

# S12 All-Access Bootloader for the HCS12 Microcontroller Family

by: Rafael Peralez and Gabriel Sanchez  
RTAC Americas

## 1 Introduction

This application note covers the operation and use of the S12 all-access bootloader for the HCS12 microcontroller family. A bootloader is a program that runs at the time a MCU starts which may load software, or may call software to run. This bootloader enables the user to program and erase the Flash and EEPROM memory. Thus, enabling the user to reprogram firmware without the need of special programming adaptors.

### 1.1 Overview

Although to originally Flash a bootloader into the MCU Flash the user needs a special programming adaptor, the biggest advantage of using a bootloader is that MCUs can then be reprogrammed very easily and with as little special hardware as possible. Because all HCS12 MCUs include an on-chip serial communications interface (SCI), almost no additional external hardware is required to communicate with a computer. However, there is one exception, an RS-232 level translator chip so that the

## Contents

1	Introduction . . . . .	1
1.1	Overview . . . . .	1
2	Functional Description . . . . .	2
2.1	Hardware Configuration . . . . .	2
2.2	Communications . . . . .	3
2.3	Oscillator . . . . .	3
2.4	Computer Operating Properly (COP) . . . . .	3
2.5	Memory Configuration . . . . .	4
2.6	Interrupt Use . . . . .	5
2.7	Flash Explanation . . . . .	5
2.8	Flash Protection . . . . .	5
2.9	Backdoor Key Explanation . . . . .	5
2.10	EEPROM Explanation . . . . .	6
2.11	EEPROM Protection . . . . .	6
2.12	Auto Update . . . . .	6
2.13	Constraints and Considerations . . . . .	7
3	User Guide . . . . .	8
3.1	Setup Procedure . . . . .	8
3.2	S12 All-Access Bootloader Guide . . . . .	8
3.3	User Software Guide . . . . .	13

## Functional Description

MCU may communicate with a PC. Yet, in many systems, the level translator chip may already be part of the system design since the SCI may be used as a diagnostic port. If a RS-232 is not part of the basic system design, a small adaptor board may be used when using the bootloader.

In addition to the SCI port used, a single input pin is needed to tell the bootloader whether to execute the bootloader code or to jump to the user application. This may be done with a switch, pushbutton, or jumper. An advantage is that if the bootloader begins to run user code, that pin may still be used as an output pin.

When executing from the bootloader code, the user has the ability to:

- Program or erase user Flash and EEPROM
- Secure the MCU using the backdoor keys
- Protect the EEPROM memory

### NOTE

When programming the user Flash the user may use both banked and non-banked S19s.

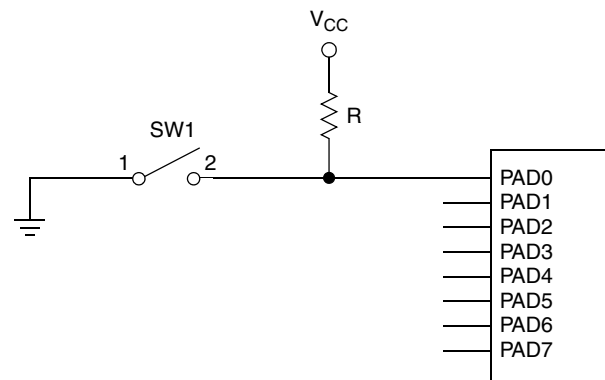
One major advantage for this bootloader is that the same .s19 file that is generated by the compiler may be used on all HCS12s devices. What this means is that the user can put the all-access S12 bootloader into any HCS12 MCU without worrying about the EEPROM and Flash size or its location in memory since the bootloader takes care of it. The bootloader only redirects the system reset vector. This is an important feature when developing code because once the user executes their code, there will be no latency in executing an interrupt routine.

## 2 Functional Description

### 2.1 Hardware Configuration

After reset, the S12 all-access bootloader's startup routine is called. In this routine the status of the PAD0 pin is read. If the value of this pin is a logical one, the user's startup routine is called, but if the value is zero the Bootloader's startup routine is the one called. This is done in the startup.c file; however, the user can modify the code to select a different input to determine its code. If a different code is not selected, the bootloader is the one that is going to be executed.

A switch, jumper, or pushbutton may be placed on this pin in order to have the option of calling the bootloader section or running the user's code. The bootloader uses the internal pullup found on the MCU pin. See [Figure 1](#).



