Freescale Semiconductor
Application Note

Document Number: AN3561
Rev. 0, 11/2007

USB Bootloader for the MC9S08JM60

by: Patrick Yang
Asia & Pacific Operation Microcontroller Division

1 Introduction

This application note describes a bootloader for the MC9S08JM60 allowing in-circuit reprogramming of the flash memory via a universal serial bus (USB).

In-circuit programming (ICP) is a process where the MCU is programmed or erased on the printed circuit board which is the target system. This allows the user code to be changed during product development, production, and code upgrades.

The MC9S08JM60 provided by Freescale Semiconductor is a member of the low-cost, high performance HCS08 family of 8-bit microcontroller units (MCUs). It has a 60 KB embedded flash memory that can be programmed or erased without special voltage input. In-circuit programming is possible through several communication paths. The MC9S08JM60 has a USB 2.0 full-speed module, that makes this MCU suitable for in-circuit programming via a USB interface. The USB speed is fast and can program 60 KB flash in 2-3 seconds, faster than a BDM cable.

Contents

freescale™ semiconductor

The flash based bootloader code can be divided into three functional groups:

- A USB low level driver to enumerate and transfer information between the board and PC
- Command interpretation and .s19 information data download
- Flash program and erase operations

Figure 1 shows the system environment for a USB bootloader.

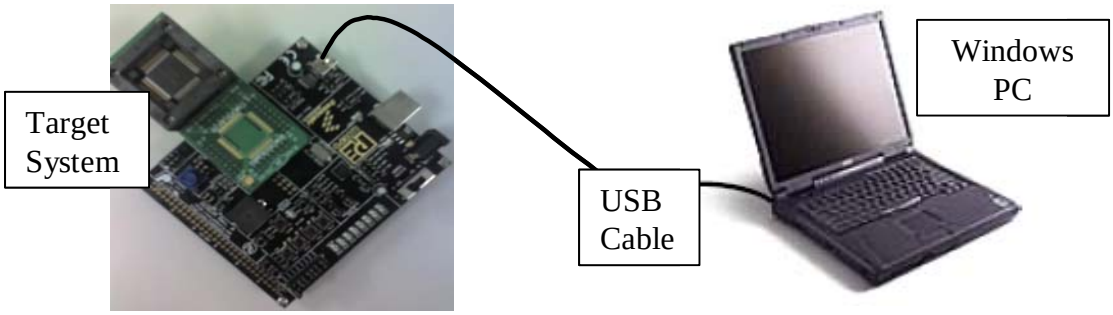

Figure 1. System Environment

The PC and target system communicate via a USB interface with Freescale Semiconductor specific USB protocol. The target system acts as a USB device and the PC as a USB host. The PC programs the target system described in Section 2.5, “PC Driver and PC GUI Tool.”

2 Bootloader Overview

2.1 Resources for the Bootloader

The bootloader uses as few MCU resources as possible to minimize the impact in an application.

- USB module — The USB interrupt is not used in the bootloader. Only the control transfer endpoint is used (endpoint 0).
- Memory footprint — The bootloader makes effecient use of memory. The code is less than 1 KB and uses only 70 bytes of RAM. The RAM has 11 bytes of bootloader variables and 59 bytes are used for flash programming/erasing. The command for programing/erasing flash must be executed from the RAM.

Figure 2 shows the memory map of the MC9S08JM60 and the memory footprint of the bootloader. Bootloader variables are located in address 0x00B0 to 0x00BA. The bootloader code resides in the end address of the flash memory, 0xFC00 to 0xFFAF.

