
Embedded Bootloader

User's Guide

MC13192FLUG/D
Rev. 2.3, 09/2004

How to Reach Us:

USA/Europe/Locations Not Listed:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:

Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

Home Page:

www.freescale.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.

Learn More: For more information about Freescale products, please visit www.freescale.com.

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. 2004. All rights reserved.

Contents

About This Book	v
Audience	v
Organization.....	v
Conventions	v
Definitions, Acronyms, and Abbreviations.....	v
References.....	vi
Revision History	vi
Chapter 1 Embedded Bootloader Description	1-1
1.1 Target.....	1-1
1.1.1 Ported Targets.....	1-2
1.2 Features.....	1-2
1.2.1 Optional Firmware Upload Settings.....	1-3
1.3 Benefits	1-4
1.4 Potential Issues	1-4
Chapter 2 Using the Embedded Bootloader.....	2-1
2.1 Upload Firmware	2-1
2.1.1 Use Case One.....	2-1
2.1.2 Use Case Two	2-2
2.1.3 Safe Mode Boot	2-2
2.2 Updating Non-volatile Memory (NVM)	2-3
2.2.1 An Example of How to Change the MAC Address	2-3
2.3 Changing The System Bus Frequency	2-3
Chapter 3 Test Tool – Zigbee Flash Tool.....	3-1
3.1 Graphical User Interface (GUI) Version.....	3-2
3.1.1 Using the Tools Menu Option.....	3-2
3.1.2 Using the View Menu Option	3-3
3.1.3 Selecting the Firmware File to Download	3-4
3.1.4 Using the Help Menu.....	3-6
3.1.5 Changing the NVM Data in Flash or File.....	3-6
3.2 Command Line Version (CMD).....	3-8
3.2.1 Normal Use (Default) Example	3-9

3.2.2	Flash Erase Disabled Example.....	3-9
Chapter 4 Public Function Description		4-1
4.1	Compiler Defines Application	4-1
4.2	Compiler Defines Embedded Bootloader	4-1
4.3	Embedded Bootloader Support Source Code	4-1
4.4	DigiType.h	4-1
4.4.1	Gb60_io.h	4-1
4.4.2	Crt0.c and Crt.h	4-2
4.4.3	FreeLoader_inf.c and FreeLoader_inf.h	4-2
4.4.4	NV_Data.c and NV_Data.h	4-2
4.4.5	ISR Vector Table Location	4-2
4.4.6	Reset Vector Location	4-2
4.4.7	System Clock Setup.....	4-2
4.5	802.15.4 Accessible Functions	4-3
4.6	Enable_Download_Firmware	4-3
4.7	Hard_Reset.....	4-5
4.8	Update_NV_RAM.....	4-5
4.9	NV_Flash_Setup.....	4-6
4.10	FL_ICG_Setup.....	4-6
Chapter 5 Embedded Bootloader Memory Map		5-1

About This Book

This guide provides a detailed description of Freescale's Embedded Bootloader.

The Embedded Bootloader is intended for use with the IEEE® 802.15.4 evaluation kits (EVK). However, it will be possible to use the Embedded Bootloader to upload applications in the development phase, but it cannot provide any kind of debug facilities.

The Embedded Bootloader provides an easy and inexpensive way to upload new firmware and eliminate the requirements for expensive debug/development tools. The only requirement is a standard PC with an RS232 UART/USB interface running Windows 2000 or XP.

The Embedded Bootloader must be used with the Zigbee Flash Tool which can be found in the Test Tool Suite 'Test Tool.exe'.

This document describes Embedded Bootloader version 4.10.

Audience

This document is intended for application developers.

Organization

This document is organized into five chapters.

Chapter 1	Embedded Bootloader Description — This chapter gives an overview of the Embedded Bootloader.
Chapter 2	Using the Embedded Bootloader — This chapter describes the basic functionality of the Embedded Bootloader.
Chapter 3	Test Tool, Zigbee Flash Tool — This chapter describes the Zigbee Flash Tool. The GUI and Command Line versions are covered.
Chapter 4	Public Function Description — This chapter provides a description of the defines and functions in the Embedded Bootloader.
Chapter 5	Memory Map — This chapter describes the Bootloader Memory Map.

Conventions

This document uses the following notational conventions:

- Courier monospaced type indicates commands, command parameters, code examples, expressions, data types, and directives.
- Italic type indicates replaceable command parameters.
- All source code examples are in C.

Definitions, Acronyms, and Abbreviations

BDM	Background Debug Module
GUI	Graphical User Interface
MAC	Medium Access Control

MCU	MicroController Unit
NVM	None-Volatile Memory
PC	Personal Computer
PCB	Printed Circuit Board
S19	‘S19’ is the file extension used for the Motorola binary image format. The S19 file encapsulates the binary image as a list of ASCII records. Each record contains a length -, address -, data - and checksum field. The 16 bit address field allows a memory space for up to 64 KB. The S19 can be generated with Metroworks Codewarrior IDE and is the product from the linking process. S19 does not contain additional information to a debugger (where to look for source files).
BDM debugger	A debugger using the BDM interface for communication with the MCU. An example is the P&E BDM Multilink debugger for HCS08.
EVK board	This term covers the DIG-528-2 (EVK) and DIG536-2 (SARD) boards.
Safe Mode Boot	The Embedded Bootloader boots up using safe default system values.

References

- [1] Freescale 802.15.4 MAC/PHY Software Reference Manual, 802154MPSRM/D.
- [2] Zigbee.hlp (see Test Tool installation directory \help)
- [3] Freescale MC908HCS08GB60/GT60 MCU Data Sheet, MC9S08GB60/D
- [4] Freescale Application Note, Handling MAC Address Erasure, AN2825/D
- [5] Freescale Application Note, Zigbee/802.15.4 Evaluation Kit, Quick Start Guide, AN2772/D

Revision History

The following table summarizes revisions to this manual since the previous release (Rev. 2.2).

Revision History	
Location	Revision
Entire Document	This document's updated format reflects that the Semiconductor Products Sector of Motorola, Inc. became Freescale Semiconductor, Inc. in 2004.

Chapter 1

Embedded Bootloader Description

The Embedded Bootloader is intended for use with the IEEE® 802.15.4 evaluation kits (EVK). However, it will be possible to use the Embedded Bootloader to upload applications in the development phase, but it cannot provide any kind of debug facilities.

The Embedded Bootloader provides an easy and inexpensive way to upload new firmware and eliminate the requirements for expensive debug/development tools. The only requirement is a standard PC with an RS232 UART/USB interface running Windows 2000 or XP.

The Embedded Bootloader must be used with the Zigbee Flash Tool which can be found in the Test Tool Suite 'Test Tool.exe'.

This document describes Embedded Bootloader version 4.10.

The Embedded Bootloader is located in a protected 4 KB flash block in the highest memory area (0xF000-0xFFFF) of the Freescale MC908HCS08GB60/GT60 microcontroller. (It cannot be accidentally erased.) A BDM debugger is required to erase the Embedded Bootloader. See the *Handling MAC Address Erasure Application Note*, AN2825/D

1.1 Target

The Embedded Bootloader runs on the Freescale MC908HCS08GB60/GT60 MCU.

The MC908HCS08GB60/GT60 is a member of Freescale's low-cost, high-performance HCS08 family. The MC908HCS08GB60/GT60 has 60 KB embedded flash (flash sector size of 512 bytes) and 4 KB embedded RAM.

The Embedded Bootloader uses the MC13192 CLK0, which runs at 62.5 KHz.

1.1.1 Ported Targets

The Embedded Bootloader must be ported (I/O mapped) to a specific PCB for proper functionality and is currently ported to the following Freescale PCBs:

Axiom AXM-0308:

- PC Communication Interface: RS232/UART on COM1 (SCI0)/
RS232/UART on COM2 (SCI1)
- Safe Mode Boot Short pins 2-3 on COM1 (SCI0)/
COM2 (SCI1)
- Version Number “AX-0308 Ver 4.10”

DIG528-2 EVK

- PC Communication Interface: RS232/UART on COM1 (SCI0)/
USB (SCI1)
- Safe Mode Boot Short pins 2-3 on COM1-port (SCI0)
- Version Number “528&536 Ver 4.10”

DIG536-2 SARD

- PC Communication Interface: RS232/UART on COM1 (SCI0)
- Safe Mode Boot Short pins 2-3 on COM1-port (SCI0)
- Version Number “528&536 Ver 4.10”

1.2 Features

The following features are supported:

- Upload firmware (802.15.4/application) in Motorola S19 record format through UART/USB. The S1 data record length must be set to 32 bytes
- Initialize memory of uploaded firmware
- Initialize the system clock. Self clocked mode and MC13192 clock setup. Power save mode supported
- Run time update/change of the NVM

NOTE

See the *Freescale 802.15.4 MAC/PHY Software Reference Manual*, 802154MPSRM/D, for a detailed description on NVM layout and values.