**Figure 1. Connection Used to Select Either User's or Bootloader Code Execution**

## 2.2 Communications

Communications are made using the HCS12's SCI. The MCU uses an 8-data bit, no parity, and stop bit protocol without flow control.

The baud rate that the MCU will operate at will depend on the crystal or oscillator used. With this bootloader, the maximum baud rate allowed is 19,200 bps. [Table 1](#) describes baud rates in regards to typical oscillator frequencies.

**Table 1. Oscillator Frequency Baud Rates**

Oscillator Frequency	Baud Rate
2 MHz	2400 bps
4 MHz	4800 bps
8 MHz	9600 bps
16 MHz	19,200 bps

Through the use of a HyperTerminal at a specified baud rate, a user may send commands to the MCU to access the different menus and use all the features of the S12 bootloader.

## 2.3 Oscillator

The oscillator must be of a value greater than 2 MHz in order to program and erase the Flash. Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the Flash cannot be performed if the external clock reference is running on a frequency smaller than 2 MHz.

At the beginning of the bootloader, the value of the oscillator is asked for in kilohertz. This is done with the intention of configuring the Flash clock divider and making it possible to program and erase the Flash. It is very important to set the right value of the oscillator in order to set the right value in the Flash clock.

The main reason for setting the right oscillator value is:

- If the Flash clock frequency is less than 150 kHz, the Flash could be destroyed due to overstress.
- If the Flash clock frequency is greater than 200 kHz, incomplete programming or erasing of the cells can result.

### NOTE

The bus frequency is the external reference divided by 2.

## 2.4 Computer Operating Properly (COP)

Because of the routines used in the bootloader, the COP is disabled during the whole execution of its routines. The COP remains disabled if the user's code is called (see [Section 2.1, "Hardware Configuration"](#)). If the user's code needs a watchdog, this must be set in its code by writing the COPCTL register along with the desired timeout value.

## 2.5 Memory Configuration

Since the S12 all-access bootloader allows programming of the EEPROM and also needs to know the RAM location, it is necessary to remap the memory in order to make this bootloader usable for the different S12 derivatives. It is necessary to consider that there are S12 derivatives with or without 1-K, 2-K or 4-K of EEPROM, and that these derivatives have different RAM sizes.

The microcontroller’s registers start at address 0x0000. During the bootloader’s startup routine, the MCU relocates the beginning of the EEPROM to address 0x2000. Also, the RAM is relocated to address 0x5000 (see Figure 2). By doing this, the bootloader always knows where the EEPROM and RAM base address are located. No matter the size of EEPROM or RAM that the microcontroller has, the bootloader is able to access both in any of the S12 derivatives. Having access to the EEPROM and RAM makes it possible to:

- Run routines in RAM memory which allow the programming and erasing of all the Flash locations of each derivative
- Program EEPROM when present

In user’s code, registers, RAM, and EEPROM start addresses may be remapped to wherever the user may see fit.

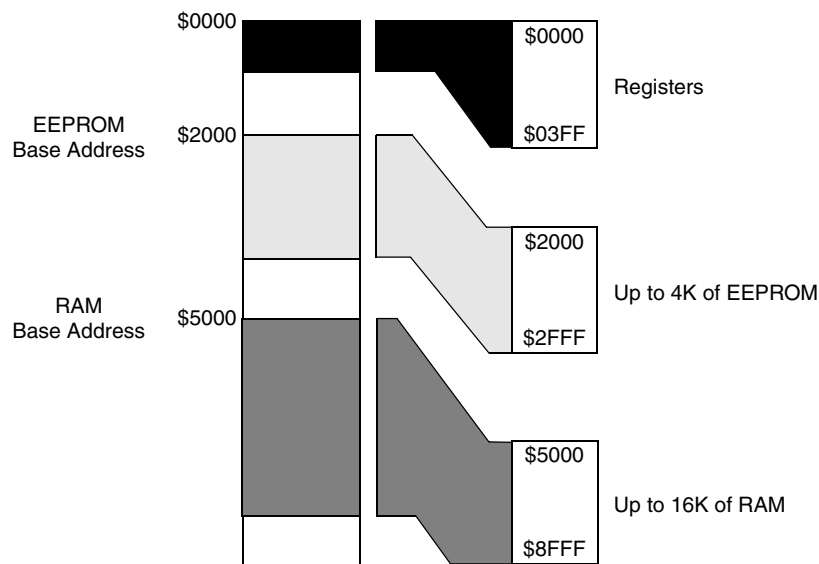


Figure 2. Bootloader Memory Map

The RAM will be used supposing that worst case is present. The smallest RAM in the S12 derivatives is 1K. After remapping the RAM, the bootloader will use the following 1024 bytes for its buffers, variables, and stack.

