                         * Data will be $00 if BREAK or $00 received
bf67 26 03                             BNE     NOTZERO     Bypass jump if not $00
bf69 7e 0d 00                          JMP     EEPMSTR     Jump to EEPROM if $00
bf6c                     NOTZERO       EQU     *
                         * Check div by 26 (9600 baud at 2 MHz)
bf6c 81 f0                             CMPA    #$F0        $F0 seen as $F0...
bf6e 27 1d                             BEQ     BAUDOK      if baud was correct
                         * Check div by 208 (1200 baud at 2 MHz)
bf70 c6 d0                             LDAB    #$D0        Init B for this rate
bf72 81 80                             CMPA    #$80        $FF seen as $80...
bf74 27 10                             BEQ     SLOBAUD     if baud was correct
                         * Check div by 64 (3906 baud at 2 MHz)
                         * (equals: 8192 baud at 4.2 MHz)
bf76 c6 40                             LDAB    #$40        Init B for this rate
bf78 85 20                             BITA    #$20        $FD has bit 5 clear...
bf7a 27 0a                             BEQ     SLOBAUD     if baud was correct
                         * Change to div by 32 (7812 baud at 2 MHz)
                         * (equals: 8192 baud at 2.1 MHz)
bf7c c6 20                             LDAB    #$20        Init B for this rate
bf7e d7 71                             STAB    SCBD+1
bf80 85 08                             BITA    #$08        $FF has bit 3 set...
bf82 26 09                             BNE     BAUDOK      if baud was correct
                         * Change to div by 36 (7777 baud at 640KHz x 14)
                         *
bf84 c6 24                             LDAB    #$24        By default

bf86                     SLOBAUD       EQU     *
bf86 d7 71                             STAB    SCBD+1      Store baudrate
bf88 cc 15 ab                          LDD     #DELAYS     Switch to slower...
bf8b dd 16                             STD     TOC1        delay constant
bf8d                     BAUDOK        EQU     *
bf8d 18 ce 00 80                       LDY     #RAMSTR     Point to start of RAM

bf91                     WAIT          EQU     *
bf91 de 16                             LDX     TOC1        Move delay const to X
bf93                     WTLOOP        EQU     *
bf93 12 74 20 05                       BRSET   SCSR1 $20 NEWONE Exit if RDRF set
bf97 09                                DEX                 Decrement count
bf98 26 f9                             BNE     WTLOOP      Loop if not timed out
bf9a 20 0f                             BRA     STAR        Quit on timeout

bf9c                     NEWONE        EQU     *
bf9c 96 77                             LDAA    SCDRL       Get received data
bf9e 18 a7 00                          STAA    $00,Y       Store to next RAM location
bfa1 97 77                             STAA    SCDRL       Transmit for h/shake
bfa3 18 08                             INY                 Point to next RAM loc
bfa5 18 8c 08 80                       CPY     #RAMEND+1   See if past end
bfa9 26 e6                             BNE     WAIT        If not, get another

bfab                     STAR          EQU     *
bfab 7e 00 80                          JMP     RAMSTR      ** To start of RAM **
```

```
                         ***************************************************************
                         * Block fill unused bytes with zero

bfae 00 00 00 00 00 00                 BSZ     $BFBB-*
00 00 00 00 00 00
00

                         ***************************************************************
                         * Boot ROM revision level in ASCII
                         *       (ORG  $BFBB)
bfbb 42                                FCC     "B"
                         ***************************************************************
                         * Mask set I.D. ($0000 for EPROM parts)
                         *       (ORG$BFBC)
bfbc 00 00                             FDB     $0000
                         ***************************************************************
                         * 711PH8 I.D. - can be used to determine MCU type
                         * Bit 15 is a 0 if the part is ROM (or ROMless,
                         * 1 -> EPROM)
                         * Bit 14 is a 0 if unsecured
                         * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
                         * Bits 8 - Bit 4 are lower 5 bits of 2nd ASCII letter
                         * Bits 3 - Bit 0 are last digit of part number
                         * (note: $50 = P , $48 = H in ASCII)
                         *       (ORG$BFBE)
bfbe e0 88                             FDB     %1110000010001000
                         ***************************************************************
                         * VECTORS - point to RAM for pseudo-vector JUMPs

bfc0 00 00                             FDB     $0000       reserved
bfc2 00 00                             FDB     $0000       reserved
bfc4 00 00                             FDB     $0000       reserved
bfc6 00 00                             FDB     $0000       reserved
bfc8 00 00                             FDB     $0000       reserved
bfca 00 00                             FDB     $0000       reserved
bfcc 00 b5                             FDB     $100-75     BIT SYNC & LOCK
bfce 00 b8                             FDB     $100-72     PORTH WOI
bfd0 00 bb                             FDB     $100-69     8 BIT TIMERS
bfd2 00 be                             FDB     $100-66     SCI2
bfd4 00 c1                             FDB     $100-63     SPI2
bfd6 00 c4                             FDB     $100-60     SCI1
bfd8 00 c7                             FDB     $100-57     SPI1
bfda 00 ca                             FDB     $100-54     PULSE ACCUM INPUT EDGE
bfdc 00 cd                             FDB     $100-51     PULSE ACCUM OVERFLOW
bfde 00 d0                             FDB     $100-48     TIMER OVERFLOW
bfe0 00 d3                             FDB     $100-45     TIMER OUTPUT COMPARE 5
bfe2 00 d6                             FDB     $100-42     TIMER OUTPUT COMPARE 4
bfe4 00 d9                             FDB     $100-39     TIMER OUTPUT COMPARE 3
bfe6 00 dc                             FDB     $100-36     TIMER OUTPUT COMPARE 2
bfe8 00 df                             FDB     $100-33     TIMER OUTPUT COMPARE 1
bfea 00 e2                             FDB     $100-30     TIMER INPUT CAPTURE 3
bfec 00 e5                             FDB     $100-27     TIMER INPUT CAPTURE 2
bfee 00 e8                             FDB     $100-24     TIMER INPUT CAPTURE 1
bff0 00 eb                             FDB     $100-21     REAL TIME INT
bff2 00 ee                             FDB     $100-18     IRQ
bff4 00 f1                             FDB     $100-15     XIRQ
bff6 00 f4                             FDB     $100-12     SWI
bff8 00 f7                             FDB     $100-9      ILLEGAL OP-CODE
bffa 00 fa                             FDB     $100-6      COP FAIL
bffc 00 fd                             FDB     $100-3      CLOCK MONITOR
bffe bf 03                             FDB     BEGIN       RESET
                                       END
```

Listing 10. MC68HC11PH8 Bootloader ROM Listing

```
                 *****************************************************************
                 * SECURED BOOTLOADER FOR MC68HC11PH8 - 26 AUG 93
                 *****************************************************************
                 * Features of this bootloader are...
                 *
                 * Auto baud select between 7812, 1200, 9600, 5208
                 *   and 3906 (E = 2 MHz).
                 * 0 - 768 byte variable length download:
                 *   reception of characters quits when an idle of at
                 *   least four character times occurs.  (Note: at 9600
                 *   baud rate this is almost five bit times and at
                 *   5208 and 3906 rates the timeout is even longer).
                 * Jump to EEPROM at $0D00 if first download byte = $00.
                 * Part I.D. at $BFBE is $2088. {PH8}
                 *****************************************************************
                 * Revision A (12 MAY 93) -
                 *
                 * Based on 7PH8 Bootloader Rev A.
                 *****************************************************************
                 * Revision B (26 AUG 93) -
                 *
                 * Security mode caused an overwrite of registers
                 * with $FF. Fix implemented - (CPX)
                 *
                 * Altered PLL startup behaviour as follows :
                 * 1/ Changed programmed value read from PORT B to PORT F
                 * 2/ If $FF is found then use value $CC instead. This
                 *    winds a 38.4KHz crystal to 8MHz (i.e. * 208).
                 * 3/ Default delay of 10ms has been adjusted to be for a
                 *    38.4KHz crystal rather than 640KHz.
                 *
                 *****************************************************************

                 * Equates (registers in direct space)
                 *
0004             PORTB     EQU      $04
0005             PORTF     EQU      $05
0008             PORTD     EQU      $08
0009             DDRD      EQU      $09
                 *
000e             TCNT      EQU      $0E
0016             TOC1      EQU      $16
0023             TFLG1     EQU      $23
002e             PLLCR     EQU      $2E
002f             SYNR      EQU      $2F
                 * Bit equates for TFLG1
0080             OC1F      EQU      $80
                 *
                 *
0035             BPROT     EQU      $35
0037             INIT2     EQU      $37
003b             PPROG     EQU      $3B
003d             INIT      EQU      $3D
003e             TEST1     EQU      $3E
003f             CONFIG    EQU      $3F
                 *
0070             SCBD      EQU      $70
0072             SCCR1     EQU      $72
0073             SCCR2     EQU      $73
0074             SCSR1     EQU      $74
0075             SCSR2     EQU      $75
0076             SCDRH     EQU      $76
0077             SCDRL     EQU      $77

                 * Memory configuration equates
                 *
0d00             EEPMSTR   EQU      $0D00             Start of EEPROM
0fff             EEPMEND   EQU      $0FFF             End of EEPROM
                 *
0080             RAMSTR    EQU      $0080             Start of RAM
087f             RAMEND    EQU      $087F             End of RAM
```

```
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 -> $CC
0085                  * Now check to see if the PLL is active:
0086                  * If VDDSYN pin is low the PLL is inactive and the
0087                  * MCU will continue using the EXTAL frequency
0088                  * This is detected since the PLLON bit is forced to
0089                  * zero.
0090                  * If VDDSYN pin is high then the MCU will switch
0091                  * to the new higher frequency. After a delay of
0092                  * 10ms (for a crystal of 38.4KHz).
0093                  * The value to be stored in the SYNR register is
0094                  * loaded from port F.
0095                  * IMPORTANT NOTE: IF SOME PINS ON PORTF ARE
0096                  *                 SHORTED TO GROUND, BE SURE
0097                  *                 TO REMOVE SHORTS BEFORE
0098                  *                 SETTING MDA BIT FOR EXPANDED
0099                  *                 BUSSES
0100                  * Procedure for PLL is:
0101                  *  1/ If PLLON=0 then continue bootloader
0102                  *  2/ Load value from port F - ACCB
0103                  *  3/ If ACCB=$FF then ACCB:=$CC
0104                  *  4/ Reset BCS=0
0105                  *  5/ Reset PLLON=0
0106                  *  6/ Store ACCB in SYNR
0107                  *  7/ Set PLLON=1
0108                  *  8/ If PLLON=0 then continue bootloader
0109                  *  9/ If PLLON=1 then wait 10ms (@ 38.4KHz)
0110                  * 10/ Set BCS=1 and MCS=1
0111                  * 11/ Continue bootloader
0112                  ********
0113 bed5             PLLSTRT   EQU      *
0114 bed5 96 2e                 LDAA     PLLCR              Get control values
0115 bed7 2a 26                 BPL      DONE               Not active carry on
0116 bed9 d6 05                 LDAB     PORTF              Get SYNR value
0117 bedb c1 ff                 CMPB     #$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     SYNR               New value for SYNR
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      *
```