1.2.1 Optional Firmware Upload Settings

- Skip flash erase
 - Disabled **The 802.15.4/Application/NVM (except production data section) is erased (default)**
 - Enabled **The 802.15.4/Application/NVM is NOT erased**

- Erase production data (get production data from firmware file)
 - Disabled **The production data section (with MAC address) in NVM are preserved. All other NVM values are updated with the values from the S19 record file (default)**
 - Enabled **The production data (with MAC address) and all other values in NVM are erased. All NVM values are updated with the values from the S19 record file**

NOTE

Care must be taken when enabling this option. The user must save a backup of vital production data (MAC address). The production data can be manually added to the NVM structure in the NV_Data.c file.

- Do not reset after upload
 - Disabled **The system is automatically reset after upload (default)**
 - Enabled **The system must be manually reset by user**

- Skip firmware checksum verification
 - Disabled **A checksum verification of the S19 file data record is performed ²⁾ (default).**

NOTE

The Embedded Bootloader will report the address of the first data mismatch found in the current S19 data record. System must be reset.

- Enabled **No checksum verification is performed. Flash programming errors cannot be detected.**

- The UART/USB communication link is also protected with a protocol checksum.

1.3 Benefits

- Users do not have to buy expensive third party debug/development tools to get started.
- Users can update the 802.15.4/Application firmware without having to build in additional code for interfacing to the Embedded Bootloader (See Section 2.1.3, Safe Mode Boot).
- Firmware can be updated after production by users. However, this requires that the final product has a communication interface (UART/USB).
- Auto-Detection of PC communication interface (see Section 1.1, Target).
- User can update the 802.15.4/Application firmware even when it is malfunctioning (See Section 2.1.3, Safe Mode Boot).
- The 802.15.4/Application firmware does not have to include initializing or flash programming code and can thereby minimize code size.
- Any NVM data, specified by the 802.15.4/Application, can be updated.

1.4 Potential Issues

- Uses 4 KB of flash (~6.7 % on a MC908HCS08GB60/GT60) and 272 bytes of RAM (~6.7 % on a MC908HCS08GB60/GT60)
- Extended power/boot up time (~17ms) because the Embedded Bootloader must detect the presence of an application.

Chapter 2

Using the Embedded Bootloader

This chapter describes the Embedded Bootloader functionality. A detailed description of the functions briefly mentioned in this chapter can be found in Chapter 5, Embedded Bootloader Memory Map.

2.1 Upload Firmware

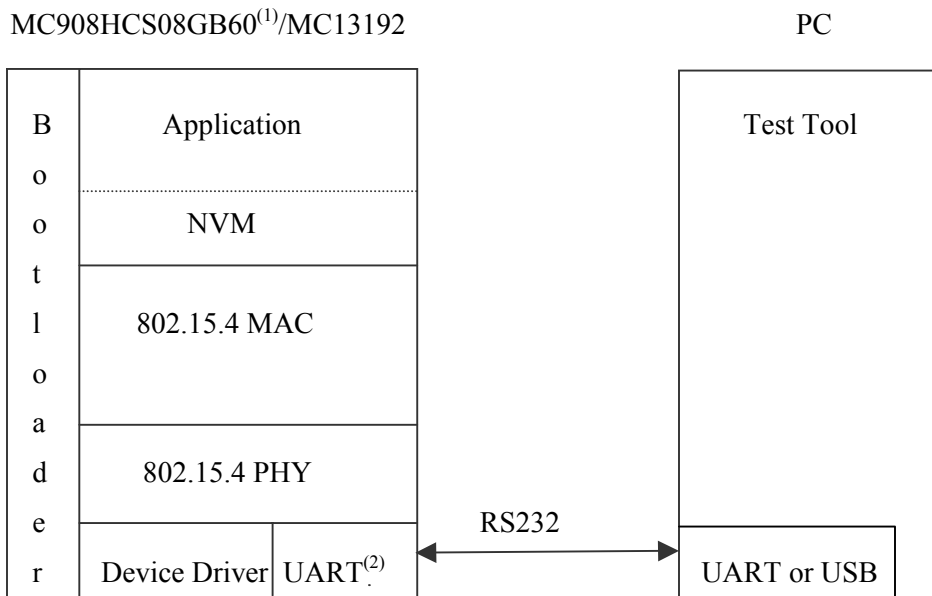
The Embedded Bootloader can be used in different system configurations depending on the PCB and the application. The application should have a user interface feature that makes it possible for the application to call the `Enable_Download_Firmware()` function:

The application can optionally call the `Hard_Reset()` function to perform a reset, or the board can be reset manually. The board will now start up in Embedded Bootloader mode.

Start the PC-Tool. See Chapter 3 for more information.

2.1.1 Use Case One

In this case, users send a specific command via the UART to enable firmware upload



¹⁾ Or MC908HCS08GT60

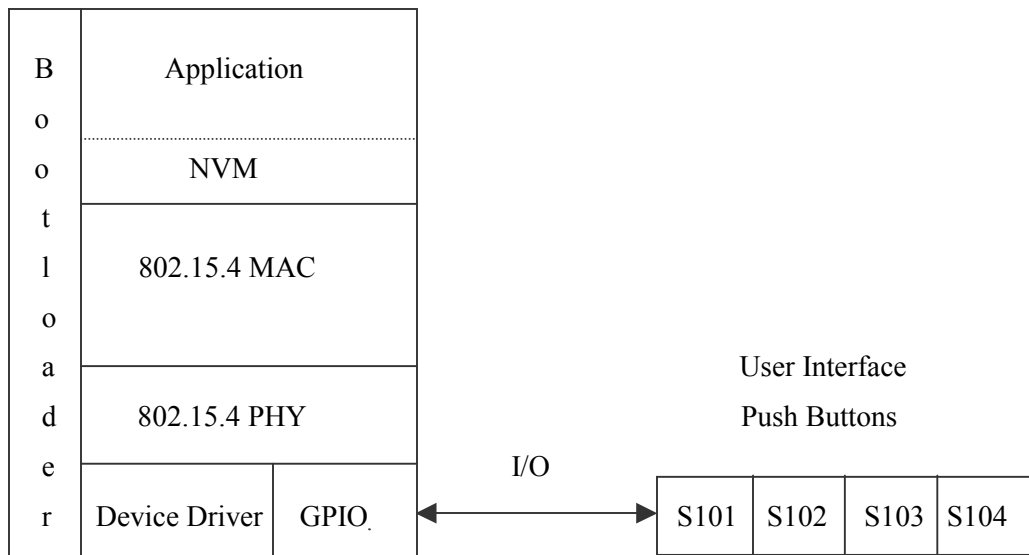
²⁾ Or USB

Figure 1. Application Supporting RS232 UART or USB Interface

2.1.2 Use Case Two

In this use case, users push a button on the board to enable firmware upload.

MC908HCS08GB60⁽¹⁾/MC13192



¹⁾ Or MC908HCS08GT60

Figure 2. Application Supporting an I/O Interface

2.1.3 Safe Mode Boot

The Safe Mode Boot Mode is a special startup mode where the Embedded Bootloader boots using safe system settings thereby resetting the system to a known state.

The Safe Mode Boot can be used to disable the detection of an invalid/malfunctioning application due to things like code errors, corrupt NVM data, or internal flash programming errors among others. All the NVM memory (except production data and MAC address) is completely erased.

The Safe Mode Boot can also be used to upload firmware without first calling the Enable_Download_Firmware() function. This could be useful if the use cases (as shown in Section 2.1, Upload Firmware) are not applicable.

Users must perform the following steps to conduct a Safe Mode Boot.