## 2.6 Interruption Use

The S12 all-access bootloader allows the user to use all interrupt vectors normally, except for the system reset vector. The system reset vector will have a delay of 15 cycles from what it would have in a typical application. This delay is the amount of cycles that the MCU takes to check if it should enter the bootloader or jump into user program code.

One thing to note is that the interrupt vectors reside in the highest page of memory, and in order to change them the bootloader program must erase and reprogram the last page of memory. This creates a problem if anything goes wrong during the reprogramming since the reset vector to the bootloader may not be restored correctly. If this occurs, it would be impossible to recover without the use of special programming hardware. For this reason, it is important that the user keep a steady MCU input voltage and avoid any ESD onto the board to avoid this problem.

## 2.7 Flash Explanation

The HCS12 family uses an advanced, third generation, nonvolatile Flash EEPROM memory that is used to store application program code and constant data. The Flash memory can be erased and reprogrammed many times over and is ideally suited to the development phase of a product. Flash memory is also suitable for the production phase, as product inventories can be reduced by having a common microcontroller for similar products. Any software changes, upgrades, or fixes can be implemented immediately during production, without the delay and costs associated with a new ROM mask. Furthermore, products in the field can be reprogrammed as required without having to replace the microcontroller. Over the product life span Flash offers significant potential cost savings when compared to ROM.

## 2.8 Flash Protection

No hardware Flash protection is selected — this allows the reprogramming of interrupt vectors, securing and editing backdoor keys, and upgrading the bootloader. Because of this, special software considerations have been taken into consideration in the bootloader. If the user's code tries to write in a section between 0xF000 to 0xFF00, the bootloader will display an error indicating that the code sent tried to write in the bootloader's section and this section of the code will not be written. All the interrupt vectors except the reset vector will be written in their normal locations. The reset vector will be written into address 0xEFFE because the bootloader must own the reset vector to be able to determine if the user's code or the bootloader code is the one to be called.

## 2.9 Backdoor Key Explanation

The HCS12 family has a memory security feature that enables the user to protect intellectual property by preventing unauthorized access to the NVM. By using the different menus of the bootloader, the user can set the backdoor keys and secure the microcontroller. In some S12 derivatives, securing the microcontroller makes it impossible to perform certain operations. So, it is important to check the mask of the microcontroller and read the different errata sheets to know if write and erase functions can be performed if the microcontroller is secured. This is very important since no error is displayed when a write operation fails due to the chip security. The security feature still allows the user to program and erase user Flash, so a new user application may be loaded while the device is secured.

It is important to note again that the Flash security register, along with the backdoor keys, are located in the highest page of memory. So, in order to modify any of them, the bootloader must erase and reprogram the last page of Flash where the reset vector is also located. This creates a problem if anything goes wrong during the reprogramming since the reset vector for the bootloader may not be restored correctly. If this occurs, it would be impossible to recover without the use of special programming hardware. For this reason, it is important that the user keep a steady MCU input voltage and avoid any ESD onto the board to avoid this problem.

## 2.10 EEPROM Explanation

Most HCS12 microcontrollers also incorporate EEPROM that may be used to store data variables. HCS12 microcontrollers that do not have EEPROM may use Flash to emulate EEPROM, refer to application note AN2302 for details and example software. The EEPROM on HCS12 microcontrollers is constructed using the same basic technology as the Flash memory, but with some adjustments to make it more suitable for data storage applications. The most obvious of these is the erase sector size, which is 4 bytes.

Once programmed, the EEPROM retains data until it is erased and reprogrammed. The EEPROM can be erased and reprogrammed many times over, refer to the specific microcontroller's electrical specifications for current data retention and write/erase endurance figures.

## 2.11 EEPROM Protection

The S12 all-access bootloader makes it possible for the user to protect and unprotect the EEPROM memory. This enables the user to be able to keep the user program from accidentally erasing or programming the EEPROM.