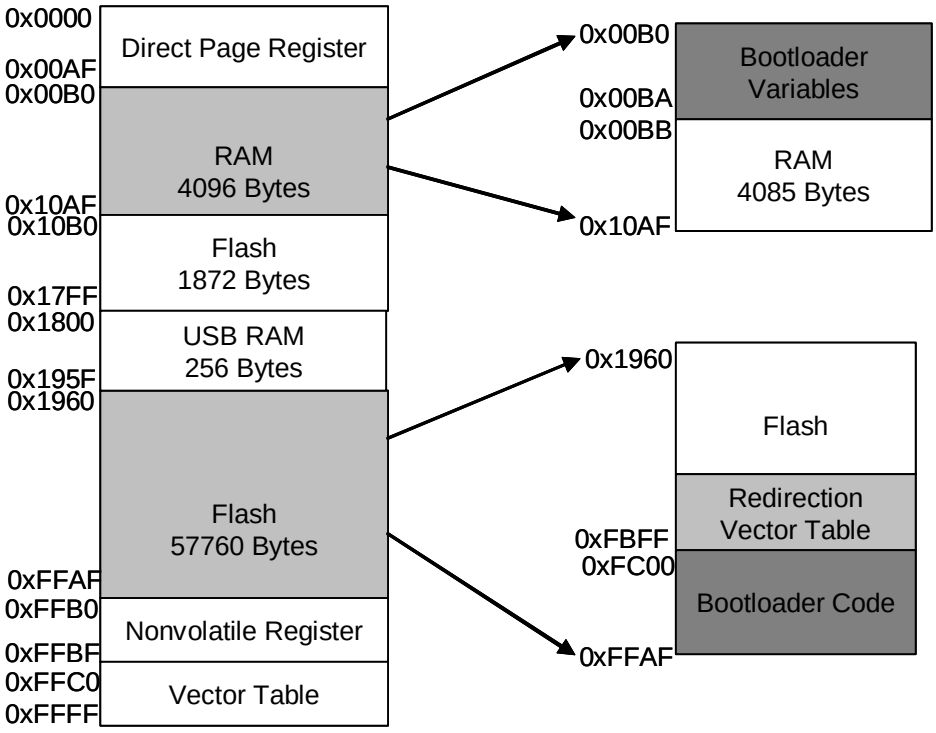

Figure 2. Memory Map

2.2 Flash Memory Protection

The flash memory (address 0xFC00–0xFFAF) must be protected to avoid accidentally erasing the bootloader code. The flash protection register controls the flash memory protection of the MC9S08JM60. Block protection begins at 512 bytes boundary below the last address of the flash memory, if the protection is enabled. For example, the flash memory with the address 0xFC00 through 0xFFFF is protected if the value 0xFA is set to the flash protection register. It is the user's responsibility to program the NVPROT register to protect the bootloader code. For detailed information on the protection mechanism, please refer to the Section 4.5.6 of MC9S08JM60 Data Sheet.

2.3 Vector Redirection

The vector table (address 0xFFC0–0xFFFF) cannot be updated, if the flash memory protection mechanism is enabled. All vectors except for reset, must be redirected to the proper addresses for the interrupt service subroutines in the user code. The MC9S08JM60 supports vector redirection if the FNORED-bit in the NVOPT is programmed to zero. For example, if the flash memory with the address 0xFC00–0xFFFF is protected, the clear FNORED-bit redirects the vector table from 0xFFC0–0xFFFD to the location 0xFBC0–0xFBFD. It is the user's responsibility to program the FNORED-bit to redirect the vector table. For detailed information on the vector redirection mechanism, please refer to the Section 4.5.7 of the MC9S08JM60 Data Sheet.

2.4 Software Startup Process

Figure 3 shows the bootloader startup process.

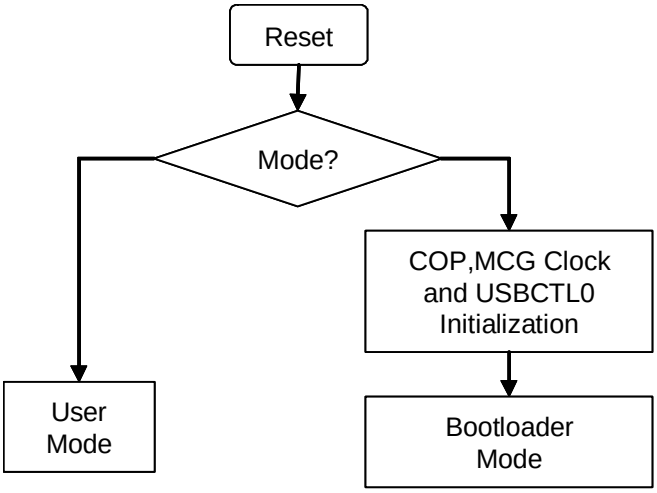

Figure 3. Bootloader Startup Process

Upon reset a small amount of code is executed to determine the mode of operation. In the example bootloader application project, refer to Appendix A, "Example Project," a general purpose input/output PTG0 is used for mode determination. If the PTG0 is high, the target system enters user mode, otherwise it enters the bootloader mode.

2.4.1 User Mode

If the target system enters user mode a jump instruction is executed and jumps to the user application code. The user application code is located in an unprotected flash memory area (0x1960–0xFBFF).

The entry address of the user application code can vary every time the linker builds the project. The mode determination code resides in a protected flash memory, and the code cannot be updated. An absolute entry address of the user mode must be provided to the bootloader. Figure 4 shows an example of how to jump to the user application code using the absolute entry address of the user mode in 0xFB00. If the user mode is determined, the instruction JMP 0xFB00 is executed to jump to address 0xFB00. At address 0xFB00, another JMP instruction calls _Startup() function to run the user application code. This makes the system jump to address 0xFB00 and to enter user mode no matter where the _Startup() is.

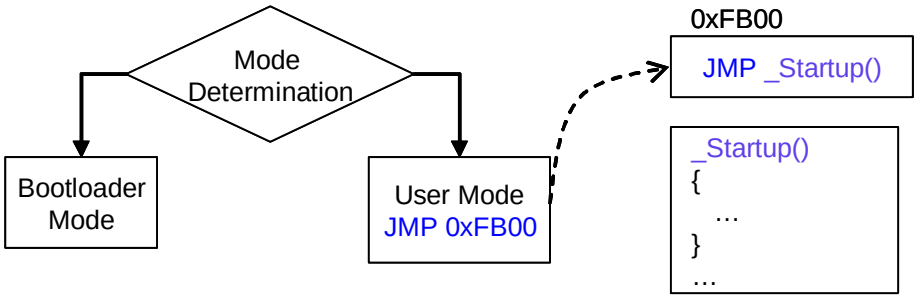

Figure 4. Fixed Entry Address of User Mode

The _Startup() is a standard startup function of C programming. It is the responsibility of the user to completely configure the device.