```
bf00 8e 08 7f              LDS      #RAMEND           Initialize stack pntr
bf03 8d d0                 BSR      PLLSTRT           Turn on PLL

bf05             CONTINU   EQU      *
bf05 cc 00 1a              LDD      #$001A            Initialize baud for...
bf08 dd 70                 STD      SCBD              9600 baud at 2 MHz
bf0a cc 40 0c              LDD      #$400C            Put SCI in wire-OR mode...
bf0d dd 72                 STD      SCCR1             Enable Xmtr and Rcvr
                 * Test the security bit
bf0f 12 3f 08 3d           BRSET    CONFIG $08 NOSEC
                 *****************************************************************
                 * WE ARE IN SECURITY MODE
                 * OUTPUT $FF ON TRANSMITTER
bf13             AGAIN     EQU      *
bf13 96 74                 LDAA     SCSR1
bf15 86 ff                 LDAA     #$FF
bf17 97 77                 STAA     SCDRL             Transmit $FF
                 * ACCA NOW IS SET FOR $FF

                 * ERASE EEPROM
bf19 7f 00 35              CLR      BPROT             Turn off Block Protect
bf1c c6 06                 LDAB     #$06              For EELAT and BULK ERASE
bf1e d7 3b                 STAB     PPROG             Set EELAT=1 & ERASE=1
bf20 f7 0d 00              STAB     EEPMSTR           Store in any EEPROM location
bf23 5c                    INCB                       Set EEPGM bit
bf24 d7 3b                 STAB     PPROG             to begin programming
bf26 bd bf ac              JSR      ERASE             Wait 10ms (@ 2MHz)
                 * If EEPROM is not enabled then we can't check its
                 * erased - ACCA is still $FF
bf29 13 3f 01 0d           BRCLR    CONFIG $01 NOEE
bf2d ce 0d 00              LDX      #EEPMSTR
                 * Check the EEPROM is erased
bf30             LOOP      EQU      *
bf30 a1 00                 CMPA     0,X
bf32 26 df                 BNE      AGAIN             EEPROM not erased
bf34 08                    INX
bf35 8c 10 00              CPX      #EEPMEND+1        All checked
bf38 26 f6                 BNE      LOOP
bf3a             NOEE      EQU      *
                 *****************************************************************
                 * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES WHICH
                 * ARE USED BY THE STACK AND RAMSTR WHICH IS LEFT
                 * INTACT. ACCA IS STILL $FF
bf3a             ERAM      EQU      *
bf3a ce 08 7d              LDX      #RAMEND-2
bf3d             LOP1      EQU      *
bf3d a7 00                 STAA     0,X
bf3f 09                    DEX
bf40 8c 00 7f    CPX       #RAMSTR-1**** REVISION B
bf43 26 f8                 BNE      LOP1
                 *****************************************************************
                 * NOW ERASE CONFIG REGISTER
bf45             ECONFG    EQU      *
bf45 5a                    DECB
bf46 d7 3b                 STAB     PPROG             B still = $06
bf48 d7 3f                 STAB     CONFIG            BULK ERASE CONFIG
bf4a 5c                    INCB
bf4b d7 3b                 STAB     PPROG
bf4d bd bf ac              JSR      ERASE

                 *****************************************************************
                 * NON-SECURITY AND SECURITY MODES MEET HERE
                 *
bf50             NOSEC     EQU      *
bf50 cc 03 56              LDD      #DELAYF           Delay for fast baud rates
bf53 dd 16                 STD      TOC1              Set as default delay
                 * Send BREAK to signal ready for download
bf55 14 73 01              BSET     SCCR2 $01         Set send break bit
bf58 12 08 01 fc           BRSET    PORTD $01 *       Wait for RxD pin to go low
bf5c 15 73 01              BCLR     SCCR2 $01         Clear send break bit

bf5f 13 74 20 fc           BRCLR    SCSR1 $20 *       Wait for RDRF
bf63 96 77                 LDAA     SCDRL             Read data
                 * Data will be $00 if BREAK or $00 received
```

```
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
0276 bfb2 26 fc                   BNE      BK1
0277 bfb4 7f 00 3b                CLR      PPROG             Clear
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                  *****************************************************************
```

```
                 * 11PH8 I.D. - can be used to determine MCU type
                 * Bit 15 is a 0 if the part is ROM (or ROMless,
                 * 1 -> EPROM)
                 * Bit 14 is a 0 if unsecured
                 * Bits 13 - Bit 9 are lower 5 bits of first ASCII letter
                 * Bits 8 - Bit 4 are lower 5 bits of second ASCII letter
                 * Bits 3 - Bit 0 are last digit of part number
                 * (note: $50 = P , $48 = H in ASCII)
                 *       (ORG$BFBE)
bfbe 20 88                 FDB      %0010000010001000
                 *****************************************************************
                 * VECTORS - point to RAM for pseudo-vector JUMPs

bfc0 00 00                 FDB      $0000              reserved
bfc2 00 00                 FDB      $0000              reserved
bfc4 00 00                 FDB      $0000              reserved
bfc6 00 00                 FDB      $0000              reserved
bfc8 00 00                 FDB      $0000              reserved
bfca 00 00                 FDB      $0000              reserved
bfcc 00 b5                 FDB      $100-75            BIT SYNC & LOCK
bfce 00 b8                 FDB      $100-72            PORTH WOI
bfd0 00 bb                 FDB      $100-69            8 BIT TIMERS
bfd2 00 be                 FDB      $100-66            SCI2
bfd4 00 c1                 FDB      $100-63            SPI2
bfd6 00 c4                 FDB      $100-60            SCI1
bfd8 00 c7                 FDB      $100-57            SPI1
bfda 00 ca                 FDB      $100-54            PULSE ACCUM INPUT EDGE
bfdc 00 cd                 FDB      $100-51            PULSE ACCUM OVERFLOW
bfde 00 d0                 FDB      $100-48            TIMER OVERFLOW
bfe0 00 d3                 FDB      $100-45            TIMER OUTPUT COMPARE 5
bfe2 00 d6                 FDB      $100-42            TIMER OUTPUT COMPARE 4
bfe4 00 d9                 FDB      $100-39            TIMER OUTPUT COMPARE 3
bfe6 00 dc                 FDB      $100-36            TIMER OUTPUT COMPARE 2
bfe8 00 df                 FDB      $100-33            TIMER OUTPUT COMPARE 1
bfea 00 e2                 FDB      $100-30            TIMER INPUT CAPTURE 3
bfec 00 e5                 FDB      $100-27            TIMER INPUT CAPTURE 2
bfee 00 e8                 FDB      $100-24            TIMER INPUT CAPTURE 1
bff0 00 eb                 FDB      $100-21            REAL TIME INT
bff2 00 ee                 FDB      $100-18            IRQ
bff4 00 f1                 FDB      $100-15            XIRQ
bff6 00 f4                 FDB      $100-12            SWI
bff8 00 f7                 FDB      $100-9             ILLEGAL OP-CODE
bffa 00 fa                 FDB      $100-6             COP FAIL
bffc 00 fd                 FDB      $100-3             CLOCK MONITOR
bffe bf 00                 FDB      BEGIN              RESET

                 *********************** TBRPH8.ASM 12/5/93 ***********************
                 *   Motorola Copyright 1993                                                *
                 *   MCU resident, Interrupt driven Communication routines for 68HC11       *
                 *   monitor.Provides low level memory and stack read/write operations.     *
                 *                                                                          *
                 *   This talker DOES NOT use XIRQ                                          *
                 *   ------------------------------                                         *
                 *
                 * N.B. TBRPH8 is designed to work with the 68HC11PH8 or other              *
                 *      compatible MCU types. This version of the TALKER is designed to     *
                 *      execute from MC68HC11PH8 Boot ROM.                                  *
                 *      To initiate communication with TBRPH8, the standard bootloader      *
                 *      must be used to initialise the redirected vector table and then     *
                 *      cause a jump to USER START                                          *
                 *      This talker does NOT support SWI handling (trace and break)         *
                 *
                 * CONSTANTS
be40             TALKBASE  equ $BE40Start of RAM
00c4             BOOTVECT  equ $00C4Start of bootstrap vector jump table.
087f             STACK     equ $087FAt end of this talker
                 *
00c4             JSCI      equ $00C4SCI interrupt service
00f1             JXIRQ     equ $00F1XIRQ interrupt service
00f4             JSWI      equ $00F4SWI interrupt service
00f7             JILLOP    equ $00F7Illegal opcode service
00fa             JCOP      equ $00FACOP timeout reset service
007e             JMPEXT    equ $7E  Mnemonic for jump extended instruction
004a             BRKCODE   equ $4A  Break point signal code to host.
```

```
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     #$2C
0390 be59 d7 73                 STAB     SCCR2             and enable SCI Rx interrupt & Tx.
0391 be5b 86 40                 LDAA     #$40              Enable STOP, and I bit interrupts, disable XIRQ.
0392 be5d 06                    TAP                        Set up CCR
0393                  *
0394 be5e 7e be 5e    IDLE      JMP      IDLE              Now hang around for SCI interrupt from host.
0395                  *
0396                  *A RESET from host changes above jump destination to start of user code.
0397                  *
0398 be61             SCISRV    EQU      *                 On detecting interrupt,
0399 be61 13 74 20 fc BRCLR     SCSR1    #RDRF SCISRV
0400                  *
0401 be65             RXSRV     EQU      *                 Talker code processes received data.
0402 be65 96 77                 LDAA     SCDRL             Read command byte, & echo it as acknowledge
0403 be67 43                    COMA                       inverted
0404 be68 8d 3b                 BSR      OUTSCI            to host.
0405 be6a 2a 40                 BPL      INH1              If command bit 7 set, then process inherent command
0406 be6c 8d 2c                 BSR      INSCI             else read byte count from host into ACCB.(0=256)
0407 be6e 8f                    XGDX                       Save command and byte count.
0408 be6f 8d 29                 BSR      INSCI             Read high address byte
0409 be71 17                    TBA                        into ACCA
0410 be72 8d 26                 BSR      INSCI             then low address byte into ACCB
0411 be74 8f                    XGDX                       Restore command in ACCA,count in ACCB,address in X
0412 be75 81 fe                 CMPA     #$FE
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     #$BE
0428 be88 26 0f                 BNE      RXSRVEX           If command is memory write then
0429                  *
0430 be8a 17                    TBA                        move byte count to ACCA
0431 be8b             TWRITMEM  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
0438 be96 4a                    DECA                       Decrement byte count
0439 be97 26 f2                 BNE      TWRITMEM          UNTIL all done
0440 be99             RXSRVEX   EQU      *                 and return
0441 be99 3b          NULLSRV   RTI
```