### NOTE

While the EEPROM is protected, the user will not be able to erase or program the EEPROM.

## 2.12 Auto Update

By adding the auto update feature, it is possible to generate a new version of the bootloader and update it without the need of an external programmer. This feature allows improved functionality, generates new options, removes or optimizes functions, etc.

When this option is selected the new bootloader is saved temporarily in the 4K previous to the location of the bootloader that is running at the time (from 0xE000 to 0xEFFF). When the new bootloader is fully received the program jumps to the new update function. This function:

- Erases the previous bootloader (0xF000 to 0xFFFF)
- Copies the new received bootloader from the temporary section to the bootloader Flash space
- Executes a mass erase function with the intention of erasing the temporary bootloader created
- Finishes this option with all of the Flash memory available for writing

An important thing to consider when using the Auto Update feature is that once the new bootloader version is fully copied into the memory section of the bootloader (see Figure 3) all the contents of the Flash memory will be erased (either program or data). So, any application that was previously stored should be programmed again after updating the bootloader.

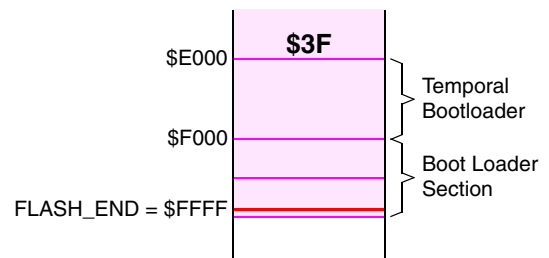


Figure 3. Temporal Location to Update

## 2.13 Constraints and Considerations

The bootloader, by its generic nature, has no knowledge of each user's application. This results in certain compromises and limitations of which the user should be aware.

1. Due to the memory mapping used by the bootloader, the user cannot use the Flash programming and erase routines implemented in the bootloader.
2. The bootloader does not change the state of any I/O following reset. It reads the level of one input to determine if the user's code, or if the bootloader is going to be executed. After this, the user can use this pin as a general-purpose output or leave it unused.
3. The bootloader does not use the COP watchdog timer; therefore, the bootloader doesn't activate it. If the user wants to use the COP watchdog timer, then it must be enabled within the user's application code.
4. Although the software denies the programming of user code into bootloader space, the user application code should not try to overwrite the section where the bootloader resides.
5. There is always a risk when programming an application into Flash memory using downloaded programming algorithms that an external disturbance may corrupt the programming process. It may not be possible to recover from this in all cases. Certain considerations can help minimize the risk of corruption. The higher risk operations are those that modify the following:
  - The MCU reset vector — While the reset vector is erased, a reset will not restart the bootloader. If the bootloader is to remain resident, then erasing and programming the reset vector sequentially will minimize the window during which a reset could cause the bootloader function to become corrupted.
  - The application start vector — With the bootloader resident, once the application start vector is programmed, the bootloader function following reset is disabled. If a premature reset or a Flash write issue occurs, a reset will then cause a jump to an incomplete application. Programming the application start vector as the last operation can minimize this risk.
  - The bootloader code — If, for some reason, it is necessary to erase the bootloader code from Flash, a premature reset or Flash programming problem will fail to successfully restart the bootloader process.

In any of these three cases, recovery can be facilitated by designing easy access to the signals required for interfacing a BDM interface cable to the MCU.

## 3 User Guide

### 3.1 Setup Procedure

1. Load S12 all-access bootloader S19 into the MCU
2. Connect to serial communications
3. Open Hyperterm to a specified baud rate depending on OSC. Baud rate should not be greater than that specified in this bootloader application note.
4. Configure hardware pins so that the MCU will enter bootloader mode
5. Turn on the MCU

### 3.2 S12 All-Access Bootloader Guide

#### 3.2.1 Setting the Crystal or Oscillator Value

The only thing that needs to be configured for the bootloader is the value of the crystal or oscillator used. The speed must be set in kilohertz. This configuration is used because it is easier to get a more accurate value when using this unit of measurement instead of megahertz.