1. Power off the board
2. Disconnect RS232 UART cable (if the RS232 UART interface is used)
3. Short UART TX and RX (pin 2-3)
4. Power up again. All LEDs are off.
5. Wait until LED1 goes on (< 1 second)
6. Power off the board

7. Remove short from UART TX and RX and connect UART cable again (if the RS232 UART interface is used)
8. Power up again
9. Embedded Bootloader is ready to receive new firmware (all LEDs on)
10. Start the PC-Tool. See Chapter 3, for more information.

2.2 Updating Non-volatile Memory (NVM)

The following steps show how to update the NVM data from an application (code).

1. The Embedded Bootloader must be present on the EVK board. All EVK boards are shipped with Embedded Bootloader pre-programmed in flash.

NOTE

The Embedded Bootloader can only be erased/programmed with a BDM debugger.

2. Call the `Update_NV_RAM()` function. This function can change any NVM data.

2.2.1 An Example of How to Change the MAC Address

The following code shows an example of how to change the MAC address.

```
Update_NV_RAM(&(NV_RAM_ptr->MAC_Address)[0], &pPacket[DATA_INDEX], 8);
```

NOTE

`pPacket` – contains the new MAC address.

Any NVM data can in code be read as a normal construct. For example, use the `NV_RAM_ptr` to get access to individual data.

2.3 Changing The System Bus Frequency

The MC908HCS08GB60/GT60 starts in 4 MHz self clocked mode. The init code changes this to 8 MHz after a few instructions from reset.

If NVM data is found the system clock (MC908HCS08GB60/GT60 ICG module and MC13192 CLKO), UART baud rate and other options are setup as specified by the downloaded application. The baud rate depends on the NVM values specified by the application. See the *Freescale 802.15.4 MAC/PHY Software Reference Manual*, 802154MPSRM/D, for more details.

If no NVM data can be found, the following (safe mode boot) values are used:

- MC13192 CLKO = 62.5 KHz
- MC908HCS08GB60/GT60 bus clock = 16 MHz
- UART baud rate 19200 kbps, 8 data, 1 start, 1 stop, none parity.



Chapter 3

Test Tool – Zigbee Flash Tool

The Zigbee flash tool is a part of the general Zigbee Test Tool. This chapter provides a brief description of how to use the Zigbee Test Tool to upload new firmware. For more details about installation and other features, see the documentation for the Zigbee Test Tool and the Zigbee.hlp file in Test Tool installation directory \help.

The flash programming part of the Test Tool can be used with two different user interfaces.

1. The GUI-version in 'Test Tool.exe'
2. The command line version in 'Bootloader.exe' in the 'S19' folder.

This description covers version 4.10 of the Test Tool.

Uploadable applications in Motorola S19 file format must be copied to the \Freescale\Test Tool\S19 directory in advance. Copy any new applications in S19 format to this folder.

NOTE

Look for new or updated applications at the Freescale Zigbee/802.15.4 web page.

3.1 Graphical User Interface (GUI) Version

To use the GUI version of the Test Tool, execute the following file:

```
.\Freescale\Test Tool\“Test Tool.exe”
```

3.1.1 Using the Tools Menu Option

After clicking on the Tools menu option, click the Communication Settings option to choose the baud rate specified for the current embedded application.

Notice that applications require that you push one or more buttons or some other functions to enable upload of new firmware. See the *ZigBee/802.15.4 Evaluation Kit Quick Start Guide, AN2772/D*, for a description of what to do for a specific application.

If no application is downloaded, use the default settings specified for the Embedded Bootloader. See Section Chapter 2 for more information.

Use the Add... buttons. Click “Close”.

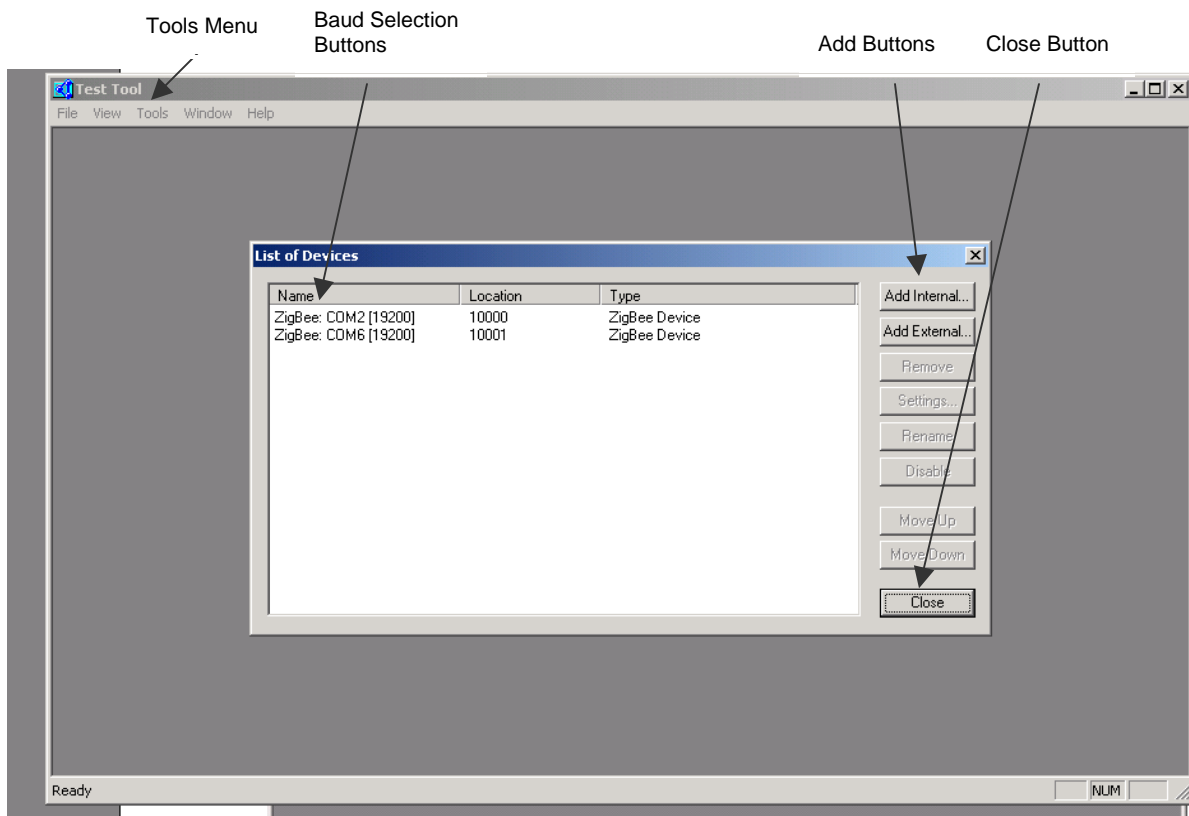


Figure 3. Tools Menu Selection

3.1.2 Using the View Menu Option

After clicking the View menu option, select the Zigbee ‘Embedded Bootloader’ menu option. Then select port COMx and click “OK”. Choose the baud rate specified for the current embedded application. See the *Zigbee/802.15.4 Evaluation Kit Quick Start Guide, AN2772/D*, for a description of what to do for a specific application.

If no application is uploaded, you must use the default settings specified for the Embedded Bootloader. See Section Chapter 2 for more information.

NOTE

If the USB interface is used, the USB option appears as a new COM port.