2.4.2 Bootloader Initialization

The user must take the following steps before the MCU enters the bootloader mode:

1. Disable computer operating properly (COP) before entering the bootloader mode.

   The COP may lead to a USB enumeration failure and flash programming/erase error. The COPT-bits can be set in the system options register 1 (SOPT1) to disable the COP. For detailed information on setting the COP, please refer to the Section 5.4 of the MC9S08JM60 Data Sheet.

   NOTE

   SOPT1 is a write-once register. To use the COP in the application, please set SOPT1 register after the mode determination.

2. Set the multi-purpose clock generator (MCG).

   The MC9S08JM60's USB module requires two clock sources, a 24 MHz bus clock and a 48 MHz reference clock. The 48 MHz reference clock is sourced directly from the MCGOUT. For USB operation on the MC9S08JM60, the MCG must be configured as PLL engaged external (PEE) mode using an external crystal to achieve an MCGOUT frequency of 48 MHz. Please refer to Chapter 12 of the MC9S08JM60 Data Sheet for detailed information.

3. Set the on-chip regulator and the USBDP pullup.

   The MC9S08JM60 has a 3.3 V on-chip voltage regulator that provides a stable power source to power the USB internal transceiver. If the on-chip regulator is enabled, it requires a voltage supply (V_{DD}) of 3.9 V to 5.5 V. The MCU and USB can operate at different voltages, if the on-chip 3.3 V regulator is disabled, a 3.3 V source must be provided through the VUSB33 pin to power the USB transceiver.

   The pullup resistor on the USBDP line is required for full-speed operation by the USB specification Rev. 2.0. An on-chip pullup resistor is implemented in the MC9S08JM60 USB module. This on-chip pullup resistor can also be disabled, and the USB module can be configured to use an external pullup resistor for the USBDP line.

   To configure the on-chip regulator and the on-chip pull up resistor, set or clear the USBPU-bit and USBVREN-bit in the USBCTL0 register. This step must occur before entering bootloader mode. The bootloader code retains this information when it updates the other bits in the USBCTL0 register. Table 1 summarizes how to configure the USBCTL0 register before it enters bootloader mode. For detailed on-chip regulator and on-chip pullup resistor information, please refer to Section 17.3.1 of the MC9S08JM60 Data Sheet.

Table 1. USBCTL0 Configuration for Different Applications

3.3 V Voltage Regulator	USBDP Pullup Resistor	USBCTL0 Setting
Internal	Internal	0x44
Internal	External	0x04
External	Internal	0x40
External	External	0x00

If the startup process is modified by the user, care must be taken to configure SOPT1, MCG and USBCTL0. Improper configuration may lead to a bootloader malfunction and damage the target system.

2.4.3 Bootloader Mode

Figure 5 shows the software flowchart of the bootloader mode. The USB enumeration starts after the USB cable is plugged in. There is a request to install a PC Driver the first time the USB cable is plugged in. If the enumeration succeeds, the target system is identified as an MC9S08JM60 USB ICP Device. A PC GUI Tool can then be used for in-circuit programming. The PC Driver and PC GUI Tool is explained in Section 2.5, "PC Driver and PC GUI Tool."

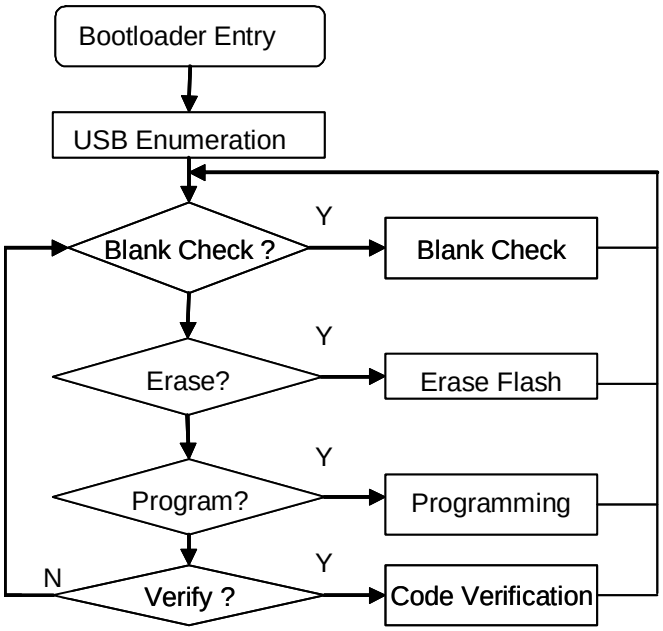

Figure 5. Bootloader Mode Flowchart

The code for the bootloader mode is available in the source code and the object binary library. The interface between the user code and library is called Bootloader_Main(). The target system enters bootloader mode by calling the Bootloader_Main() function. This code must be located in the flash memory (address 0xFC00–0xFFAF) while compiling and linking. To exit Bootloader Mode a power on reset (POR) must be conducted.

NOTE

If using the bootloader code in the library format the Bootloader_Main () which is the entry function cannot be changed.