```
                 *
be9a             INSCI     EQU       *
be9a 12 74 0a a2 BRSET     SCSR1     #$0A TLKRSTART    Restart talker if break detected
be9e 13 74 20 f8           BRCLR     SCSR1 #RDRF INSCI Loop to INSCI if no character received
bea2 d6 77                 LDAB      SCDRL             then read data received from host
bea4 39          EPRG      RTS                         and return with data in ACCB
                 *
bea5             OUTSCI    EQU       *                 Only register Y modified.
bea5 13 74 80 fc OUTSCI1   BRCLR     SCSR1 #$80 OUTSCI1Loop until set before storing next byte
bea9 97 77                 STAA      SCDRL             Important - Updates CCR!
beab 39                    RTS
                 *
beac             INH1      EQU       *
beac 81 7e                 CMPA      #$7E              If command is read MCU registers then INH1A
beae 26 0c                 BNE       INH2              else jump to INH2
                 *
beb0 30          INH1A     TSX                         Move stack pointer to X
beb1 8f                    XGDX                        then to ACCD
beb2 8d f1                 BSR       OUTSCI            send stack pointer to host (high byte first)
beb4 17                    TBA
beb5 8d ee                 BSR       OUTSCI            then low byte
beb7 30                    TSX                         Restore X (=stack pointer)
beb8 c6 09                 LDAB      #9                then return 9 bytes on stack
beba 20 bd                 BRA       TREADMEM          i.e. CCR,ACCB,ACCA,IXH,IXL,IYH,IYL,PCH,PCL
                 *
bebc             INH2      EQU       *
bebc 81 3e                 CMPA      #$3E              If command is write MCU registers then don't jump
bebe 26 d9                 BNE       RXSRVEX           else quit processing
                 *
bec0 8d d8                 BSR       INSCI             get stack pointer from host (High byte first)
bec2 17                    TBA
bec3 8d d5                 BSR       INSCI
bec5 8f                    XGDX                        Move to X reg
bec6 35                    TXS                         and copy to stack pointer
bec7 86 09                 LDAA      #9                Then put next 9 bytes from host on to stack
bec9 20 c0                 BRA       TWRITMEM
                 *
                 ******************************************************************
                 * Block fill unused bytes with zero

becb 00 00 00 00 00 00     BSZ       $BED5-*
00 00 00 00

                 ******************************************************************

                           END

```

Listing 11. MC68HC711E20 Secured Bootloader ROM Listing

```
                  **********************************************************************
                  * FIRMWARE FOR SECURED 68HC711E20 - 11 December 1992
                  *
                  **********************************************************************
                  * SECURED EPROM VERSION
                  *
                  * BASED ON SECURE 711E9 BOOTLOADER 09 MAY 92
                  * EXTENDED TO CHECK BOTH ROM BLOCKS OF E20
                  *
                  * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
                  * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
                  * OF AT LEAST FOUR WORD TIMES OCCURS
                  *
                  * EQUATES FOR USE WITH INDEX OFFSET = $1000
                  *
                  *
0008              PORTD              EQU         $08
0009              DDRD               EQU         $09
```

```
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 b600              EEPSTR              EQU        $B600            START OF EEPROM
0034 b7ff              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 0db0              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
```

```
bf16 86 ff                            LDAA       #$FF
bf18 a7 2f                            STAA       SCDAT,X
                  * ACCA NOW IS SET FOR $FF
                  *
                  * ERASE EEPROM:
                  * TURN OFF BLOCK PROTECT
bf1a 6f 35                            CLR        BPROT,X
                  * SET ERASE AND EELAT BITS BEFORE USING "ERASE"
bf1c 54                               LSRB                        CHANGE $0C TO $06
bf1d e7 3b                            STAB       PPROG,X
bf1f f7 b6 00                         STAB       EEPSTR           WRITE EEPROM LOCATION
bf22 8d 34                            BSR        ERASE
                  * ACCB IS NOW SET FOR $06
                  *
                  ********
                  * ERASE CYCLE IS COMPLETE
                  *
                  * IF THE EEPROM IS NOT ENABLED,
                  * WE CAN'T CHECK THAT THE EEPROM IS ERASED
bf24 1f 3f 01 11                      BRCLR      CONFIG,X $01 NOEE
                  * EEPROM IS ON,
                  * NOW CHECK THAT THE EEPROM IS ERASED
bf28 18 ce b6 00                      LDY        #EEPSTR
bf2c              LOOP                EQU        *
bf2c 18 a1 00                         CMPA       0,Y              (A = $FF)
                  * ANY UNERASED BYTE SENDS US BACK TO ERASE AGAIN
bf2f 26 e3                            BNE        AGAIN
bf31 18 08                            INY
bf33 18 8c b8 00                      CPY        #EEPEND+1
bf37 26 f3                            BNE        LOOP
bf39              NOEE                EQU        *
                  *********************************************************************
                  * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES
                  * WHICH ARE USED BY THE STACK & $0000 WHICH IS
                  * LEFT INTACT
                  *
bf39              ERAM                EQU        *
bf39 3c                               PSHX
bf3a ce 02 fd                         LDX        #RAMEND-2
bf3d              LOP1                EQU        *
bf3d a7 00                            STAA       $00,X
bf3f 09                               DEX
bf40 26 fb                            BNE        LOP1
                  *                   PULX                        <<<<
                  * DO NOT SEPARATE RAM AND EPROM ROUTINES WITHOUT
                  * FIXING THE STACK (PULX)
                  *********************************************************************
                  * CONFIRM THAT EPROM IS ERASED
                  *
bf42 ce d0 00                         LDX        #EP1STR
bf45 8d 1f        LOP2                BSR        TSTLP
bf47 26 fc                            BNE        LOP2
bf49 ce 90 00                         LDX        #EP2STR
bf4c a1 00        LOP3                CMPA       0,X              (A=$FF)
bf4e 8d 16                            BSR        TSTLP
bf50 8c b0 00                         CPX        #EP2END+1
bf53 26 f7                            BNE        LOP3
bf55 38                               PULX                        <<<<
bf56 20 14                            BRA        ECONFG

                  *********************************************************************
                  * BOOTLOADER SUBROUTINES
                  *
                  * EEPROM ERASE SUBROUTINE
                  *
                  * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
                  * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
                  * IS TO BE ERASED
                  *
                  * ENTRY X=$1000
                  * EXIT X=$1000, Y=$0000
                  *
bf58              ERASE               EQU        *
                  * SET EEPGM BIT
bf58 1c 3b 01                         BSET       PPROG,X $01
                  * 10 MILLISEC DELAY @ 2.1 MHZ
bf5b 18 ce 0b b8                      LDY        #3000
bf5f 18 09        BK1                 DEY
bf61 26 fc                            BNE        BK1
                  * TURN OFF ERASE AND EELAT BITS
bf63 6f 3b                            CLR        PPROG,X
bf65 39                               RTS
                  *
                  * EPROM TEST SUBROUTINE
                  *
bf66 a1 00        TSTLP               CMPA       0,X              (A=$FF)
bf68 26 fe                            BNE        *
bf6a 08           TSTOK               INX
bf6b 39                               RTS

                  *********************************************************************
                  * NOW ERASE CONFIG REGISTER
                  *
bf6c              ECONFG              EQU        *
                  * SET ERASE AND EELAT BITS
bf6c e7 3b                            STAB       PPROG,X          (B STILL = $06)
                  * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
bf6e e7 3f                            STAB       CONFIG,X
bf70 8d e6                            BSR        ERASE
                  ********
                  * ERASE CYCLE IS COMPLETE
                  *
                  *********************************************************************
                  * NON-SECURITY AND SECURITY MODES MEET HERE
                  *
bf72              NOSEC               EQU        *
                  ********
                  * SET UP DELAY FOR FASTEST BAUD RATE
bf72 cc 02 1b                         LDD        #DELAYF
bf75 ed 16                            STD        TOC1,X
                  ********
                  * SEND BREAK TO SIGNAL START OF DOWNLOAD
bf77 1c 2d 01                         BSET       SCCR2,X $01
                  * CLEAR BREAK AS SOON AS START BIT IS DETECTED
bf7a 1e 08 01 fc                      BRSET      PORTD,X $01 *
bf7e 1d 2d 01                         BCLR       SCCR2,X $01      CLEAR BREAK
                  * WAIT FOR FIRST CHARACTER (USERS SEND $FF)
bf81 1f 2e 20 fc                      BRCLR      SCSR,X $20 *     WAIT FOR RDRF
bf85 a6 2f                            LDAA       SCDAT,X          READ DATA
                  * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
bf87 26 03                            BNE        NOTZERO
bf89 7e b6 00                         JMP        EEPSTR
bf8c              NOTZERO             EQU        *
                  * IF DATA = $FF, THEN /16 IS CORRECT BAUD
bf8c 81 ff                            CMPA       #$FF
bf8e 27 08                            BEQ        BAUDOK
                  * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
bf90 1c 2b 33                         BSET       BAUD,X $33
                  * SET UP DELAY FOR SLOWER BAUD RATE
bf93 cc 0d b0                         LDD        #DELAYS
bf96 ed 16                            STD        TOC1,X
                  *
bf98              BAUDOK              EQU        *
bf98 18 ce 00 00                      LDY        #RAMSTR          PNTR TO START OF RAM
                  *
                  * TIME EACH BYTE
```

Listing 11. MC68HC711E20 Secured Bootloader ROM Listing