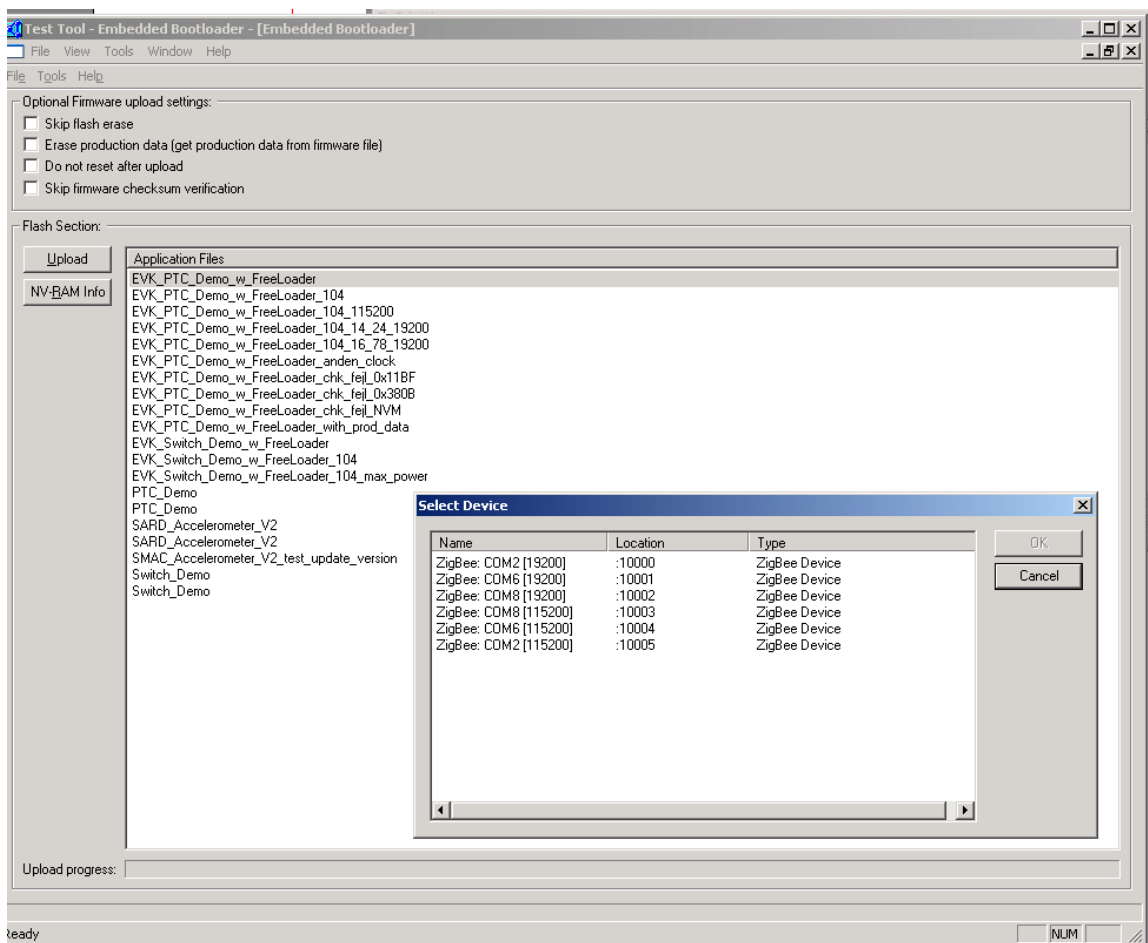


Figure 4. View Menu Option

3.1.3 Selecting the Firmware File to Download

In the 'Flash Section' window, click on the application file to be uploaded and then click on the "Upload" button. The upload progress bar begins to indicate upload progress.

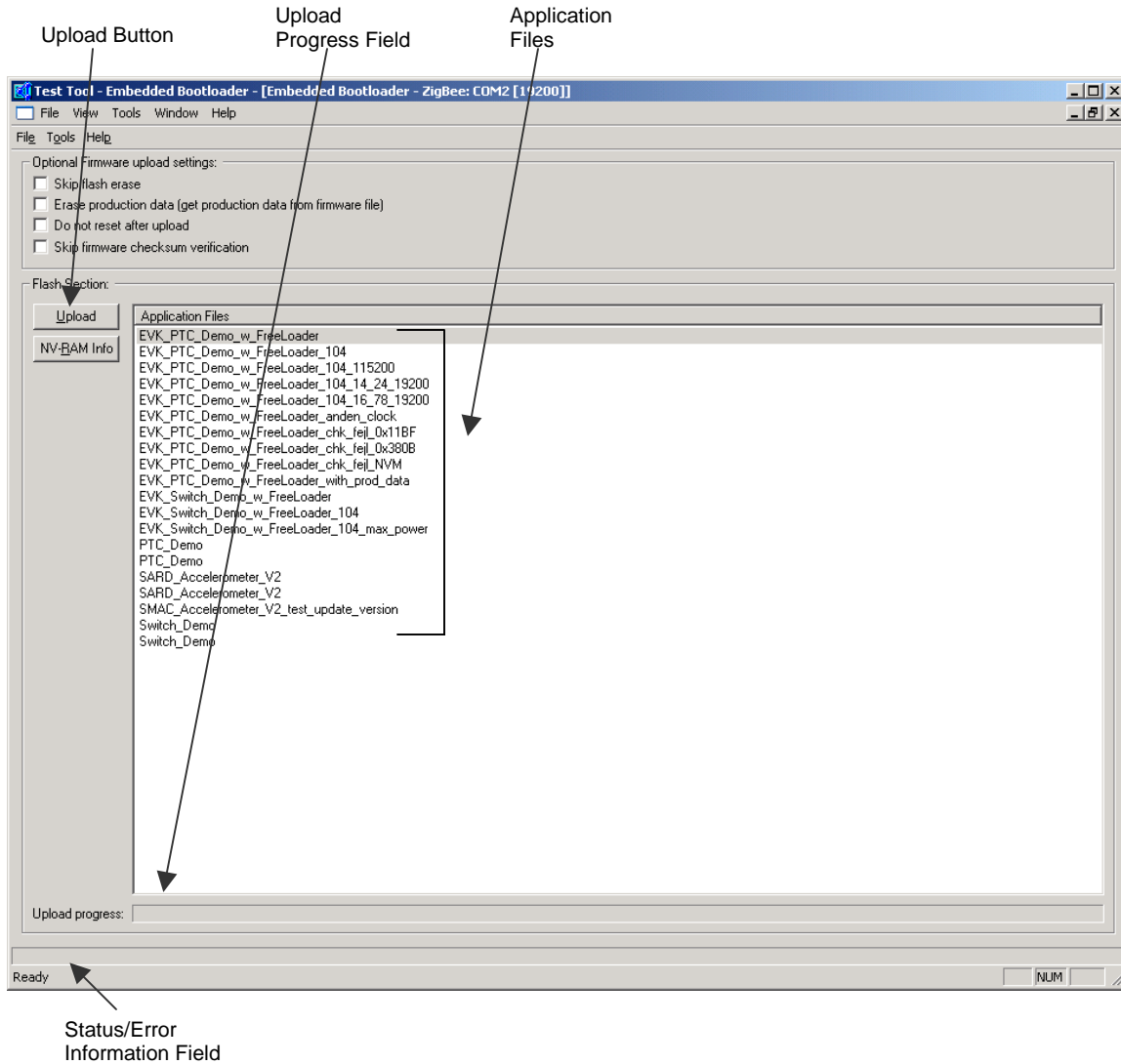


Figure 5. Firmware Download

The Status/Error Information Field shows the Status/Error Information received from the Embedded Bootloader.

After a successful upload of the firmware, the window appears as shown in Figure 6.

NOTE

A system reset is performed (default) after upload as shown in Figure 6.