2.5 PC Driver and PC GUI Tool

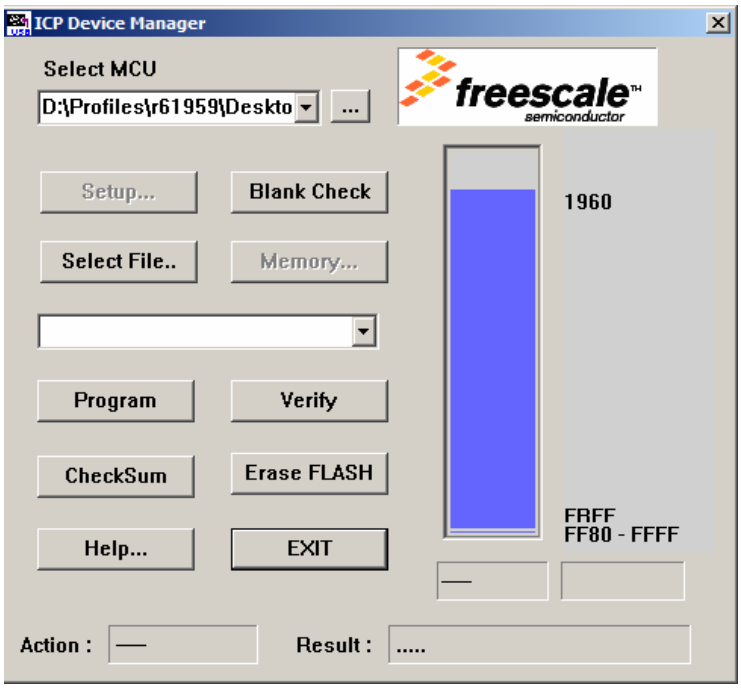

Figure 6. USB PC GUI Tool

The PC Driver and PC GUI Tool for the MC9S08JM60 are almost the same as those for the MC68HC908JB16. Please refer to AN2399 and its software. The PC Driver is the PC resident software that communicates with the USB bootloader. If the PC Driver is not installed, the PC does not recognize the MC9S08JM60 in bootloader mode. The driver for the MC9S08JM60 contains USBICP.inf and the USBICP.sys. They can be found in the software for this application note.

The PC GUI Tool sends commands to the target system to program or erase. Figure 6 shows the snapshot of this PC GUI Tool. The details on how to use this tool is explained in Section 3.3, “Running PC Tool.”

3 Tutorial for Bootloader Implementation

This section details the procedure to implement the USB bootloader for the MC9S08JM60. An example project is provided for reference in Appendix A, “Example Project.” This example project can be used as a template to create a project. Please see AN3561SW for a Codewarrior project containing the software.

3.1 Software Integration

The compiler for the bootloader application is CodeWarrior for the HC(S)08, version 5.1 or above. The following steps show how to create a simple project with the bootloader supported.

1. Create a new project

   Use CodeWarrior new project wizard to create a new project for the MC9S08JM60.

2. Modify the prm link file as shown in Figure 7

   a) Flash address 0xFC00–0xFFAF is reserved for the bootloader code. The bootloader code segment named Bootloader_ROM is placed in this area.

   NOTE

   The bootloader segment must be named Bootloader_ROM to compile the bootloader into the protected flash address 0xFC00–0xFFAF.

   b) The RAM address 0xB0 to 0xBA is used as bootloader variables. The start address of Z_RAM must be changed to 0xBB .

   c) Modify reset vector. Use the _Entry function instead of _Startup function as the VECTOR 0. Comment out _Startup function and add the _Entry function.

```
SEGMENTS /* Here all RAM/ROM areas of the device are listed. Used in PLAC
    ROM                 =  READ_ONLY    0x1960 TO 0xFBFF;
    Bootloader          =  READ_ONLY    0xFC00 TO 0xFFAF;
    Z_RAM               =  READ_WRITE   0x00BB TO 0x00FF;
    RAM                 =  READ_WRITE   0x0100 TO 0x10AF;
    RAM1                =  READ_WRITE   0x1860 TO 0x195F;
    ROM1                =  READ_ONLY    0x10B0 TO 0x17FF;
    ROM2                =  READ_ONLY    0xFFC0 TO 0xFFC3;
END

PLACEMENT /* Here all predefined and user segments are placed into the SE
    DEFAULT_RAM                         INTO  RAM,RAM1;
    DEFAULT_ROM, ROM_VAR, STRINGS       INTO  ROM; /* ROM1,ROM2 In case
    DATA_ZEROPAGE, MY_ZEROPAGE          INTO  Z_RAM;
    Bootloader_ROM                      INTO  Bootloader;
END

STACKSIZE 0x50

//VECTOR 0 _Startup /* Reset vector: this is the default entry point for
VECTOR 0 _Entry
```

Figure 7. Modify prm File

3. Initialize NV registers

   The NVOPT and NVPROT register must be initialized for flash block protection and vector table redirection. Use the commands shown in Figure 8 to initialize the NV registers.

```
const byte NVOPT_INIT  @0x0000FFBF = 0x02;     // vector redirect, flash unsecure
const byte NVPROT_INIT @0x0000FFBD = 0xFA;     // 0xFC00-0xFFFF are protected
```

Figure 8. Initialize NV Registers

NOTE

The NVOPT and NVPROT registers are located in the nonvolatile memory. They can be written once and it must be the first lines of the code. Please refer to Appendix A, "Example Project."

4. Load bootloader library to the project

   Select CodeWarrior menu "Project -> Add Files." as shown in Figure 9 and add the JM60_Bootloader _Vx.y.lib to the library.