```
bf9c ec 16        WAIT                LDD        TOC1,X           PUT DELAY TIME IN ACCD
                  *
bf9e 1e 2e 20 07  WTLOOP              BRSET      SCSR,X $20 NEWONE
bfa2 8f                               XGDX                        DELAY INTO X
bfa3 09                               DEX                         DECREMENT DELAY
bfa4 8f                               XGDX                        RETURN DELAY TO ACCD
bfa5 26 f7                            BNE        WTLOOP
bfa7 20 0f                            BRA        STAR
                  * DID NOT TIME OUT
bfa9              NEWONE              EQU        *
                  * READ IN BYTE AND PUT INTO RAM
bfa9 a6 2f                            LDAA       SCDAT,X
bfab 18 a7 00                         STAA       $00,Y
bfae a7 2f                            STAA       SCDAT,X          HANDSHAKE
bfb0 18 08                            INY
bfb2 18 8c 03 00                      CPY        #RAMEND+1
bfb6 26 e4                            BNE        WAIT
                  *********************************************************************
                  * START USER'S PROGRAM
                  *
bfb8              STAR                EQU        *
bfb8 7e 00 00                         JMP        RAMSTR
                  *********************************************************************
                  * FILL UNUSED BYTES WITH ZERO
                  *
bfbb                                  BSZ        $BFBB-*
                  *
                  *********************************************************************
                  * REVISION LEVEL IN ASCII
                  *       (ORG        $BFB)
bfbb 41                               FCC        "A"
                  *********************************************************************
                  * MASK I.D. ($0000 FOR EPROM PARTS)
                  *       (ORG        $BFB)
bfbc 00 00                            FDB        $0000
                  *********************************************************************
                  * 711E20 I.D. - can be used to determine MCU type
                  * Bit 15 is a 0 if the part is ROM (or ROMless,
                  * 1 -> EPROM)
                  * Bit 14 is a 0 if unsecured
                  * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
                  * Bits 8 - Bit 5 are $F => last five bits are number
                  * Bits 4 - Bit 0 are last digit of part number
                  * (note: $45 = E in ASCII)
                  *                  (ORG$BFB)
bfbe cb f4                            FDB        %1100101111110100
                  *********************************************************************
                  * VECTORS           ($BDC0)
                  *
bfc0 00 00                            FDB        $0000            reserved
bfc2 00 00                            FDB        $0000            reserved
bfc4 00 00                            FDB        $0000            reserved
bfc6 00 00                            FDB        $0000            reserved
bfc8 00 00                            FDB        $0000            reserved
bfca 00 00                            FDB        $0000            reserved
bfcc 00 00                            FDB        $0000            reserved
bfce 00 00                            FDB        $0000            reserved
bfd0 00 00                            FDB        $0000            reserved
bfd2 00 00                            FDB        $0000            reserved
bfd4 00 00                            FDB        $0000            reserved
bfd6 00 c4                            FDB        $100-60          SCI
bfd8 00 c7                            FDB        $100-57          SPI
bfda 00 ca                            FDB        $100-54          PULSE ACCUM INPUT EDGE
bfdc 00 cd                            FDB        $100-51          PULSE ACCUM OVERFLOW
bfde 00 d0                            FDB        $100-48          TIMER OVERFLOW
bfe0 00 d3                            FDB        $100-45          TIMER OUTPUT COMPARE 5
bfe2 00 d6                            FDB        $100-42          TIMER OUTPUT COMPARE 4
bfe4 00 d9                            FDB        $100-39          TIMER OUTPUT COMPARE 3
```

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


                  ************************ 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
be40              TALKBASE            equ        $BE40
00c4              BOOTVECT            equ        $00C4            Start of boot vectors
02ff              STACK               equ        $02FF            User may alter this
1000              REGBASE             equ        $1000
                  *
00c4              JSCI                equ        $00C4
00f1              JXIRQ               equ        $00F1
00f4              JSWI                equ        $00F4
00f7              JILLOP              equ        $00F7
00fa              JCOP                equ        $00FA
007e              JMPEXT              equ        $7E              Mnemonic for jump ext
004a              BRKCODE             equ        $4A              Break point code
004a              BRKACK              equ        $4A              Break point ack
                  *
                  * REGISTERS
002f              SCDR                equ        $2F
                  *
0020              RDRF                equ        $20
0080              TDRE                equ        $80
0008              OR                  equ        $08
0002              FE                  equ        $02
                  *
                  * PROGRAM
be40                                  org TALKBASE
                  *
be40              TLKRSTART           EQU        *                Initialise SCI int
be40 86 7e                            LDAA       #JMPEXT
be42 97 c4                            STAA       JSCI
be44 ce be 5e                         LDX        #SCISRV
be47 df c5                            STX        JSCI+1
be49              USERSTART           EQU        *
be49 8e 02 ff                         LDS        #STACK
be4c ce 10 00                         LDX        #REGBASE
be4f 6f 2c                            CLR        SCCR1,X
be51 cc 30 2c                         LDD        #$302C           Init SCI to 9600 baud,
be54 a7 2b                            STAA       BAUD,X            no parity, no int
be56 e7 2d                            STAB       SCCR2,X           & enable SCI tx & rx.
```

```
be58 86 40                            LDAA       #$40             Enable STOP, and I bit
be5a 06                               TAP                         int, disable XIRQ.
                  *                                               Now hang around for
be5b 7e be 5b     IDLE                JMP        IDLE              SCI int from host.
                  * A RESET from host changes above jump destination to
                  *  start of user code.
be5e              SCISRV              EQU        *                On detecting int,
be5e b6 10 2e                         LDAA       SCSR+REGBASE     assume rx caused it.
be61 84 20                            ANDA       #RDRF
be63 27 f9                            BEQ        SCISRV           otherwise loop here
                  *
be65              RXSRV               EQU        *                Process received data.
be65 b6 10 2f                         LDAA       SCDR+REGBASE     Read cmd byte, & tx it
be68 43                               COMA                         inverted
be69 8d 46                            BSR        OUTSCI            as ack to host.
be6b 2a 51                            BPL        INH1             Bit 7 set? => inherent
be6d 8d 33                            BSR        INSCI            else B = byte count
be6f 8f                               XGDX                        Save cmd and byte cnt.
be70 8d 30                            BSR        INSCI            Read high address byte
be72 17                               TBA                         into ACCA
be73 8d 2d                            BSR        INSCI            then B = low addr byte
be75 8f                               XGDX                        A=cmd; B=cnt; X=addr
be76 81 fe                            CMPA       #$FE
be78 26 0d                            BNE        RXSRV1           If cmd is mem read ...
                  *
be7a              TREADMEM            EQU        *                REPEAT
be7a a6 00                            LDAA ,X     read required address
be7c 8d 33                            BSR OUTSCI  send it to host
be7e 17                               TBA                          (save byte count)
be7f 8d 21                            BSR INSCI   and wait for ack
be81 16                               TAB                          (restore byte count)
be82 08                               INX                          Increment address
be83 5a                               DECB                         Decrement byte count
be84 26 f4                            BNE TREADMEM UNTIL all done
be86 3b                               RTI                         & return
                  *
be87              RXSRV1              EQU *
be87 81 be                            CMPA #$BE
be89 26 16                            BNE RXSRVEX If cmd is mem write ..
                  *
be8b 17                               TBA                         move byte count to A
be8c              TWRITMEM            EQU        *                REPEAT
be8c 8d 14                            BSR        INSCI             Read next byte into B
be8e e7 00                            STAB       ,X                and store at req addr
be90 18 ce 00 01                      LDY        #$0001            Set up wait loop
be94 18 09        WAITPOLL            DEY
be96 26 fc                            BNE        WAITPOLL
be98 e6 00                            LDAB       ,X                Read stored byte and
be9a f7 10 2f                         STAB       SCDR+REGBASE      echo it back to host,
be9d 08                               INX
be9e 4a                               DECA                         Decrement byte count
be9f 26 eb                            BNE        TWRITMEM         UNTIL all done
bea1              RXSRVEX             EQU        *                and return
bea1 3b           NULLSRV             RTI
                  *
bea2              INSCI               EQU *
bea2 f6 10 2e                         LDAB       SCSR+REGBASE     Wait for RDRF=1
bea5 c5 00                            BITB       #(FE+OR)         If break detected then
bea7 26 97                            BNE        TLKRSTART        restart talker.
bea9 c4 20                            ANDB       #RDRF
beab 27 f5                            BEQ        INSCI
bead f6 10 2f                         LDAB       SCDR+REGBASE     read data received
beb0 39                               RTS                         & return B = data
                  *
beb1              OUTSCI              EQU        *                Only Y modified.
beb1 18 8f                            XGDY                        Enter with A = tx data
beb3 b6 10 2e     OUTSCI1             LDAA       SCSR+REGBASE
beb6 2a fb                            BPL        OUTSCI1          MS bit is TDRE flag
```

```
beb8 18 8f                            XGDY
beba b7 10 2f                         STAA        SCDR+REGBASE     Important: Updates CCR
bebd 39                               RTS
                  *
bebe              INH1                EQU         *
bebe 81 7e                            CMPA        #$7E             If cmd is read regs ..
bec0 26 0c                            BNE         INH2
                  *
bec2 30           INH1A               TSX                          Move SP to X
bec3 8f                               XGDX                         then to ACCD
bec4 8d eb                            BSR         OUTSCI           send SP high byte 1st
bec6 17                               TBA
bec7 8d e8                            BSR         OUTSCI           then low byte
bec9 30                               TSX                          Restore X (=SP)
beca c6 09                            LDAB        #9               Tx 9 bytes on stack
becc 20 ac                            BRA         TREADMEM         i.e. CCR,ACCB,ACCA,IXH
                  *                                                IXL,IYH,IYL,PCH,PLC
bece              INH2                EQU         *
bece 81 3e                            CMPA        #$3E             If cmd is write regs..
bed0 26 12                            BNE         SWISRV1
                  *
bed2 8d ce                            BSR         INSCI            get SP High byte first
bed4 17                               TBA
bed5 8d cb                            BSR         INSCI
bed7 8f                               XGDX                         Move to X reg
bed8 35                               TXS                          and copy to SP
bed9 86 09                            LDAA        #9               Then 9 bytes to stack
bedb 20 af                            BRA         TWRITMEM
                  *
bedd              SWISRV              EQU         *                Breakpoints by SWI
bedd 86 4a                            LDAA        #BRKCODE         Force host to proc BR
bedf 8d d0                            BSR OUTSCI  by sending it BREAK
bee1 0e           SWIIDLE             CLI
bee2 20 fd                            BRA SWIIDLE then wait for response
                  *
bee4              SWISRV1             EQU *
bee4 81 4a                            CMPA #BRKACK If host acknowledges
bee6 26 b9                            BNE RXSRVEX
bee8 30                               TSX                          move SP to SWI stack &
bee9 c6 09                            LDAB #9
beeb 3a                               ABX                          Send user code
beec 35                               TXS                           breakpoint return
beed ec 07                            LDD 7,X      address to host
beef 8d c0                            BSR OUTSCI  (high byte first)
bef1 17                               TBA
bef2 8d bd                            BSR OUTSCI  (low byte next)
bef4 cc be e1                         LDD #SWIIDLE force idle loop on
bef7 ed 07                            STD 7,X     return from breakpoint
bef9 20 c7                            BRA INH1A   but first return all
                  *                                                MCU registers to host
                  ***********************************************************************
                  * FILL UNUSED BYTES WITH ZERO
                  *
befb 00 00 00 00 00                   BSZ         $BF00-*
                  *
                  ***********************************************************************

                                      END