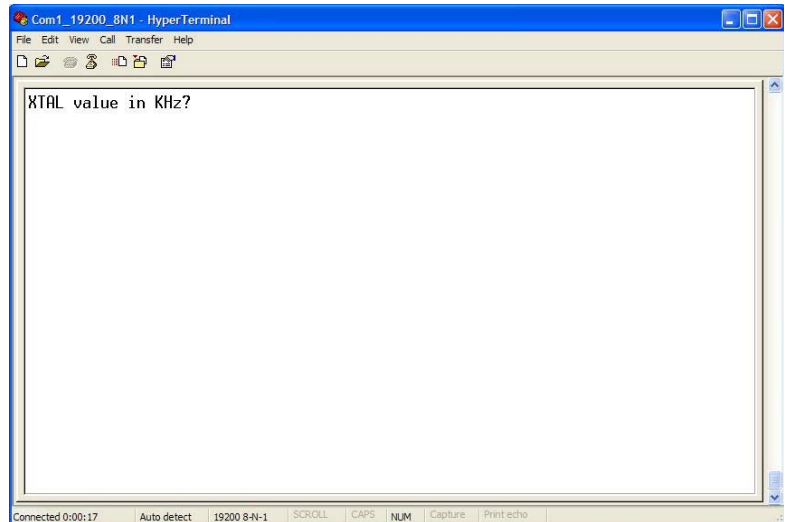


Figure 4. Setting the Crystal or Oscillator



## 3.2.2 Navigating through the Loader

In all the windows displayed in the HyperTerminal, the bootloader will show all the available options for the current menu. Each of the options will be associated to a number. This number is the one that you have to press on the keyboard in order to access the desired menu, or to perform the selected option in the bootloader.

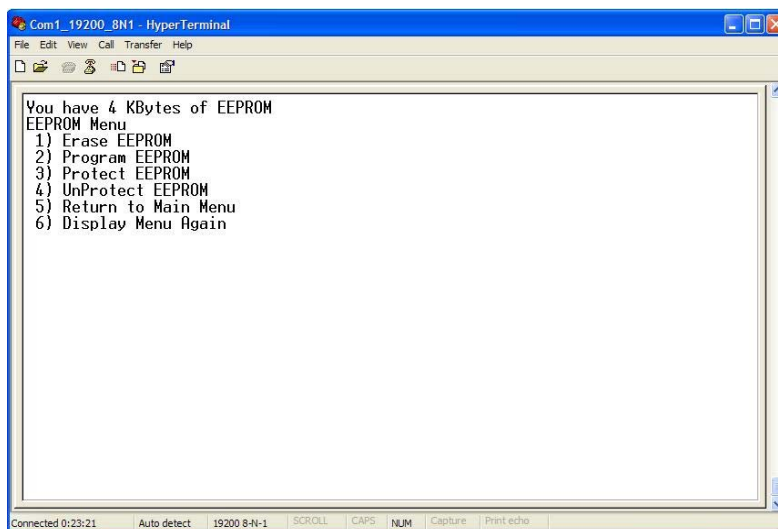


Figure 5. Example of a Bootloader Menu

## 3.2.3 Erase Flash

In order to erase the Flash from a device you have to select option number 1 in the first appearing menu. The bootloader will erase all the Flash except for the bootloader area of the device and will send a message once the erase command is fully executed. If the Flash of the device is either protected or not working properly, the bootloader will send an error message that the command could not be executed.