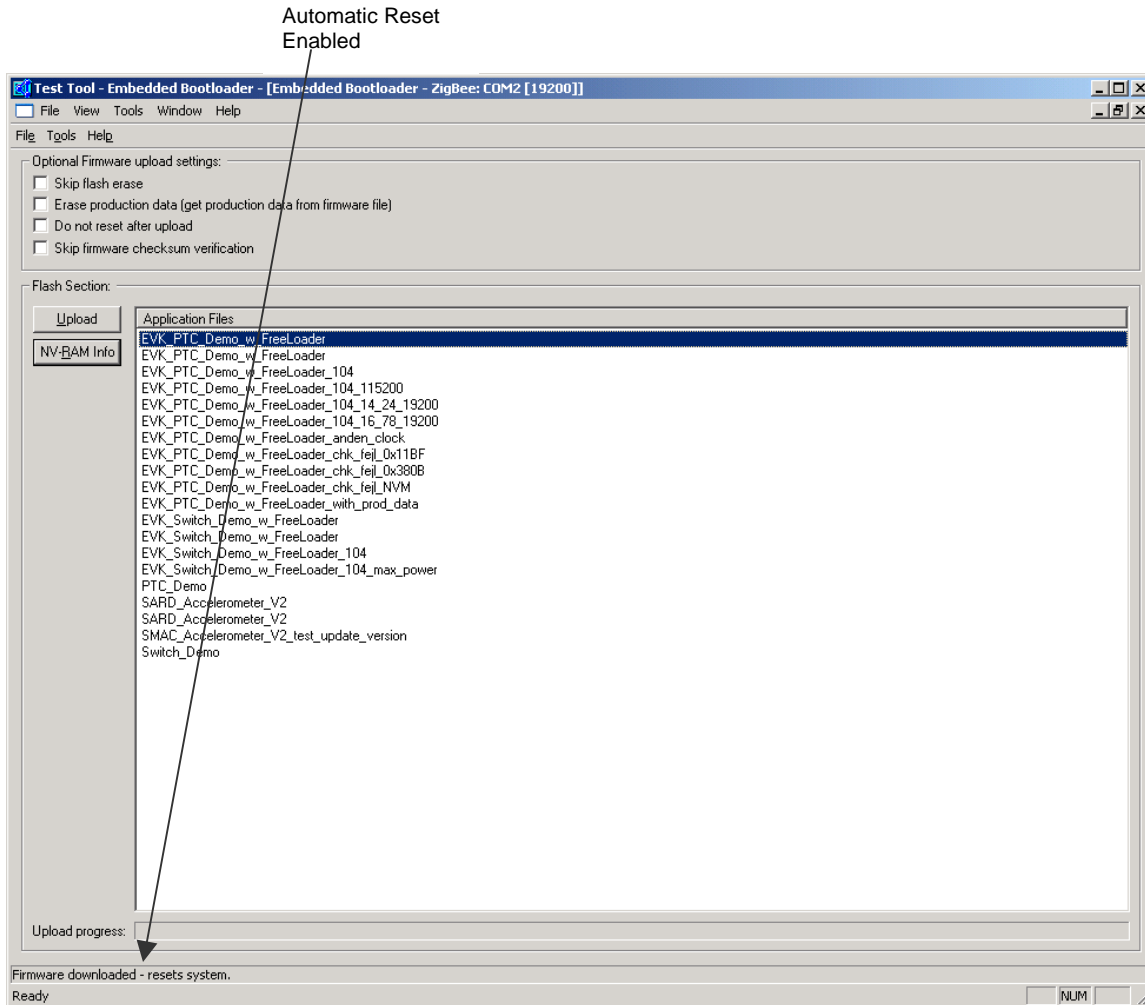


Figure 6. After a Successful Firmware Upload

NOTE

If the uploaded application uses another baud rate, it is required to change baud rate values in order to be able to communicate with the board.

3.1.4 Using the Help Menu

Click on Help, then Click on the About menu option to see Test Tool version number.

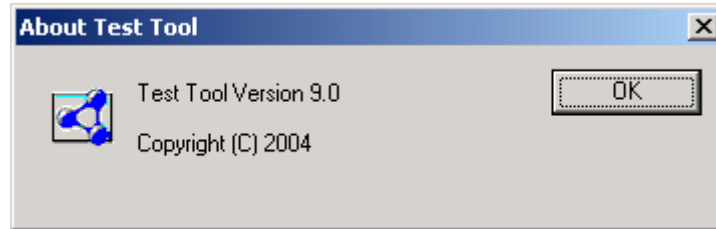


Figure 7. Test Tool Version

3.1.5 Changing the NVM Data in Flash or File.

If you click on an application in the list, it displays the NVM values in the selected application file. If the uploaded application supports NVM dump, it is possible to read the NVM from the board.

The NVM data can be changed in two different ways:

1. If the uploaded application does not support NVM data editing it is possible to edit the NVM data in the application file before it is uploaded to the board and save it back in the application file for later upload.
2. If the uploaded application does support NVM data editing it is possible to edit the NVM data after the board has been uploaded with the application. For example, the file contains default values. The Freescale EVK PTC application supports this feature.

NOTE

No production specific information is available in this example window.

Use caution when editing the clock related configuration values. The values must match each other. There is no sanity check on the user entered values, though there a check on length. See the MC908HCS08GB60/GT60 MCU Data Sheet, MC9S08GB60/D about how to specify valid register values for the HCS08.

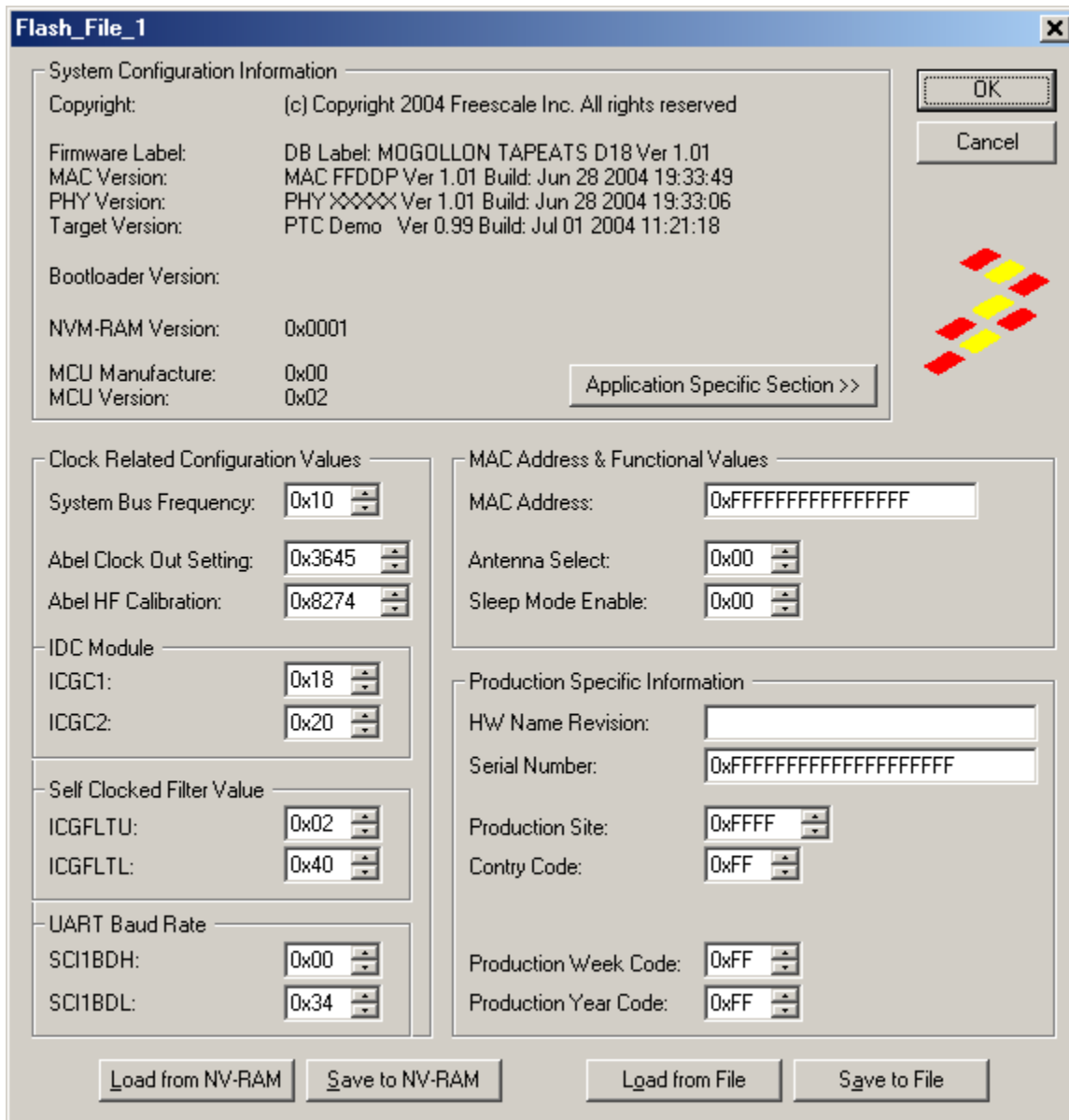
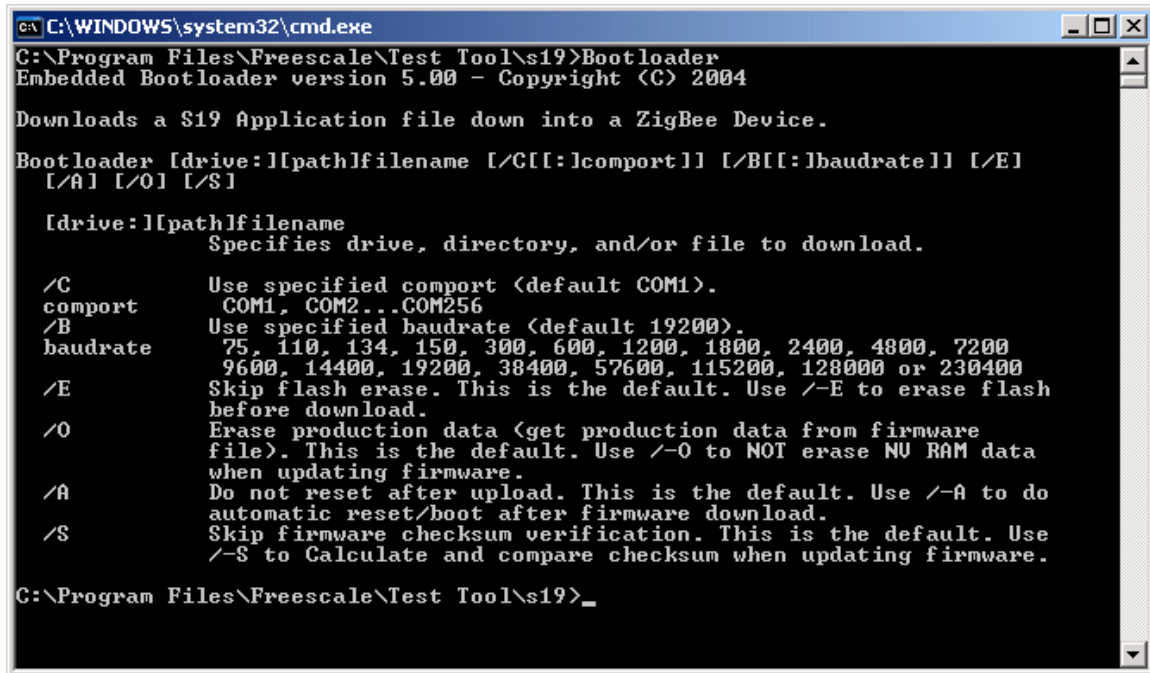