```

Listing 12. MC68HC711E32 Secured Bootloader ROM Listing

```
                        *******************************************************************
                        * FIRMWARE FOR SECURED 68HC711E32 - 16 July 1993
                        *
                        *******************************************************************
                        * SECURED EPROM VERSION
                        *
                        * Part I.D. at $BFBE is $CBE0. {7E32}
                        *******************************************************************
                        * REVISION A
                        * BASED ON SECURE 711E20 BOOTLOADER 11 DECEMBER 92
                        * EXTENDED TO CHECK BOTH LARGER ROM BLOCKS OF E32
                        *
                        *******************************************************************
                        * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
                        * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
                        * OF AT LEAST FOUR WORD TIMES OCCURS
                        *
                        * EQUATES FOR USE WITH INDEX OFFSET = $1000
                        *
                        *
0008                    PORTD          EQU     $08
0009                    DDRD           EQU     $09
0016                    TOC1           EQU     $16                   [STORAGE (POOR STYLE)]
0028                    SPCR           EQU     $28                   (FOR DWOM BIT)
002b                    BAUD           EQU     $2B
002c                    SCCR1          EQU     $2C
002d                    SCCR2          EQU     $2D
002e                    SCSR           EQU     $2E
002f                    SCDAT          EQU     $2F
0035                    BPROT          EQU     $35
003b                    PPROG          EQU     $3B
003e                    TEST1          EQU     $3E
003f                    CONFIG         EQU     $3F
                        *
                        * MORE EQUATES
                        *
b600                    EEPSTR         EQU     $B600                 START OF EEPROM
b7ff                    EEPEND         EQU     $B7FF                 END OF EEPROM
                        *
c000                    EP1STR         EQU     $C000                 START OF EPROM 1
ffff                    EP1END         EQU     $FFFF                 END OF EPROM 1
7000                    EP2STR         EQU     $7000                 START OF EPROM 2
afff                    EP2END         EQU     $AFFF                 END OF EPROM 2
                        *
0000                    RAMSTR         EQU     $0000
07ff                    RAMEND         EQU     $07FF
                        *
0db0                    DELAYS         EQU     3504                  DELAY AT SLOW BAUD
021b                    DELAYF         EQU     539                   DELAY AT FAST BAUD
                        *
                        ********
                        * THIS BOOTSTRAP PROGRAM ALLOWS THE USER TO
                        * DOWNLOAD A PROGRAM OF 0 - 2048 BYTES.
                        * THE PROGRAM MUST START AT $0000.
                        * EACH BYTE OF THE PROGRAM IS RECEIVED BY THE SCI.
                        * THE FIRST BYTE ESTABLISHES BAUD RATE.
                        * THEN THE PROGRAM IS DOWNLOADED STARTING WITH
                        * THE $0000 BYTE AND WORKING UP TOWARD THE $03FF
                        * A DELAY OF FOUR WORD TIMES (AT EITHER BAUD RATE)
                        * CAUSES THE RECEPTION OF CHARACTERS TO STOP AND
                        * A JUMP TO $0000.
                        *
                        * THE TRANSMITTER WILL BE USED FOR THE PURPOSE
                        * OF COMMUNICATION TO THE OUTSIDE WORLD.
                        *
                        *******************************************************************
```

```
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    EEPSTR                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                    LOP1           EQU     *
0131 bf3d a7 00                             STAA    $00,X
0132 bf3f 09                                DEX
0133 bf40 26 fb                             BNE     LOP1
0134                         *                PULX                        <<<<
```

```
                        * DO NOT SEPARATE RAM AND EPROM ROUTINES WITHOUT
                        * FIXING THE STACK (PULX)
                        ******************************************************************
                        * CONFIRM THAT EPROM IS ERASED
                        *
bf42 ce c0 00                          LDX     #EP1STR
bf45 8d 1f              LOP2           BSR     TSTLP
bf47 26 fc                             BNE     LOP2
bf49 ce 70 00                          LDX     #EP2STR
bf4c a1 00              LOP3           CMPA    0,X                   (A=$FF)
bf4e 8d 16                             BSR     TSTLP
bf50 8c b0 00                          CPX     #EP2END+1
bf53 26 f7                             BNE     LOP3
bf55 38                                PULX                          <<<<
bf56 20 14                             BRA     ECONFG

                        ******************************************************************
                        * BOOTLOADER SUBROUTINES
                        *
                        * EEPROM ERASE SUBROUTINE
                        *
                        * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
                        * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
                        * IS TO BE ERASED
                        *
                        * ENTRY X=$1000
                        * EXIT X=$1000, Y=$0000
                        *
bf58                    ERASE          EQU     *
                        * SET EEPGM BIT
bf58 1c 3b 01                          BSET    PPROG,X $01
                        * 10 MILLISEC DELAY @ 2.1 MHZ
bf5b 18 ce 0b b8                       LDY     #3000
bf5f 18 09              BK1            DEY
bf61 26 fc                             BNE     BK1
                        * TURN OFF ERASE AND EELAT BITS
bf63 6f 3b                             CLR     PPROG,X
bf65 39                                RTS
                        *
                        * EPROM TEST SUBROUTINE
                        *
bf66 a1 00              TSTLP          CMPA    0,X                   (A=$FF)
bf68 26 fe                             BNE     *                     (Loop here forever)
bf6a 08                 TSTOK          INX
bf6b 39                                RTS

                        ******************************************************************
                        * NOW ERASE CONFIG REGISTER
                        *
bf6c                    ECONFG         EQU     *
                        * SET ERASE AND EELAT BITS
bf6c e7 3b                             STAB    PPROG,X               (B STILL = $06)
                        * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
bf6e e7 3f                             STAB    CONFIG,X
bf70 8d e6                             BSR     ERASE
                        ********
                        * ERASE CYCLE IS COMPLETE
                        *
                        ******************************************************************
                        * NON-SECURITY AND SECURITY MODES MEET HERE
                        *
bf72                    NOSEC          EQU     *
                        ********
                        * SET UP DELAY FOR FASTEST BAUD RATE
bf72 cc 02 1b                          LDD     #DELAYF
bf75 ed 16                             STD     TOC1,X
                        ********
                        * SEND BREAK TO SIGNAL START OF DOWNLOAD
bf77 1c 2d 01                          BSET    SCCR2,X $01
```

```
                        * CLEAR BREAK AS SOON AS START BIT IS DETECTED
bf7a 1e 08 01 fc                       BRSET   PORTD,X $01 *
bf7e 1d 2d 01                          BCLR    SCCR2,X $01           CLEAR BREAK
                        * WAIT FOR FIRST CHARACTER (USERS SEND $FF)
bf81 1f 2e 20 fc                       BRCLR   SCSR,X $20 *           WAIT FOR RDRF
bf85 a6 2f                             LDAA    SCDAT,X               READ DATA
                        * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
bf87 26 03                             BNE     NOTZERO
bf89 7e b6 00                          JMP     EEPSTR
bf8c                    NOTZERO        EQU     *
                        * IF DATA = $FF, THEN /16 IS CORRECT BAUD
bf8c 81 ff                             CMPA    #$FF
bf8e 27 08                             BEQ     BAUDOK
                        * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
bf90 1c 2b 33                          BSET    BAUD,X $33
                        * SET UP DELAY FOR SLOWER BAUD RATE
bf93 cc 0d b0                          LDD     #DELAYS
bf96 ed 16                             STD     TOC1,X
                        *
bf98                    BAUDOK         EQU     *
bf98 18 ce 00 00                       LDY     #RAMSTR               PNTR TO START OF RAM
                        *
                        * TIME EACH BYTE
bf9c ec 16              WAIT           LDD     TOC1,X                PUT DELAY TIME IN ACCD
                        *
bf9e 1e 2e 20 07        WTLOOP         BRSET   SCSR,X $20 NEWONE
bfa2 8f                                XGDX                          DELAY INTO X
bfa3 09                                DEX                           DECREMENT DELAY
bfa4 8f                                XGDX                          RETURN DELAY TO ACCD
bfa5 26 f7                             BNE     WTLOOP
bfa7 20 0f                             BRA     STAR
                        * DID NOT TIME OUT
bfa9                    NEWONE         EQU     *
                        * READ IN BYTE AND PUT INTO RAM
bfa9 a6 2f                             LDAA    SCDAT,X
bfab 18 a7 00                          STAA    $00,Y
bfae a7 2f                             STAA    SCDAT,X               HANDSHAKE
bfb0 18 08                             INY
bfb2 18 8c 08 00                       CPY     #RAMEND+1
bfb6 26 e4                             BNE     WAIT
                        *******************************************************************
                        * START USER'S PROGRAM
                        *
bfb8                    STAR           EQU     *
bfb8 7e 00 00                          JMP     RAMSTR
                        *******************************************************************
                        * FILL UNUSED BYTES WITH ZERO
                        *
bfbb                                   BSZ     $BFBB-*
                        *
                        *******************************************************************
                        * REVISION LEVEL IN ASCII
                        *       (ORG   $BFBB)
bfbb 41                                FCC     "A"
                        *******************************************************************
                        * MASK I.D. ($0000 FOR EPROM PARTS)
                        *       (ORG   $BFBC)
bfbc 00 00                             FDB     $0000
                        ************************************************
                        * 711E32 I.D. - can be used to determine MCU type
                        * Bit 15 is a 0 if the part is ROM (or ROMless,
                        * 1 -> EPROM)
                        * Bit 14 is a 0 if unsecured
                        * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
                        * Bits 8 - Bit 6 are $7 => last six bits are number
                        * Bits 5 - Bit 0 are last digit of part number
                        * (note: $45 = E in ASCII)
                        *
                        *     15 14 13   9 8 6 5    0
                        * i.e. 1  1  00101 111 100000 (CBE0)
                        *     EP SEC  'E'  nul   32
                        *
                        *              (ORG$BFBE)
bfbe cb e0                             FDB      %1100101111100000
                        *******************************************************************
                        * VECTORS      ($BDC0)
                        *
bfc0 00 00                             FDB     $0000                reserved
bfc2 00 00                             FDB     $0000                reserved
bfc4 00 00                             FDB     $0000                reserved
bfc6 00 00                             FDB     $0000                reserved
bfc8 00 00                             FDB     $0000                reserved
bfca 00 00                             FDB     $0000                reserved
bfcc 00 00                             FDB     $0000                reserved
bfce 00 00                             FDB     $0000                reserved
bfd0 00 00                             FDB     $0000                reserved
bfd2 00 00                             FDB     $0000                reserved
bfd4 00 00                             FDB     $0000                reserved
bfd6 00 c4                             FDB     $100-60              SCI