   The Bootloader library is named JM60_Bootloader _Vx.y.lib. The Vx.y is the version number. This library can be found in the folder \Template with Bootloader\Sources\.

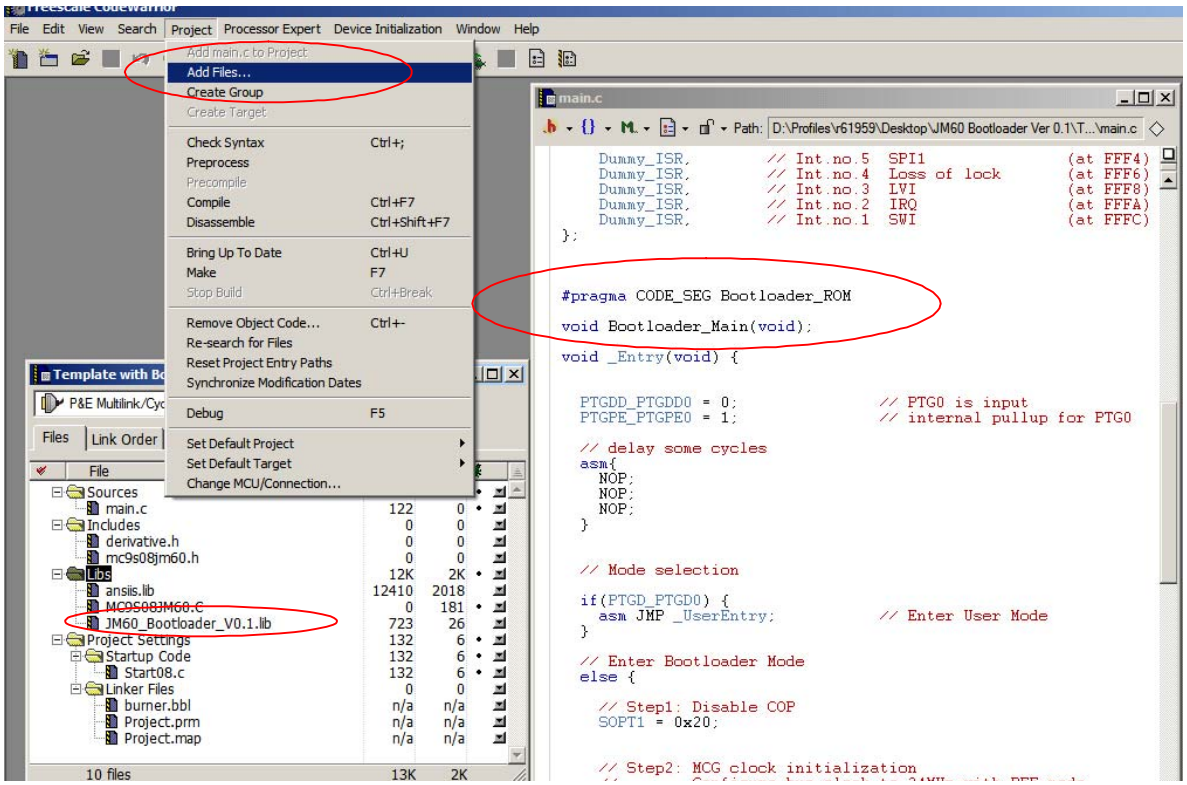

Figure 9. _Entry Function

5. Mode determination

   Figure 9 shows an example on how to determine a mode. An I/O port PTG0 is configured as input. If the PTG0 is logic 1 during the startup period, the target system enters the user mode, otherwise the bootloader initializes and enters the bootloader mode.

6. Fix the entry address of the user mode

   Please refer to Section 2.4.1, “User Mode” on how to fix the entry address of the user mode.

7. Bootloader initialization

   To ensure that the code is located in the protected flash area a segment declaration #pragma CODE_SEG Bootloader_ROM must be set before the entry function. The bootloader initialization must include the COP disable, MCG clock and the USBCTL0 register initialization.

Compile and link the project after all the steps are finished. Read the .map file to check whether the bootloader code is located in the protected flash address. The application code can now be added to this project.

3.2 PC Driver Installation

The software needs to be downloaded to the target system after the project is built. The software must be downloaded by other tools such as the BDM Multilink because the target system has no bootloader code. Please refer to *Development Support of HCS08 Family Reference Manual*, (chapter 7) about the BDM Multilink.

Below are steps to install the PC Driver:

1. Start the target system, choose to enter bootloader mode, driver install window appears when the USB cable is plugged in as shown in Figure 10. Select Install from a list or specific location (Advanced) and then click Next.