Figure 8. System Configuration Window

3.2 Command Line Version (CMD)

The command line version of the Zigbee Flash Tool must be called with parameters:

Execute 'Bootloader.exe' without parameters to show a parameter list:



```
C:\WINDOWS\system32\cmd.exe
C:\Program Files\Freescale\Test Tool\s19>Bootloader
Embedded Bootloader version 5.00 - Copyright (C) 2004

Downloads a S19 Application file down into a ZigBee Device.

Bootloader [drive:][path]filename [/C[:]comport] [/B[:]baudrate] [/E]
[/A] [/O] [/S]

[drive:][path]filename
    Specifies drive, directory, and/or file to download.

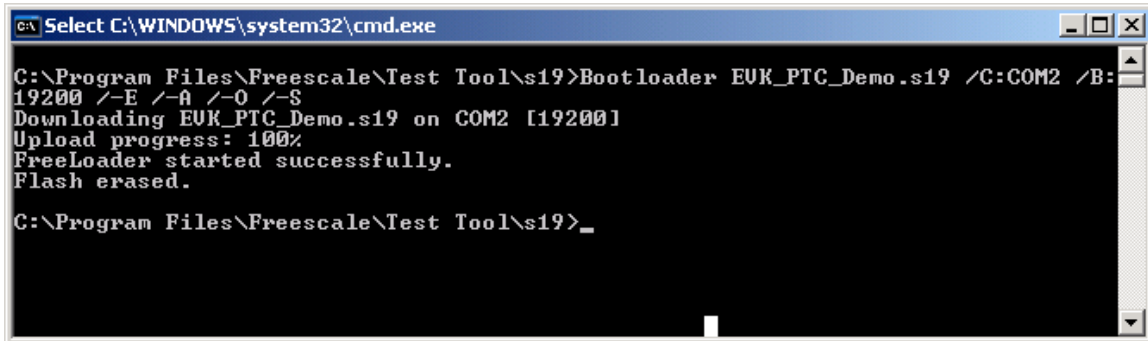
/C      Use specified comport (default COM1).
comport COM1, COM2...COM256
/B      Use specified baudrate (default 19200).
baudrate 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200
          9600, 14400, 19200, 38400, 57600, 115200, 128000 or 230400
/E      Skip flash erase. This is the default. Use /-E to erase flash
before download.
/O      Erase production data (get production data from firmware
file). This is the default. Use /-O to NOT erase NU RAM data
when updating firmware.
/A      Do not reset after upload. This is the default. Use /-A to do
automatic reset/boot after firmware download.
/S      Skip firmware checksum verification. This is the default. Use
/-S to Calculate and compare checksum when updating firmware.

C:\Program Files\Freescale\Test Tool\s19>_
```

Figure 9. Command Line Version (Parameter List)

3.2.1 Normal Use (Default) Example

”Bootloader EVK_PTC_Demo.s19 /C:COM2 /B:19200 /-E /-A /-O /-S”



```

c:\Select C:\WINDOWS\system32\cmd.exe
C:\Program Files\Freescale\Test Tool\s19>Bootloader EVK_PTC_Demo.s19 /C:COM2 /B:
19200 /-E /-A /-O /-S
Downloading EVK_PTC_Demo.s19 on COM2 [19200]
Upload progress: 100%
FreeLoader started successfully.
Flash erased.
C:\Program Files\Freescale\Test Tool\s19>_

```

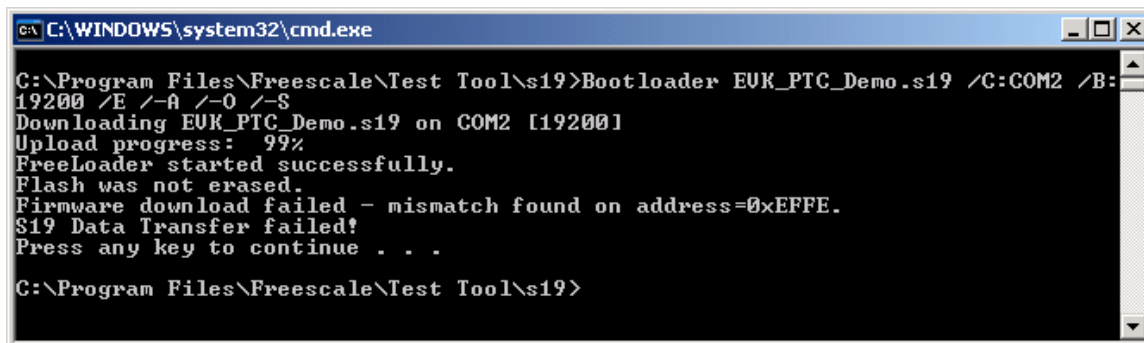
Figure 10. Command Line Version (Using Several Options)

NOTE

It is not possible to edit NVM data with the CMD version.

3.2.2 Flash Erase Disabled Example

”Bootloader EVK_PTC_Demo.s19 /C:COM2 /B:19200 /E /-A /-O /-S”



```

c:\C:\WINDOWS\system32\cmd.exe
C:\Program Files\Freescale\Test Tool\s19>Bootloader EVK_PTC_Demo.s19 /C:COM2 /B:
19200 /E /-A /-O /-S
Downloading EVK_PTC_Demo.s19 on COM2 [19200]
Upload progress: 99%
FreeLoader started successfully.
Flash was not erased.
Firmware download failed - mismatch found on address=0xEF FE.
$19 Data Transfer failed!
Press any key to continue . . .
C:\Program Files\Freescale\Test Tool\s19>

```

Figure 11. Command Line Version (Flash Erase Disable Option)

NOTE

The flash is not erased and a flash programming error is detected at address 0xEF FE. This is the address of the Embedded Bootloader control flags, which was not erased. The error message is expected.



Chapter 4

Public Function Description

The 802.15.4/application code must be linked with the following linker file in order to work with the Embedded Bootloader interface:

```
FreeLoader_HCS08GB60.ach
```

NOTE

This file must also be used with the MC908HCS08GT60 MCU.

4.1 Compiler Defines Application

The following compiler #define must be specified to enable Embedded Bootloader functionality in an 802.14.4/application:

```
#define BOOTLOADER_ENABLED
```

4.2 Compiler Defines Embedded Bootloader

The following compiler #define must be specified to enable the Embedded Bootloader functionality in the Embedded Bootloader application:

```
#define FOR_BOOTLOADER_ONLY  
#define BOOTLOADER_ENABLED
```

4.3 Embedded Bootloader Support Source Code

The 802.14.4/application must embed some support source code to make an interface to the Embedded Bootloader application. The needed files are delivered to customers who want to embed the Embedded Bootloader in their system.

4.4 DigiType.h

This is a C type define for the types described in the following sections.