bfd8 00 c7                             FDB     $100-57              SPI
bfda 00 ca                             FDB     $100-54              PULSE ACCUM INPUT EDGE
bfdc 00 cd                             FDB     $100-51              PULSE ACCUM OVERFLOW
bfde 00 d0                             FDB     $100-48              TIMER OVERFLOW
bfe0 00 d3                             FDB     $100-45              TIMER OUTPUT COMPARE 5
bfe2 00 d6                             FDB     $100-42              TIMER OUTPUT COMPARE 4
bfe4 00 d9                             FDB     $100-39              TIMER OUTPUT COMPARE 3
bfe6 00 dc                             FDB     $100-36              TIMER OUTPUT COMPARE 2
bfe8 00 df                             FDB     $100-33              TIMER OUTPUT COMPARE 1
bfea 00 e2                             FDB     $100-30              TIMER INPUT CAPTURE 3
bfec 00 e5                             FDB     $100-27              TIMER INPUT CAPTURE 2
bfee 00 e8                             FDB     $100-24              TIMER INPUT CAPTURE 1
bff0 00 eb                             FDB     $100-21              REAL TIME INT
bff2 00 ee                             FDB     $100-18              IRQ
bff4 00 f1                             FDB     $100-15              XIRQ
bff6 00 f4                             FDB     $100-12              SWI
bff8 00 f7                             FDB     $100-9               ILLEGAL OP-CODE
bffa 00 fa                             FDB     $100-6               COP FAIL
bffc 00 fd                             FDB     $100-3               CLOCK MONITOR
bffe bf 00                             FDB     BEGIN                RESET


                        *********************** TBRE32.ASC 30/6/93 ***********************
                        * Motorola Copyright 1988,1990,1992,1993                              *
                        * MCU resident, Interrupt driven Communication routines for 68HC11    *
                        * monitor.Provides low level memory & stack read/write operations     *
                        *                                                                     *
                        *   This talker DOES NOT use XIRQ                                     *
                        *   -----------------------------                                     *
                        *                                                                     *
                        * N.B. TBRE32 is designed to work with the 68HC11E32 or other         *
                        *  compatible MCU types. This version of the TALKER is designed to    *
                        *  execute from MC68HC711E32 Boot ROM.                                *
                        *  To initiate communication with TBRE32, the standard bootloader     *
                        *  must be used to initialise the redirected vector table and the     *
                        *  cause a jump to USERSTART                                          *
                        *
                        * CONSTANTS
be40                    TALKBASE       equ     $BE40
00c4                    BOOTVECT       equ     $00C4                Start of boot vectors
03ff                    STACK          equ     $03FF                User may alter this
1000                    REGBASE        equ     $1000
                        *
00c4                    JSCI           equ     $00C4
00f1                    JXIRQ          equ     $00F1
00f4                    JSWI           equ     $00F4
00f7                    JILLOP         equ     $00F7
00fa                    JCOP           equ     $00FA
007e                    JMPEXT         equ     $7E                  Mnemonic for jump ext
```

```
004a                    BRKCODE        equ     $4A                   Break point code
004a                    BRKACK         equ     $4A                   Break point ack
                        *
                        * REGISTERS
002f                    SCDR           equ     $2F
                        *
0020                    RDRF           equ     $20
0080                    TDRE           equ     $80
0008                    OR             equ     $08
0002                    FE             equ     $02
                        *
                        * PROGRAM
be40                                   org     TALKBASE
                        *
be40                    TLKRSTART      EQU     *                     Initialise SCI int
be40 86 7e                             LDAA    #JMPEXT
be42 97 c4                             STAA    JSCI
be44 ce be 5e                          LDX     #SCISRV
be47 df c5                             STX     JSCI+1
be49                    USERSTART      EQU     *
be49 8e 03 ff                          LDS     #STACK
be4c ce 10 00                          LDX     #REGBASE
be4f 6f 2c                             CLR     SCCR1,X
be51 cc 30 2c                          LDD     #$302C                Init SCI to 9600 baud,
be54 a7 2b                             STAA    BAUD,X                 no parity, no int
be56 e7 2d                             STAB    SCCR2,X               & enable SCI tx & rx.
be58 86 40                             LDAA    #$40                  Enable STOP, and I bit
be5a 06                                TAP                           int, disable XIRQ.
                        *                                            Now hang around for
be5b 7e be 5b           IDLE           JMP     IDLE                   SCI int from host.
                        * A RESET from host changes above jump destination to
                        *  start of user code.
be5e                    SCISRV         EQU     *                     On detecting int
be5e b6 10 2e                          LDAA    SCSR+REGBASE          assume rx caused it.
be61 84 20                             ANDA    #RDRF
be63 27 f9                             BEQ     SCISRV                otherwise loop here
                        *
be65                    RXSRV          EQU     *                     Process received data.
be65 b6 10 2f                          LDAA    SCDR+REGBASE          Read cmd byte, & tx it
be68 43                                COMA                           inverted
be69 8d 40                             BSR     OUTSCI                 as ack to host.
be6b 2a 4b                             BPL     INH1                  Bit 7 set? => inherent
be6d 8d 2d                             BSR     INSCI                 else B = byte count
be6f 8f                                XGDX                          Save cmd and byte cnt.
be70 8d 2a                             BSR     INSCI                 Read high address byte
be72 17                                TBA                           into ACCA
be73 8d 27                             BSR     INSCI                 then B = low addr byte
be75 8f                                XGDX                          A=cmd; B=cnt; X=addr
be76 81 fe                             CMPA    #$FE
be78 26 0d                             BNE     RXSRV1                If cmd is mem read ...
                        *
be7a                    TREADMEM       EQU     *                     REPEAT
be7a a6 00                             LDAA    ,X                     read required address
be7c 8d 2d                             BSR     OUTSCI                 send it to host
be7e 17                                TBA                            (save byte count)
be7f 8d 1b                             BSR     INSCI                  and wait for acknowledge
be81 16                                TAB                            (restore byte count)
be82 08                                INX                            Increment address
be83 5a                                DECB                           Decrement byte count
be84 26 f4                             BNE     TREADMEM              UNTIL all done
be86 3b                                RTI                           & return
                        *
be87                    RXSRV1         EQU *
be87 81 be                             CMPA    #$BE
be89 26 10                             BNE     RXSRVEX               If cmd is mem write...
                        *
be8b 17                                TBA                           move byte count to A
be8c                    TWRITMEM       EQU     *                     REPEAT
be8c 8d 0e                             BSR     INSCI                  Read next byte into B
```

```
be8e e7 00                             STAB    ,X                     and store at req addr
be90 8d 18                             BSR     EPRG
be92 e6 00                             LDAB    ,X                     Read stored byte and
be94 f7 10 2f                          STAB    SCDR+REGBASE           echo it back to host,
be97 08                                INX
be98 4a                                DECA                           Decrement byte count
be99 26 f1                             BNE     TWRITMEM              UNTIL all done
be9b                    RXSRVEX        EQU     *                     and return
be9b 3b                 NULLSRV        RTI
                        *
be9c                    INSCI          EQU     *
be9c f6 10 2e                          LDAB    SCSR+REGBASE          Wait for RDRF=1
be9f c5 00                             BITB    #(FE+OR)              If break detected then
bea1 26 9d                             BNE     TLKRSTART             restart talker.
bea3 c4 20                             ANDB    #RDRF
bea5 27 f5                             BEQ     INSCI
bea7 f6 10 2f                          LDAB    SCDR+REGBASE          read data received
beaa 39                 EPRG           RTS                           & return B = data
                        *
beab                    OUTSCI         EQU     *                     Only Y modified.
beab 18 8f                             XGDY                          Enter with A = tx data
bead b6 10 2e           OUTSCI1        LDAA    SCSR+REGBASE
beb0 2a fb                             BPL     OUTSCI1               MS bit is TDRE flag
beb2 18 8f                             XGDY
beb4 b7 10 2f                          STAA    SCDR+REGBASE          Important: Updates CCR
beb7 39                                RTS
                        *
beb8                    INH1           EQU     *
beb8 81 7e                             CMPA    #$7E                  If cmd is read regs ..
beba 26 0c                             BNE     INH2
                        *
bebc 30                 INH1A          TSX                           Move SP to X
bebd 8f                                XGDX                          then to ACCD
bebe 8d eb                             BSR     OUTSCI                send SP high byte 1st
bec0 17                                TBA
bec1 8d e8                             BSR     OUTSCI                then low byte
bec3 30                                TSX                           Restore X (=SP)
bec4 c6 09                             LDAB    #9                    Tx 9 bytes on stack
bec6 20 b2                             BRA     TREADMEM              i.e.CCR,ACCB,ACCA,IXH
                        *                                            ,IXL,IYH,IYL,PCH,PCL
bec8                    INH2           EQU     *
bec8 81 3e                             CMPA    #$3E                  If cmd is write regs..
beca 26 12                             BNE     SWISRV1
                        *
becc 8d ce                             BSR     INSCI                 Get SP High byte first
bece 17                                TBA
becf 8d cb                             BSR     INSCI
bed1 8f                                XGDX                          Move to X reg
bed2 35                                TXS                           and copy to SP
bed3 86 09                             LDAA    #9                    Then 9 bytes to stack
bed5 20 b5                             BRA     TWRITMEM
                        *
bed7                    SWISRV         EQU     *                     Breakpoints by SWI
bed7 86 4a                             LDAA    #BRKCODE              Force host to proc BR
bed9 8d d0                             BSR     OUTSCI                by sending it BREAK
bedb 0e                 SWIIDLE        CLI
bedc 20 fd                             BRA     SWIIDLE               then wait for response
                        *
bede                    SWISRV1        EQU     *
bede 81 4a                             CMPA    #BRKACK               If host acknowledges
bee0 26 b9                             BNE     RXSRVEX
bee2 30                                TSX                           move SP to SWI stack &
bee3 c6 09                             LDAB    #9
bee5 3a                                ABX                           Send user code
bee6 35                                TXS                            breakpoint return
bee7 ec 07                             LDD     7,X                    address to host
bee9 8d c0                             BSR     OUTSCI                (high byte first)
beeb 17                                TBA
beec 8d bd                             BSR     OUTSCI                (low byte next)
```