Figure 10. Driver Install Window

2. Specify the directory containing the USBICP.inf file and then click Next as shown in Figure 11.

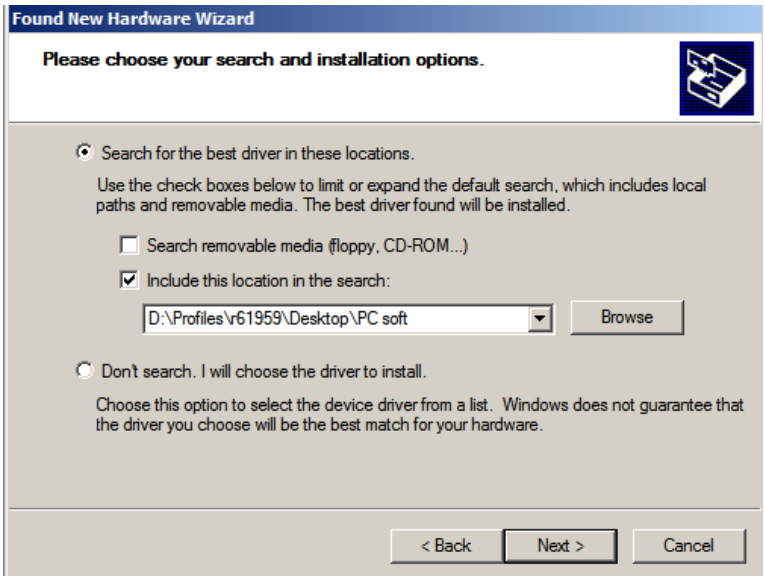

Figure 11. Locate USBICP.INF File

3. If the window as shown in Figure 12 appears, locate the directory containing the USBICP.sys file. The file below is in the PC soft folder of the software package of this application note.

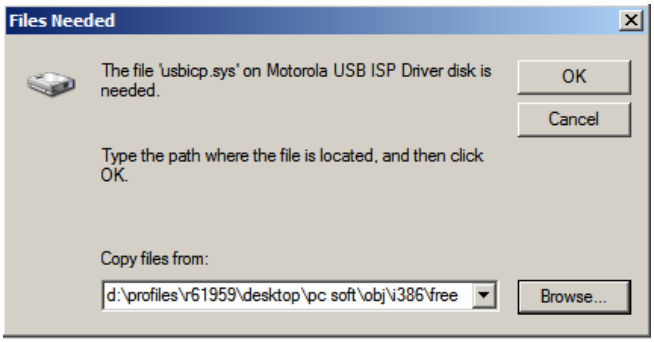

Figure 12. Install USBICP.sys File

4. If the interface as shown in Figure 13 appears, click Finish to complete the installation.

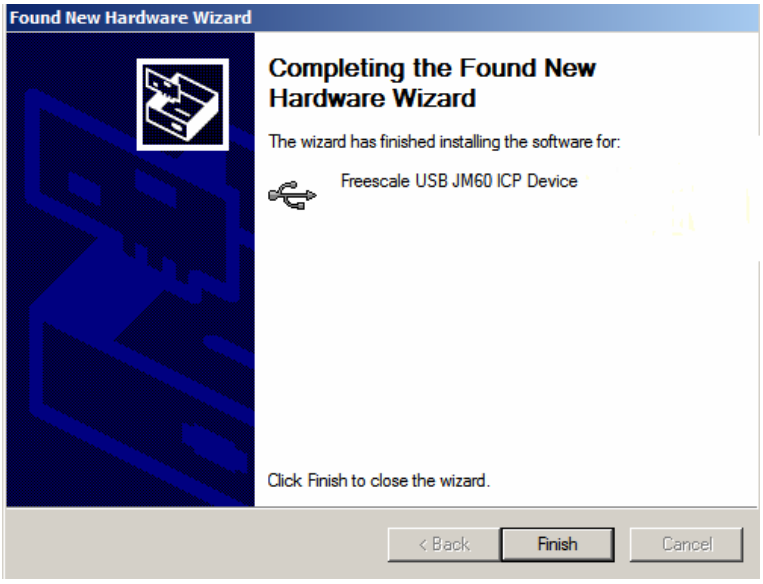

Figure 13. Driver Install Finish

3.3 Running PC GUI Tool

The PC GUI Tool is used for programming and erasing. Please see below instructions.

3.3.1 Launch PC GUI Tool

Open the USBICP.exe file and load the parametric file as shown in Figure 14. The parametric file for the MC9S08JM60 is 9S08JM60.imp. The USBICP window then appears as shown in Figure 15 and is ready to use.