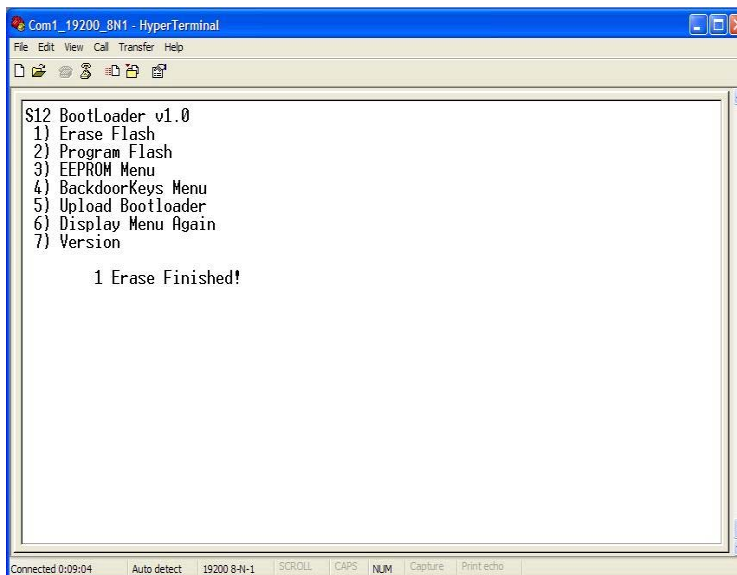


Figure 6. Erase Flash Window

### 3.2.4 Program Flash

To program the Flash of a device you need to select option number two in the first menu of the bootloader. Once you select this option, a message will appear telling you to send the S19 record that you wish to program. In order to do so, you have to click on the Transfer Menu of the HyperTerminal and then click on Send Text file. In the opening window, select to display all type of files and then look for the S19 record of your project (typically has the name of the project with extension .abs.s19 located in the bin folder of your project)

An important thing to notice in this option is if the user's code tries to write into the memory section reserved for the bootloader (from 0xF000 to 0xFF00). If this happens, the bootloader will display an error message and the code written into the device won't work properly.

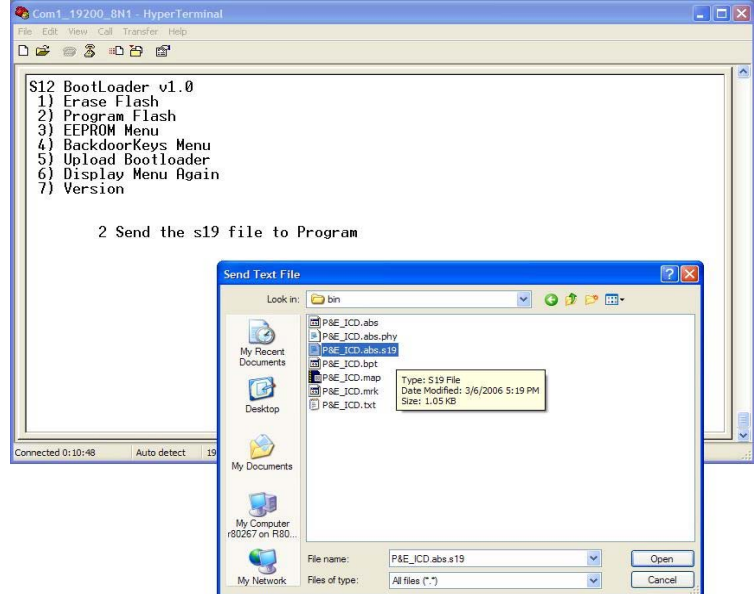


Figure 7. Programming Flash Menu

### 3.2.5 EEPROM Menu

When the user enters the EEPROM Menu, the bootloader will display the amount of EEPROM memory allocated in the device. If the device has EEPROM, a menu will display that will help the user to store and erase data in this memory. If the device doesn't have EEPROM, the bootloader will display that the MCU has 0 Kbytes of EEPROM and wait for the user to press any key. Once a key is pressed, the bootloader will display the main menu again.

The EEPROM Menu also has options to display again the options of the menu and an option to return to the main menu if the user has finished his EEPROM operations.

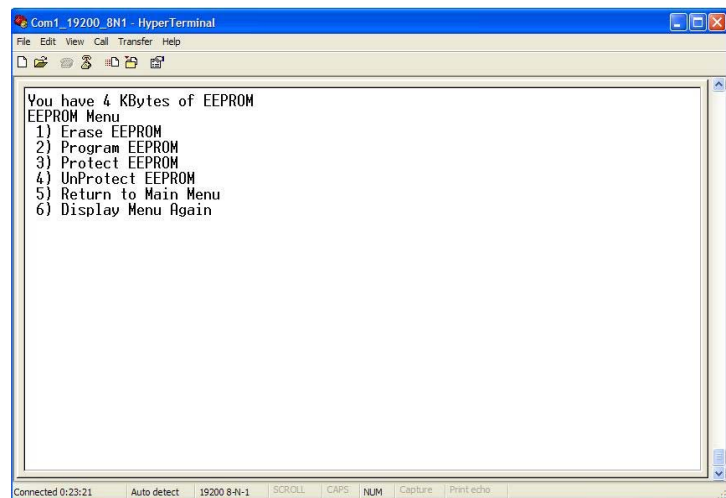


Figure 8. EEPROM Menu

### 3.2.6 Erase and Program EEPROM

The erase and program options of the EEPROM are identical to those options for Flash. When choosing to erase the EEPROM, all the contents of the memory will be erased. If you choose to program the EEPROM, you need to send a file with the same format as the S19 record with the contents of the data that the user wishes to program into EEPROM. The same steps described to program the device Flash should be followed to use this option (see [Section 3.2.4, “Program Flash”](#)). The addressing in the EEPROM is such that if the user wants to program the first byte of the EEPROM, then the S19 should try to write to address 0x0000, and so on until the end of the EEPROM is reached.