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

                        *
                        ******************************************************************

                                      END


```

Listing 13. MC68HC11E32 Secured Bootloader ROM Listing

```
                       ******************************************************************
                       * FIRMWARE FOR SECURED 68HC11E32 - 24 SEPT 1993
                       *
                       ******************************************************************
                       * SECURED EPROM VERSION
                       *
                       * Part I.D. at $BFBE is $4BE0. {E32}
                       ******************************************************************
                       * REVISION A 24 SEPT 1993
                       * BASED ON SECURE 711E32 BOOTLOADER 16 JULY 93
                       * REMOVED CHECK ON EPROM
                       *
                       ******************************************************************
                       * THIS NEW VERSION ALLOWS VARIABLE LENGTH DOWNLOAD
                       * BY QUITTING RECEPTION OF CHARACTERS WHEN AN IDLE
                       * OF AT LEAST FOUR WORD TIMES OCCURS
                       *
                       * EQUATES FOR USE WITH INDEX OFFSET = $1000
                       *
                       *
0008                   PORTD          EQU    $08
0009                   DDRD           EQU    $09
0016                   TOC1           EQU    $16                     [STORAGE (POOR STYLE)]
0028                   SPCR           EQU    $28                     (FOR DWOM BIT)
002b                   BAUD           EQU    $2B
002c                   SCCR1          EQU    $2C
002d                   SCCR2          EQU    $2D
002e                   SCSR           EQU    $2E
002f                   SCDAT          EQU    $2F
0035                   BPROT          EQU    $35
003b                   PPROG          EQU    $3B
003e                   TEST1          EQU    $3E
003f                   CONFIG         EQU    $3F
                       *
                       * MORE EQUATES
                       *
b600                   EEPSTR         EQU    $B600                   START OF EEPROM
b7ff                   EEPEND         EQU    $B7FF                   END OF EEPROM
                       *
0000                   RAMSTR         EQU    $0000
07ff                   RAMEND         EQU    $07FF
                       *
0db0                   DELAYS         EQU    3504                    DELAY AT SLOW BAUD
021b                   DELAYF         EQU    539                     DELAY AT FAST BAUD
                       *
                       ********
                       * THIS BOOTSTRAP PROGRAM ALLOWS THE USER TO
                       * DOWNLOAD A PROGRAM OF 0 - 2048 BYTES.
                       * THE PROGRAM MUST START AT $0000.
                       * EACH BYTE OF THE PROGRAM IS RECEIVED BY THE SCI.
                       * THE FIRST BYTE ESTABLISHES BAUD RATE.
                       * THEN THE PROGRAM IS DOWNLOADED STARTING WITH
                       * THE $0000 BYTE AND WORKING UP TOWARD THE END OF RAM
                       * A DELAY OF FOUR WORD TIMES (AT EITHER BAUD RATE)
                       * CAUSES THE RECEPTION OF CHARACTERS TO STOP AND
                       * A JUMP TO $0000.
                       *
                       * THE TRANSMITTER WILL BE USED FOR THE PURPOSE
                       * OF COMMUNICATION TO THE OUTSIDE WORLD.
                       *
                       ******************************************************************
bf00                                 ORG    $BF00
                       *
bf00                   BEGIN         EQU    *
                       * INIT STACK
bf00 8e 07 ff                        LDS    #RAMEND
                       * INIT X REG FOR INDEXED ACCESS TO REGISTERS
bf03 ce 10 00                        LDX    #$1000
                       ********
                       * PUT PORT D IN WIRE OR MODE
bf06 1c 28 20                        BSET   SPCR,X $20
                       * INIT SCI AND RESTART BAUD DIVIDER CHAIN
bf09 cc a2 0c                        LDD    #$A20C                  DIV BY 16
bf0c a7 2b                           STAA   BAUD,X
                       * RECEIVER & TRANSMITTER ENABLED
bf0e e7 2d                           STAB   SCCR2,X
                       ********
                       * TEST THE SECURITY BIT
bf10 1e 3f 08 45                     BRSET  CONFIG,X $08 NOSEC
                       ******************************************************************
                       * WE ARE IN SECURITY MODE
                       *
                       * OUTPUT $FF ON TRANSMITTER
bf14                   AGAIN         EQU    *
bf14 a6 2e                           LDAA   SCSR,X
bf16 86 ff                           LDAA   #$FF
bf18 a7 2f                           STAA   SCDAT,X
                       * ACCA NOW IS SET FOR $FF
                       *
                       * ERASE EEPROM:
                       * TURN OFF BLOCK PROTECT
bf1a 6f 35                           CLR    BPROT,X
                       * SET ERASE AND EELAT BITS BEFORE USING "ERASE"
bf1c 54                              LSRB                           CHANGE $0C TO $06
bf1d e7 3b                           STAB   PPROG,X
bf1f f7 b6 00                        STAB   EEPSTR                  WRITE EEPROM LOCATION
bf22 8d 21                           BSR    ERASE
                       * ACCB IS NOW SET FOR $06
                       *
                       ********
                       * ERASE CYCLE IS COMPLETE
                       *
                       * IF THE EEPROM IS NOT ENABLED,
                       * WE CAN'T CHECK THAT THE EEPROM IS ERASED
bf24 1f 3f 01 11                     BRCLR  CONFIG,X $01 NOEE
                       * EEPROM IS ON,
                       * NOW CHECK THAT THE EEPROM IS ERASED
bf28 18 ce b6 00                     LDY    #EEPSTR
bf2c                   LOOP          EQU    *
bf2c 18 a1 00                        CMPA   0,Y                     (A = $FF)
                       * ANY UNERASED BYTE SENDS US BACK TO ERASE AGAIN
bf2f 26 e3                           BNE    AGAIN
bf31 18 08                           INY
bf33 18 8c b8 00                     CPY    #EEPEND+1
```

```
bf37 26 f3                            BNE    LOOP
bf39                   NOEE           EQU    *
                       *******************************************************************
                       * WRITE OVER ENTIRE RAM, EXCEPT LAST TWO BYTES
                       * WHICH ARE USED BY THE STACK & $0000 WHICH IS
                       * LEFT INTACT
                       *
bf39                   ERAM           EQU    *
bf39 3c                               PSHX
bf3a ce 07 fd                         LDX    #RAMEND-2
bf3d                   LOP1           EQU    *
bf3d a7 00                            STAA   $00,X
bf3f 09                               DEX
bf40 26 fb                            BNE    LOP1
bf42 38                               PULX
bf43 20 0e                            BRA    ECONFG

                       *******************************************************************
                       * BOOTLOADER SUBROUTINES
                       *
                       * EEPROM ERASE SUBROUTINE
                       *
                       * ASSUMES CALLING ROUTINE HAS ALREADY SET ERASE
                       * AND EELAT BITS, AS WELL AS ACCESSED WHATEVER
                       * IS TO BE ERASED
                       *
                       * ENTRY X=$1000
                       * EXIT X=$1000, Y=$0000
                       *
bf45                   ERASE          EQU    *
                       * SET EEPGM BIT
bf45 1c 3b 01                         BSET   PPROG,X $01
                       * 10 MILLISEC DELAY @ 2.1 MHZ
bf48 18 ce 0b b8                      LDY    #3000
bf4c 18 09             BK1            DEY
bf4e 26 fc                            BNE    BK1
                       * TURN OFF ERASE AND EELAT BITS
bf50 6f 3b                            CLR    PPROG,X
bf52 39                               RTS

                       *******************************************************************
                       * NOW ERASE CONFIG REGISTER
                       *
bf53                   ECONFG         EQU    *
                       * SET ERASE AND EELAT BITS
bf53 e7 3b                            STAB   PPROG,X                (B STILL = $06)
                       * WRITE CONFIG REGISTER LATCH IT FOR ERASURE
bf55 e7 3f                            STAB   CONFIG,X
bf57 8d ec                            BSR    ERASE
                       ********
                       * ERASE CYCLE IS COMPLETE
                       *
                       *******************************************************************
                       * NON-SECURITY AND SECURITY MODES MEET HERE
                       *
bf59                   NOSEC          EQU    *
                       ********
                       * SET UP DELAY FOR FASTEST BAUD RATE
bf59 cc 02 1b                         LDD    #DELAYF
bf5c ed 16                            STD    TOC1,X
                       ********
                       * SEND BREAK TO SIGNAL START OF DOWNLOAD
bf5e 1c 2d 01                         BSET   SCCR2,X $01
                       * CLEAR BREAK AS SOON AS START BIT IS DETECTED
bf61 1e 08 01 fc                      BRSET  PORTD,X $01 *
bf65 1d 2d 01                         BCLR   SCCR2,X $01     CLEAR BREAK
                       * WAIT FOR FIRST CHARACTER (USERS SEND $FF)
bf68 1f 2e 20 fc                      BRCLR  SCSR,X $20 *    WAIT FOR RDRF
bf6c a6 2f                            LDAA   SCDAT,X                READ DATA
```