4.4.1 Gb60_io.h

This is a standard MC908HCS08GB60 MCU interface file. All peripherals embedded in the MCU are listed with their absolute addresses.

NOTE

This file must also be used with the MC908HCS08GT60 MCU.

4.4.2 Crt0.c and Crt.h

These files contain the basic init code (basic system clock, memory and stack setup). The normal basic init file(s) (like the start08.c from Metrowerks) is not needed when the Embedded Bootloader is embedded. The Embedded Bootloader handles all the basic initialization. In an 802.14.4/application with Embedded Bootloader, these files only contains the `_startupdata` structure.

4.4.3 FreeLoader_inf.c and FreeLoader_inf.h

These are the interface files for the 802.14.4/application. They contain function pointers to function accessible in the Embedded Bootloader. These files also contain absolute addresses, which should not be changed.

4.4.4 NV_Data.c and NV_Data.h

The NVM is not a part of the Embedded Bootloader. However, it is advised to make the NVM a part of the 802.14.4/application.

The Embedded Bootloader can use the information in NV memory, but it has default values (safe mode boot) to cover scenarios where no NV memory is available.

4.4.5 ISR Vector Table Location

The Embedded Bootloader redirects the ISR vector table to 0x0EFC0, i.e. the 802.14.4/application ISR vector table must be locate at 0x0EFC0 to 0xEFFD.

4.4.6 Reset Vector Location

The reset vector is in the Embedded Bootloader address space (0xFFFE). The Embedded Bootloader has 2 system variables located where the redirected “reset vector” is locate at 0xEFFE to 0xEFFF. The variables control the startup of the Embedded Bootloader.

4.4.7 System Clock Setup

The Embedded Bootloader contains a function `ICG_Setup()` which is called at power up. Applications can access this function by calling the `FL_ICG_Setup()` function. The Embedded Bootloader will setup the system clock as specified in the NVM section (ICG) if available. If not a default value is used. Thereby can the application save code space for implementing system clock setup.

The 802.14.4/application can also have an “lost clock lock” ISR function which calls the `FL_ICG_Setup()` function. The function must be added to the ISR vector table.

Here an example of an ISR function to handle lost lock of clock (CLKO):

```
__interrupt void FLL_Lost_Lock_ISR(void)
{
    // Setup ICG module again to prevent that system hangs forever.
    ICGS1 |= 0x01; // Clear FLL lost lock interrupt
    #if defined BOOTLOADER_ENABLED
    FL_ICG_Setup(); // Call ICG_Setup() in Embedded Bootloader
    #endif defined BOOTLOADER_ENABLED
}
```

4.5 802.15.4 Accessible Functions

The following functions located in the Embedded Bootloader, are made accessible for 802.15.4/applications. For example, an 802.15.4/application is not required to have programming routines to store data in flash. A function pointer, for each function of the below listed functions, is defined in Embedded Bootloader_Inf.c:

```
Enable_Download_Firmware()
Hard_Reset()
NV_Flash_Setup()
Update_NV_RAM()
FL_ICG_Setup()
```

The Freeloader_Inf.h file contains prototypes and must be included in the source file that calls these functions. They point to the function in the Embedded Bootloader with the same name:

```
extern Enable_Download_Firmware_ptr_t Enable_Download_Firmware;
extern Hard_Reset_ptr_t Hard_Reset;
extern NV_Flash_Setup_ptr_t NV_Flash_Setup;
extern Update_NV_RAM_ptr_t Update_NV_RAM;
extern ICG_Setup_ptr_t FL_ICG_Setup;
```

4.6 Enable_Download_Firmware

Prototype:

```
bool_t Enable_Download_Firmware
(
    uint8_t interface_state,
    uint8_t firmware_state
)
```

Description:

The application must call this function to make the system ready for a new firmware download. The application must provide a way for the user to interact with the system to call this function. Please note!

All bits are enabled by default. Bits can only be disabled (erased in flash) – never enabled. The bits are enabled again by the new firmware.

Input:

interface_state - The user selectable options. The below values should be OR'ed together:

Embedded Bootloader control flags:

```
#define NO_BIT_ENABLED                ((uint8_t)0x00)

// Option: "Skip flash erase"
#define ERASE_FLASH                   ((uint8_t)0x01)

// Option: "Do not reset after upload"
#define BOOT_AFTER_DOWNLOAD           ((uint8_t)0x02)

// Option: "Erase production data"
#define KEEP_NV_RAM                   ((uint8_t)0x04)

// Option: "Skip firmware checksum verification"
#define PERFORM_FLASH_VERIFICATION    ((uint8_t)0x08)
```

NOTE

The Embedded Bootloader control flags correspond to the “optional firmware upload settings” which can be disabled/enabled in the Zigbee Flash Tool.(see Section, 1.2.1, Optional Firmware Upload Settings.

The value is inverted. To enable the bit mask option, the setting must be set to 0. To disable the bit mask option, the setting must be set to 1.

Use firmware_state – control if Embedded Bootloader or the application must be started.

To enable download, the setting must be = DO_UPDATE_FIRMWARE.

Embedded Bootloader boot flag:

```
#define EXECUTE_APPLICATION           ((uint8_t)0x55)
#define DO_UPDATE_FIRMWARE            ((uint8_t)0x00)
#define FLASH_EMPTY                   ((uint8_t)0xFF)
```

Output:

True - Ready for reset.

False - Something went wrong in changing the state.

4.7 Hard_Reset

Prototype:

```
void Hard_Reset(void)
```

Description:

The application can call this function to make a system reset. The reset is done by executing an illegal instruction.

Input:

None

Output:

None

4.8 Update_NV_RAM

Prototype:

```
bool_t Update_NV_RAM  
(  
    NV_RAM_Struct_t const *NV_RAM_Distination_ptr,  
    uint8_t *Source_ptr,  
    uint16_t Source_Length  
)
```

Description:

The application can call this function to update any NVM parameter with new values specified in the input parameters. There is no validation of input parameters.

NOTE

NVM can in code be read as a normal construct.

Input:

NV_RAM_Distination_ptr A pointer to current NV RAM data (ex. The MAC address), which must be changed.

Source_ptr A pointer to new NVM data, which must be stored.

Source_Length The length (number of bytes) of the new NVM data to store.

Output:

True – NVM data stored.

False - Something went wrong (should never happen).

4.9 NV_Flash_Setup

Prototype:

```
void NV_Flash_Setup(void)
```

Description:

This function should never be called under normal conditions. However, it should be called if any of the other Embedded Bootloader functions malfunction. The function sets up the flash functions (again), i.e. copies the flash routines to RAM for execution and initialize the HCS08 flash module.

Input:

None

Output

None

4.10 FL_ICG_Setup

Prototype:

```
void FL_ICG_Setup(void)
```

Description:

This function should never be called directly by the application. The function could be automatically called if the external system clock is unstable/removed (from ISR function) and on power down/up (doze). This requires that the FL_ICG_Setup is called from an ISR function.

Input:

None

Output

None

Chapter 5

Embedded Bootloader Memory Map

Table 1. Zigbee (MC9S08GB60/GT60) 802.15.4 Embedded Bootloader Memory Map

512 Bytes in a physical flash sector

Sector Number	Address in hex start	Address in hex end (sector erase address)	Size in bytes	General HCS08 Map and Usage	Embedded Bootloader Map
<i>NM</i>	<i>0</i>	<i>7F</i>	<i>128</i>	<i>Direct Port Registers</i>	<i>Direct Port Registers</i>
NM	80	FE	127	Direct Addressing RAM "Fast memory"	Direct Addressing RAM "Fast memory"
NM	FF	FF	1	Direct Addressing RAM "Fast memory"	Sleep variable: gSeqPowerSaveMode
NM	100	1FF	256	RAM	Embedded Bootloader stack
NM	200	F5F	3424	RAM	
NM	F60	F6F	16	RAM	Unint RAM for init structure
NM	F70	F71	2	RAM	NV_RAM_ptr
NM	F72	F7F	14	RAM	Flash routines data
NM	F80	106F	240	RAM	Flash routines critical code
NM	1070	1077	8	RAM	Static variables
NM	1078	107F	8	RAM	Static no init variables
8	1080	10A2	35	FLASH (section 1)	
8	10A3	11FF	349	FLASH (section 1)	
9	1200	13FF	512	FLASH (section 1)	
10	1400	15FF	512	FLASH (section 1)	802.15.4/App. NV RAM block 0 (share)
11	1600	17FF	512	FLASH (section 1)	802.15.4/App. NV RAM block 1 (share)