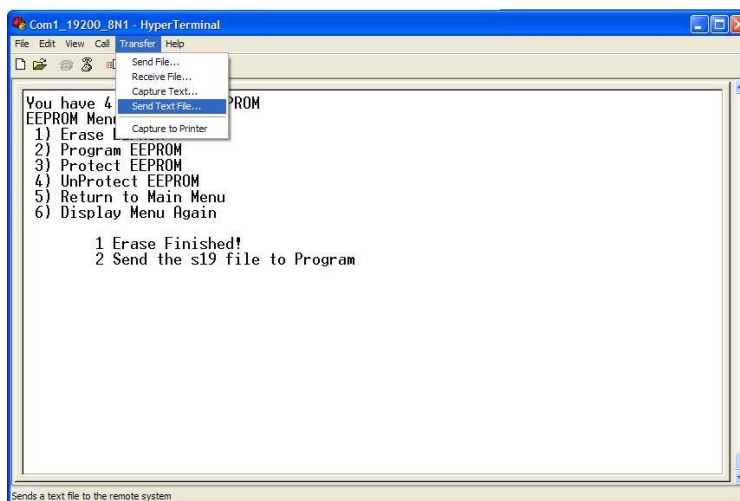


Figure 9. Erase and Program EEPROM

### 3.2.7 Protecting and Unprotecting the Contents of the EEPROM

The EEPROM has the option to protect the contents of the memory against accidental program or erase. This bootloader gives access to this feature.

To protect the EEPROM, the user only has to choose the third option in the EEPROM Menu and the contents of the memory will be protected. To disable this protection option 4 should be selected.

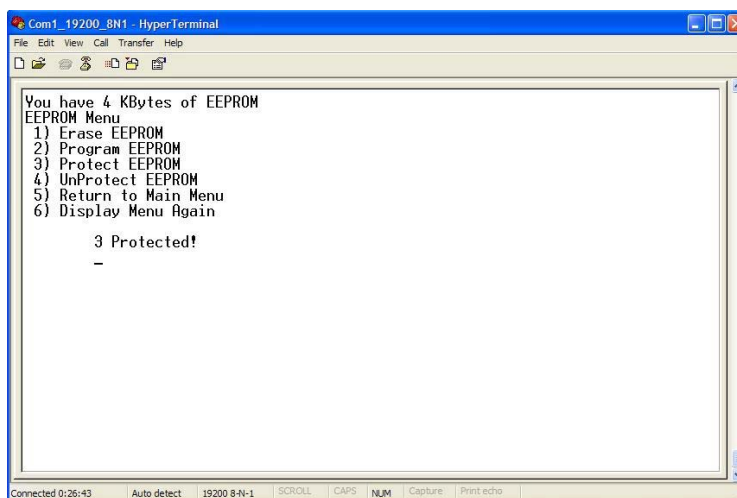


Figure 10. Using the Protection of the EEPROM

## 3.2.8 Backdoor Access Menu

When the user enters the Backdoor Access Menu, the bootloader will display the Backdoor Key Menu. This menu will show the possible executable commands in regards to the backdoor keys.

The Backdoor Access Menu also has options to display again the options of the menu and an option to return to the main menu if the user has finished his EEPROM operations.

### 3.2.9 Lock

The Lock option of the Backdoor Access Menu permits the user to enable the HCS12's backdoor key security feature. Whenever the user enters this option and the MCU is reset, the MCU will be locked.

### 3.2.10 Unlock

The Unlock option of the Backdoor Access Menu permits the user to disable the HCS12's backdoor key security feature. If the MCU is not locked, the bootloader will say that the new security has been applied. Yet if the MCU is locked, then it will ask for the backdoor key to try to unsecure the MCU. If the MCU is successfully unsecured, the bootloader will prompt saying that the new security was applied. If not, it will prompt saying that it was unable to unsecure the MCU.