```
                       * IF DATA = $00 (BREAK OR $00), THEN JUMP TO EEPROM
bf6e 26 03                            BNE    NOTZERO
bf70 7e b6 00                         JMP    EEPSTR
bf73                   NOTZERO EQU    *
                       * IF DATA = $FF, THEN /16 IS CORRECT BAUD
bf73 81 ff                            CMPA   #$FF
bf75 27 08                            BEQ    BAUDOK
                       * ELSE CHANGE TO /104 (/13 & /8) 1200 @ 2MHZ
bf77 1c 2b 33                         BSET   BAUD,X $33
                       * SET UP DELAY FOR SLOWER BAUD RATE
bf7a cc 0d b0                         LDD    #DELAYS
bf7d ed 16                            STD    TOC1,X
                       *
bf7f                   BAUDOK         EQU    *
bf7f 18 ce 00 00                      LDY    #RAMSTR                 PNTR TO START OF RAM
                       *
                       * TIME EACH BYTE
bf83 ec 16             WAIT           LDD    TOC1,X                  PUT DELAY TIME IN ACCD
                       *
bf85 1e 2e 20 07       WTLOOP         BRSET  SCSR,X $20 NEWONE
bf89 8f                               XGDX                           DELAY INTO X
bf8a 09                               DEX                            DECREMENT DELAY
bf8b 8f                               XGDX                           RETURN DELAY TO ACCD
bf8c 26 f7                            BNE    WTLOOP
bf8e 20 0f                            BRA    STAR
                       * DID NOT TIME OUT
bf90                   NEWONE         EQU    *
                       * READ IN BYTE AND PUT INTO RAM
bf90 a6 2f                            LDAA   SCDAT,X
bf92 18 a7 00                         STAA   $00,Y
bf95 a7 2f                            STAA   SCDAT,X                 HANDSHAKE
bf97 18 08                            INY
bf99 18 8c 08 00                      CPY    #RAMEND+1
bf9d 26 e4                            BNE    WAIT
                       ******************************************************************
                       * START USER'S PROGRAM
                       *
bf9f                   STAR           EQU    *
bf9f 7e 00 00                         JMP    RAMSTR
                       ******************************************************************
                       * FILL UNUSED BYTES WITH ZERO
                       *
bfa2 00 00 00 00 00 00                BSZ    $BFBB-*
     00 00 00 00 00 00
     00 00 00 00 00 00
     00 00 00 00 00 00
     00
                       *
                       ******************************************************************
                       * REVISION LEVEL IN ASCII
                       *       (ORG   $BFB)
bfbb 41                               FCC    "A"
                       ******************************************************************
                       * MASK I.D. ($0000 FOR EPROM PARTS)
                       *       (ORG   $BFB)
bfbc 00 00                            FDB    $0000
                       ******************************************************************
                       * 11E32 I.D. - can be used to determine MCU type
                       * Bit 15 is a 0 if the part is ROM (or ROMless,
                       * 1 -> EPROM)
                       * Bit 14 is a 0 if unsecured
                       * Bits 13 - Bit 9 are lower 5 bits of 1st ASCII letter
                       * Bits 8 - Bit 6 are $7 => last six bits are number
                       * Bits 5 - Bit 0 are last digit of part number
                       * (note: $45 = E in ASCII)
                       *
                       *     15 14 13   9 8 6 5    0
                       * i.e. 0  1  00101 111 100000        (4BE0)
                       *    ROM SEC  'E'  nul   32
                       *
                       *       (ORG   $BFB)
bfbe 4b e0                            FDB     %0100101111100000
                       *******************************************************************
                       * VECTORS  ($BDC0)
                       *
bfc0 00 00                            FDB    $0000                   reserved
bfc2 00 00                            FDB    $0000                   reserved
bfc4 00 00                            FDB    $0000                   reserved
bfc6 00 00                            FDB    $0000                   reserved
bfc8 00 00                            FDB    $0000                   reserved
bfca 00 00                            FDB    $0000                   reserved
bfcc 00 00                            FDB    $0000                   reserved
bfce 00 00                            FDB    $0000                   reserved
bfd0 00 00                            FDB    $0000                   reserved
bfd2 00 00                            FDB    $0000                   reserved
bfd4 00 00                            FDB    $0000                   reserved
bfd6 00 c4                            FDB    $100-60                 SCI
bfd8 00 c7                            FDB    $100-57                 SPI
bfda 00 ca                            FDB    $100-54                 PULSE ACCUM INPUT EDGE
bfdc 00 cd                            FDB    $100-51                 PULSE ACCUM OVERFLOW
bfde 00 d0                            FDB    $100-48                 TIMER OVERFLOW
bfe0 00 d3                            FDB    $100-45                 TIMER OUTPUT COMPARE 5
bfe2 00 d6                            FDB    $100-42                 TIMER OUTPUT COMPARE 4
bfe4 00 d9                            FDB    $100-39                 TIMER OUTPUT COMPARE 3
bfe6 00 dc                            FDB    $100-36                 TIMER OUTPUT COMPARE 2
bfe8 00 df                            FDB    $100-33                 TIMER OUTPUT COMPARE 1
bfea 00 e2                            FDB    $100-30                 TIMER INPUT CAPTURE 3
bfec 00 e5                            FDB    $100-27                 TIMER INPUT CAPTURE 2
bfee 00 e8                            FDB    $100-24                 TIMER INPUT CAPTURE 1
bff0 00 eb                            FDB    $100-21                 REAL TIME INT
bff2 00 ee                            FDB    $100-18                 IRQ
bff4 00 f1                            FDB    $100-15                 XIRQ
bff6 00 f4                            FDB    $100-12                 SWI
bff8 00 f7                            FDB    $100-9                  ILLEGAL OP-CODE
bffa 00 fa                            FDB    $100-6                  COP FAIL
bffc 00 fd                            FDB    $100-3                  CLOCK MONITOR
bffe bf 00                            FDB    BEGIN                   RESET


                       *********************** TBRE32.ASC 30/6/93 ************************
                       * Motorola Copyright 1988,1990,1992,1993                              *
                       * MCU resident,InterruptdrivenCommunicationroutines for 68HC11 mon-   *
                       * itor.Provides low level memory and stack read/write operations.     *
                       *                                                                     *
                       *   This talker DOES NOT use XIRQ                                     *
                       *   -----------------------------                                     *
                       *                                                                     *
                       * N.B.TBRE32 is designed to work with the 68HC11E32 or other          *
                       *   compatible MCU types.This version of the TALKER is designed to    *
                       *   execute from MC68HC11E32 Boot ROM.                                *
                       *   To initiate communication with TBRE32,the standard bootloader     *
                       *   must be used to initialise the redirected vector table and then   *
                       *   cause a jump to USERSTART                                         *
                       *
                       * CONSTANTS
be40                   TALKBASE       equ    $BE40
00c4                   BOOTVECT       equ    $00C4                   Start of boot vectors
03ff                   STACK          equ    $03FF                   User may alter this
1000                   REGBASE        equ    $1000
                       *
00c4                   JSCI           equ    $00C4
00f1                   JXIRQ          equ    $00F1
00f4                   JSWI           equ    $00F4
00f7                   JILLOP         equ    $00F7
00fa                   JCOP           equ    $00FA
007e                   JMPEXT         equ    $7E                     Mnemonic for jump ext
004a                   BRKCODE        equ    $4A                     Break point code
004a                   BRKACK         equ    $4A                     Break point ack
```

```
                       *
                       * REGISTERS
002f                   SCDR           equ    $2F
                       *
0020                   RDRF           equ    $20
0080                   TDRE           equ    $80
0008                   OR             equ    $08
0002                   FE             equ    $02
                       *
                       * PROGRAM
be40                                  org    TALKBASE
                       *
be40                   TLKRSTART EQU  *      Initialise SCI int
be40 86 7e                            LDAA   #JMPEXT
be42 97 c4                            STAA   JSCI
be44 ce be 5e                         LDX    #SCISRV
be47 df c5                            STX    JSCI+1
be49                   USERSTART   EQU*
be49 8e 03 ff                         LDS    #STACK
be4c ce 10 00                         LDX    #REGBASE
be4f 6f 2c                            CLR    SCCR1,X
be51 cc 30 2c                         LDD    #$302C                  Init SCI to 9600 baud,
be54 a7 2b                            STAA   BAUD,X                   no parity, no int
be56 e7 2d                            STAB   SCCR2,X                 & enable SCI tx & rx.
be58 86 40                            LDAA   #$40                    Enable STOP, and I bit
be5a 06                               TAP                            int, disable XIRQ.
                       *                                             Now hang around for
be5b 7e be 5b          IDLE           JMP    IDLE                     SCI int from host.
                       * A RESET from host changes above jump destination to
                       *  start of user code.
be5e                   SCISRV         EQU    *                       On detecting int
be5e b6 10 2e                         LDAA   SCSR+REGBASE            assume rx caused it.
be61 84 20                            ANDA   #RDRF
be63 27 f9                            BEQ    SCISRV                  otherwise loop here
                       *
be65                   RXSRV          EQU    *                       Process received data.
be65 b6 10 2f                         LDAA   SCDR+REGBASE            Read cmd byte, & tx it
be68 43                               COMA                            inverted
be69 8d 40                            BSR    OUTSCI                   as ack to host.
be6b 2a 4b                            BPL    INH1                    Bit 7 set? => inherent
be6d 8d 2d                            BSR    INSCI                   else B = byte count
be6f 8f                               XGDX                           Save cmd and byte cnt.
be70 8d 2a                            BSR    INSCI                   Read high address byte
be72 17                               TBA                            into ACCA
be73 8d 27                            BSR    INSCI                   then B = low addr byte
be75 8f                               XGDX                           A=cmd; B=cnt; X=addr
be76 81 fe                            CMPA   #$FE
be78 26 0d                            BNE    RXSRV1                  If cmd is mem read ...
                       *
be7a                   TREADMEM       EQU    *                       REPEAT
be7a a6 00                            LDAA   ,X                       read required address
be7c 8d 2d                            BSR    OUTSCI                   send it to host
be7e 17                               TBA                             (save byte count)
be7f 8d 1b                            BSR    INSCI                   and wait for acknowledge
be81 16                               TAB                             (restore byte count)
be82 08                               INX                             Increment address
be83 5a                               DECB                            Decrement byte count
be84 26 f4                            BNE    TREADMEM                UNTIL all done
be86 3b                               RTI                            & return
                       *
be87                   RXSRV1         EQU    *
be87 81 be                            CMPA   #$BE
be89 26 10                            BNE    RXSRVEX                 If cmd is mem write...
                       *
be8b 17                               TBA                            move byte count to A
be8c                   TWRITMEM       EQU    *                       REPEAT
be8c 8d 0e                            BSR    INSCI                    Read next byte into B
be8e e7 00                            STAB   ,X                       and store at req addr
be90 8d 18                            BSR    EPRG
```

```
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          OUTSCI1        LDAA  SCSR+REGBASE
0407 beb0 2a fb                            BPL   OUTSCI1                 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 bebc 30                INH1A          TSX                           Move SP to X
0417 bebd 8f                               XGDX                          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                               XGDX                          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
```

```
bef1 ed 07                            STD    7,X                     return from breakpoint
bef3 20 c7                            BRA    INH1A                   but first return all
                       *                                             MCU registers to host
                       *************************************************************
                       * FILL UNUSED BYTES WITH ZERO
                       *
bef5 00 00 00 00 00 00                BSZ    $BF00-*
      00 00 00 00 00
                       *
                       *************************************************************

                                      END
```

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