12	1800	182B	44	High Page Registers (COP, Flash etc.)	High Page Registers (COP, Flash etc.)
12	182C	19FF	468	FLASH (section 2)	
13	1A00	1BFF	512	FLASH (section 2)	
14	1C00	1DFF	512	FLASH (section 2)	
15	1E00	1FFF	512	FLASH (section 2)	
16	2000	21FF	512	FLASH (section 2)	
17	2200	23FF	512	FLASH (section 2)	
18	2400	25FF	512	FLASH (section 2)	
19	2600	27FF	512	FLASH (section 2)	
20	2800	29FF	512	FLASH (section 2)	
21	2A00	2BFF	512	FLASH (section 2)	
22	2C00	2DFF	512	FLASH (section 2)	
23	2E00	2FFF	512	FLASH (section 2)	
24	3000	31FF	512	FLASH (section 2)	
25	3200	33FF	512	FLASH (section 2)	
26	3400	35FF	512	FLASH (section 2)	
27	3600	37FF	512	FLASH (section 2)	
28	3800	39FF	512	FLASH (section 2)	
29	3A00	3BFF	512	FLASH (section 2)	
30	3C00	3DFF	512	FLASH (section 2)	
31	3E00	3FFF	512	FLASH (section 2)	
32	4000	41FF	512	FLASH (section 2)	
33	4200	43FF	512	FLASH (section 2)	
34	4400	45FF	512	FLASH (section 2)	
35	4600	47FF	512	FLASH (section 2)	
36	4800	49FF	512	FLASH (section 2)	
37	4A00	4BFF	512	FLASH (section 2)	

38	4C00	4DFF	512	FLASH (section 2)
39	4E00	4FFF	512	FLASH (section 2)
40	5000	51FF	512	FLASH (section 2)
41	5200	53FF	512	FLASH (section 2)
42	5400	55FF	512	FLASH (section 2)
43	5600	57FF	512	FLASH (section 2)
44	5800	59FF	512	FLASH (section 2)
45	5A00	5BFF	512	FLASH (section 2)
46	5C00	5DFF	512	FLASH (section 2)
47	5E00	5FFF	512	FLASH (section 2)
48	6000	61FF	512	FLASH (section 2)
49	6200	63FF	512	FLASH (section 2)
50	6400	65FF	512	FLASH (section 2)
51	6600	67FF	512	FLASH (section 2)
52	6800	69FF	512	FLASH (section 2)
53	6A00	6BFF	512	FLASH (section 2)
54	6C00	6DFF	512	FLASH (section 2)
55	6E00	6FFF	512	FLASH (section 2)
56	7000	71FF	512	FLASH (section 2)
57	7200	73FF	512	FLASH (section 2)
58	7400	75FF	512	FLASH (section 2)
59	7600	77FF	512	FLASH (section 2)
60	7800	79FF	512	FLASH (section 2)
61	7A00	7BFF	512	FLASH (section 2)
62	7C00	7DFF	512	FLASH (section 2)
63	7E00	7FFF	512	FLASH (section 2)
64	8000	81FF	512	FLASH (section 2)

65	8200	83FF	512	FLASH (section 2)
66	8400	85FF	512	FLASH (section 2)
67	8600	87FF	512	FLASH (section 2)
68	8800	89FF	512	FLASH (section 2)
69	8A00	8BFF	512	FLASH (section 2)
70	8C00	8DFF	512	FLASH (section 2)
71	8E00	8FFF	512	FLASH (section 2)
72	9000	91FF	512	FLASH (section 2)
73	9200	93FF	512	FLASH (section 2)
74	9400	95FF	512	FLASH (section 2)
75	9600	97FF	512	FLASH (section 2)
76	9800	99FF	512	FLASH (section 2)
77	9A00	9BFF	512	FLASH (section 2)
78	9C00	9DFF	512	FLASH (section 2)
79	9E00	9FFF	512	FLASH (section 2)
80	A000	A1FF	512	FLASH (section 2)
81	A200	A3FF	512	FLASH (section 2)
82	A400	A5FF	512	FLASH (section 2)
83	A600	A7FF	512	FLASH (section 2)
84	A800	A9FF	512	FLASH (section 2)
85	AA00	ABFF	512	FLASH (section 2)
86	AC00	ADFF	512	FLASH (section 2)
87	AE00	AFFF	512	FLASH (section 2)
88	B000	B1FF	512	FLASH (section 2)
89	B200	B3FF	512	FLASH (section 2)
90	B400	B5FF	512	FLASH (section 2)
91	B600	B7FF	512	FLASH (section 2)


92	B800	B9FF	512	FLASH (section 2)
93	BA00	BBFF	512	FLASH (section 2)
94	BC00	BDFF	512	FLASH (section 2)
95	BE00	BFFF	512	FLASH (section 2)
96	C000	C1FF	512	FLASH (section 2)
97	C200	C3FF	512	FLASH (section 2)
98	C400	C5FF	512	FLASH (section 2)
99	C600	C7FF	512	FLASH (section 2)
100	C800	C9FF	512	FLASH (section 2)
101	CA00	CBFF	512	FLASH (section 2)
102	CC00	CDFF	512	FLASH (section 2)
103	CE00	CFFF	512	FLASH (section 2)
104	D000	D1FF	512	FLASH (section 2)
105	D200	D3FF	512	FLASH (section 2)
106	D400	D5FF	512	FLASH (section 2)
107	D600	D7FF	512	FLASH (section 2)
108	D800	D9FF	512	FLASH (section 2)
109	DA00	DBFF	512	FLASH (section 2)
110	DC00	DDFF	512	FLASH (section 2)
111	DE00	DFFF	512	FLASH (section 2)
112	E000	E1FF	512	FLASH (section 2)
113	E200	E3FF	512	FLASH (section 2)
114	E400	E5FF	512	FLASH (section 2)
115	E600	E7FF	512	FLASH (section 2)
116	E800	E9FF	512	FLASH (section 2)
117	EA00	EBFF	512	FLASH (section 2)
118	EC00	EDFF	512	FLASH (section 2)


119	EE00	EFBF	448	FLASH (section 2)	
119	EFC0	EFFD	62	FLASH (section 2)	
119	EFFE	FFFF	2	FLASH (section 2)	
120	F000	F00F	16	FLASH (section 2)	Embedded Bootloader function ptrs (8 pieces)
120	F010	F013	4	FLASH (section 2)	Illegal opcode instruction
120	F014	F103	240	FLASH (section 2)	Flash routines critical code (copy to RAM)
120	F104	F1FF	252	FLASH (section 2)	
121	F200	F3FF	512	FLASH (section 2)	
122	F400	F5FF	512	FLASH (section 2)	
123	F600	F7FF	512	FLASH (section 2)	
124	F800	F9FF	512	FLASH (section 2)	
125	FA00	FBFF	512	FLASH (section 2)	
126	FC00	FDFE	512	FLASH (section 2)	
127	FE00	FFAF	432	FLASH (section 2)	
127	FFB0	FFBF	16	NV Registers	NV Registers
127	FFC0	FFFD	62	ISR vectors (31 vectors - 25 implemented)	ISR vectors (31 vectors - 25 implemented)
127	FFFE	FFFF	2	Reset vector "address in Bootloader"	Reset vector "address in Bootloader"

65535

FFFF Must be = FFFF for valid memory map.

NM = No meaning

 = May be used by Embedded Bootloader. 802.15.4/App. can reuse this space (overwrite).

 = Used by Embedded Bootloader. Data/code which must be located on the specified address and (must/will) exists when the application is running



= May be used by 802.15.4/App.

= Used by 802.15.4/App. Data/code which must be located on the specified address

If either the Embedded Bootloader and/or the 802.15.4/App. must not use a particular resource the color from the "General HCS08 Map and Usage" is kept.

Cursive = fixed register and vectors in flash

Bold = MUST be located on this particular address - DO NOT CHANGE.

Flash sector 120-127 is block protected and cannot be erased by SW.

Embedded Bootloader Application

RAM	4096	bytes available
Code	4016	bytes available (not including ISR vectors and reset vector)
NV-RAM	1024	bytes available. Read and written by Embedded Bootloader (2 sectors are used, so it occupies 1024 bytes flash).