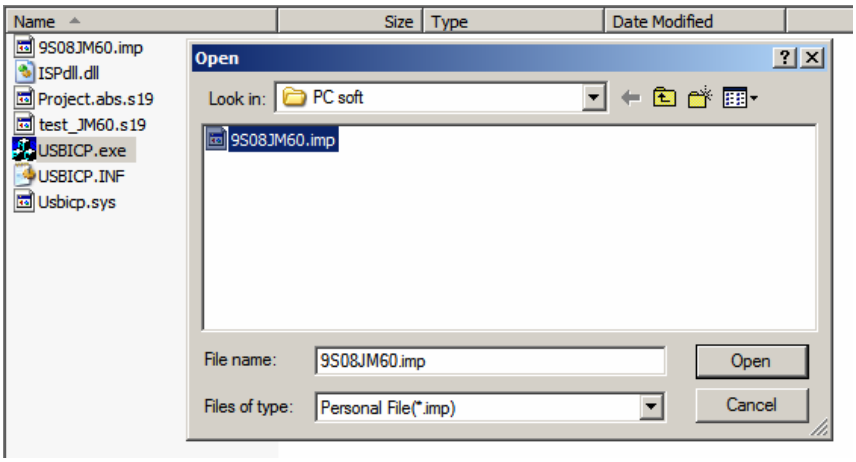

Figure 14. Launch PC Toll

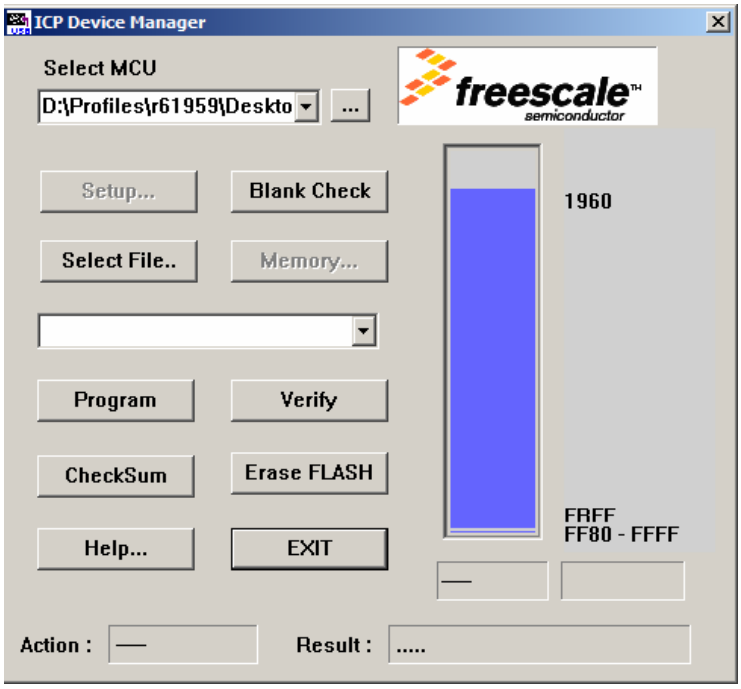

Figure 15. USBICP Window

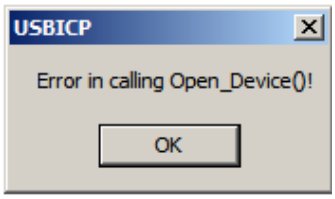

Figure 16. Error in Running Tool

If an error window appears as in Figure 16, please check if the USB enumeration succeeded or parametric file is loaded. If the USB enumeration succeeds, the Freescale USB JM60 ICP Device can be seen from the Windows device manager as shown in Figure 17. If the USB enumeration fails, please check the following:

- USB cable is plugged in
- MCG output 48 MHz clock
- COP disable
- USBCTL0 is configured as listed in Table 1
- Call Bootloader_Main() function
- PC Driver is installed

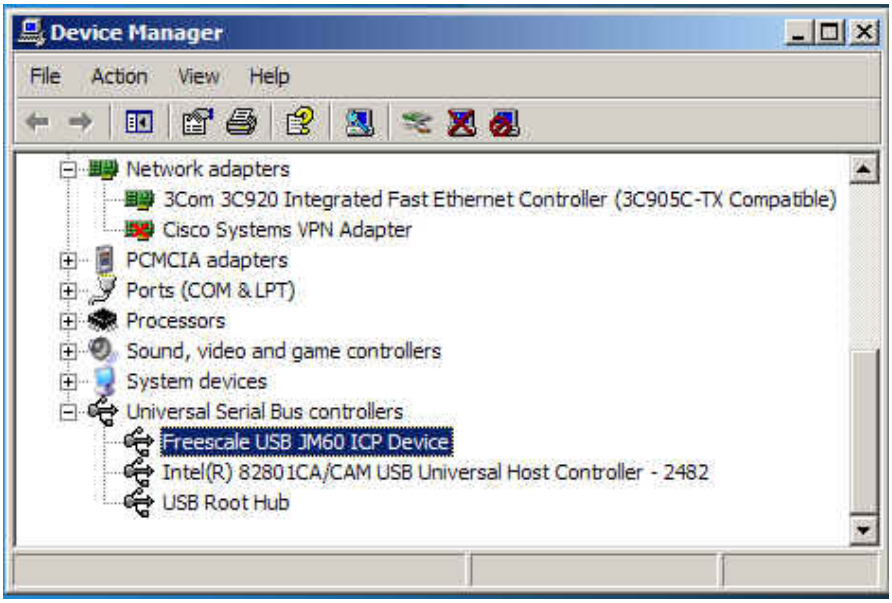

Figure 17. USB Enumeration Success

3.3.2 Erase Flash

Click FLASH button to erase the flash memory with the address 0x1960 to FBFF. If it succeeds, a window appears as shown in Figure 18. Erase the flash memory before programming.