### 3.2.11 Change Backdoor Key

The Change Backdoor Key option of the Backdoor Access Menu permits the user to change the backdoor key. When this option is selected and the MCU is not secured, the current backdoor key that is stored in Flash will be displayed. If the MCU is secured, the bootloader will ask for the user to enter the current backdoor key. Then the bootloader will try to unsecure the MCU. If the MCU is successfully unsecured, the bootloader will prompt saying that the MCU was successfully unsecured, and then ask for a new backdoor key. If the MCU could not be unsecured, then the bootloader will prompt saying that it was unable to unsecure the MCU and will not prompt for a new key.

### 3.2.12 View Backdoor Key

The View Backdoor Key option of the Backdoor Access Menu permits the user to view the current backdoor key whenever the MCU is not secured. When this option is selected, the prompt will display the current backdoor key stored in Flash if the MCU is not secured. If the MCU is secured, then the prompt will say that the backdoor key is enabled and will not display the backdoor key.

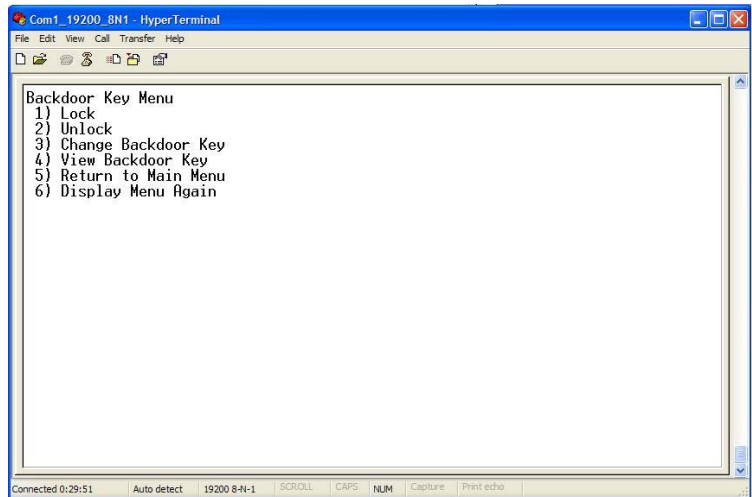


Figure 11. Backdoor Key Menu

### 3.2.13 Update Bootloader

When this option is selected, a message will appear telling you to send the .s19 for the bootloader record that you wish to program. In order to do so you have to click on the Transfer Menu of the HyperTerminal and then click on Send Text file. In the opening window select display all type of files and then look for the S19 record of your bootloader project (typically has the name of the project with extension .abs.s19 located in the bin folder of your project), just as in [Section 3.2.4, “Program Flash”](#) and [Section 3.2.6, “Erase and Program EEPROM”](#).

#### NOTE

Once the programming is finished, the MCU will do a mass erase and then reset itself. So, please be patient with this option.

### 3.2.14 Version

When the Version option is selected, the bootloader will display the version of the bootloader that is being used. This option may be very useful for a user to see exactly what build of a bootloader a MCU has.

## 3.3 User Software Guide

A user can easily make a CodeWarrior project that can be loaded into the S12 all-access bootloader. All the user needs to do is to create a new project or to use an existing project, and do the following:

Modify the ROM\_C000 segment in the project .prm file so that it does not invade the bootloader code like this:

```
ROM_C000 = READ_ONLY 0xC000 TO 0xEFFD;
```

This step is convenient if you are using Processor Expert, or if you wish to place an existing project into the S12 bootloader. This assures that the user code will not be placed within the address space where the bootloader code resides.

It is important to note that the stack size hasn't changed. The user may go into the .prm file and change the stack size to whatever size may be needed for the individual application. Once the user wishes to download the project onto the MCU, the user needs to run the bootloader and program the .s19 file that is generated by CodeWarrior. Once that is done, the user code will begin to run.

### 3.3.1 Acronyms and Abbreviations

MCU — Microcontroller Unit

NVM — Nonvolatile Memory

EEPROM — Electrically Erasable Programmable Read Only Memory

ESD — Electrostatic Discharge

### 3.3.2 References

Freescale application note entitled *HCS12 NVM Guidelines* (order number AN2400).

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **E-mail:**

[support@freescale.com](mailto:support@freescale.com)

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### **For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or 303-675-2140  
Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 that may be provided in Freescale Semiconductor 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. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor 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 Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor 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 Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Document Number: AN3275

Rev. 0

05/2006