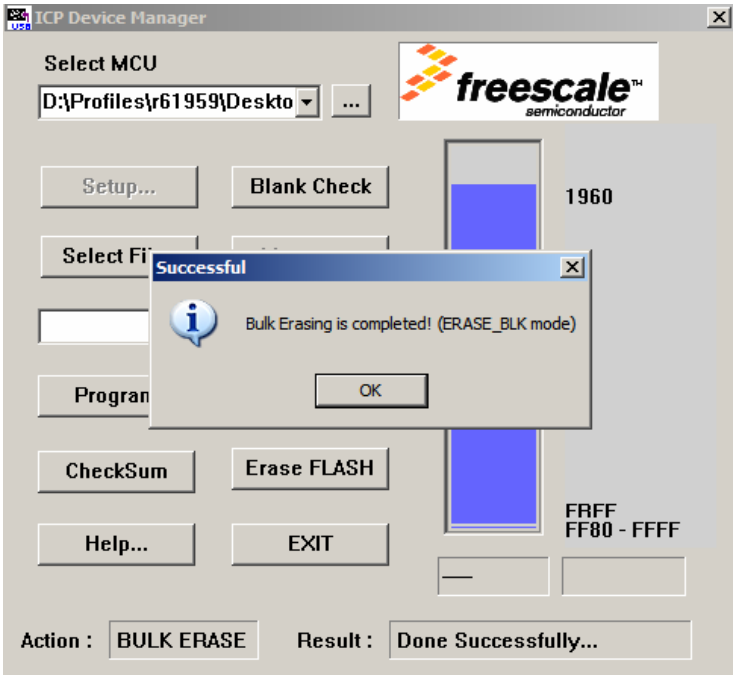

Figure 18. Erase Flash

If the flash memory can not be erased, check the following items:

- Flash memory clock setting is correct
- Flash is not secured or protected
- COP is disabled

3.3.3 Blank Check

Click the Blank Check button to check whether the flash address 0x1960 to 0xFBFF is blank. If it succeeds, a windows appears as shown in Figure 19.

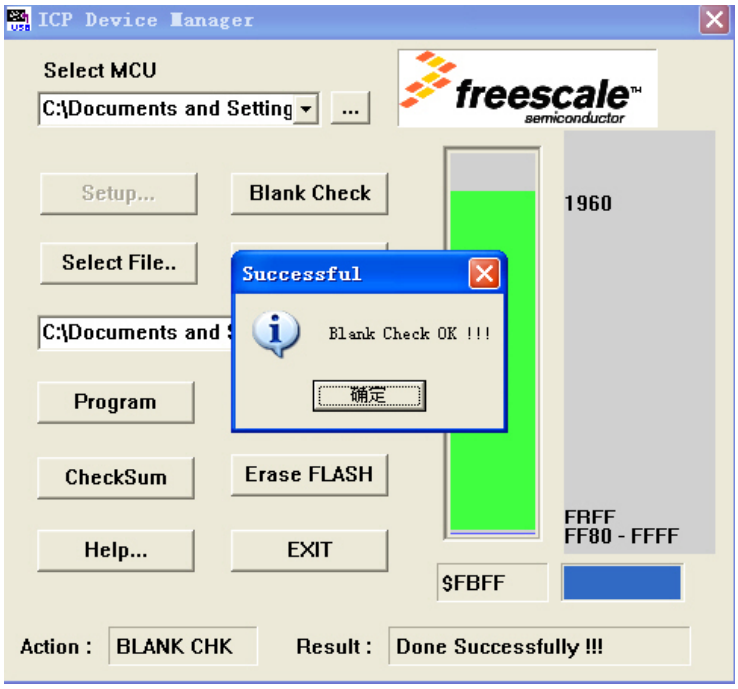

Figure 19. Blank Check

3.3.4 Programming Flash

Click the Select File button to select the .s19 file to be programmed as shown in Figure 20. Only the S-record file is supported by this USB ICP Tool.

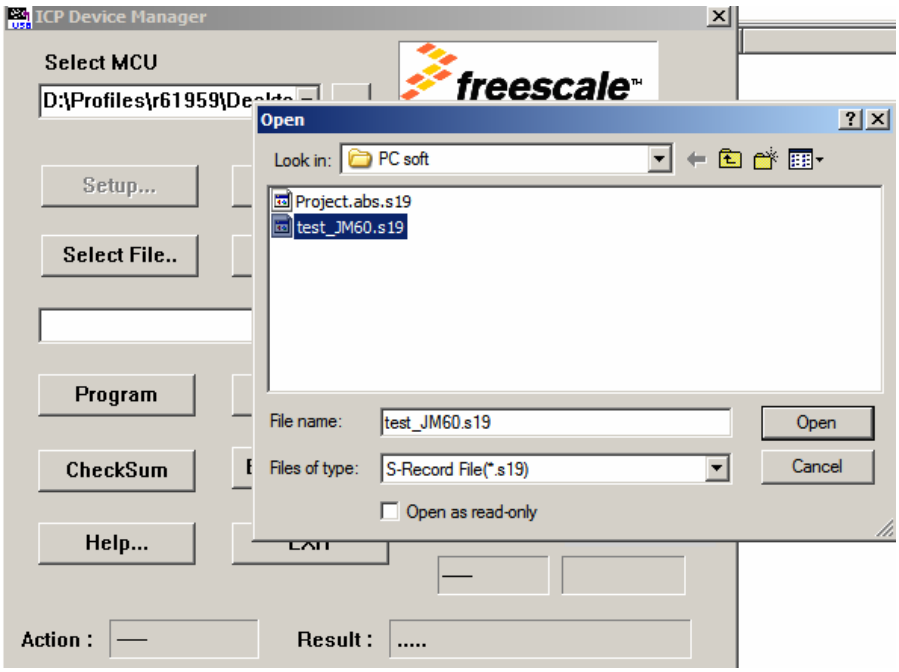

Figure 20. Load File

Click the Program button to program the flash memory. Only the flash address 0x1960 to 0xFBFF can be programmed. The flash memory out of this range is ignored. If it succeeds, a successful window appears as shown in Figure 21.

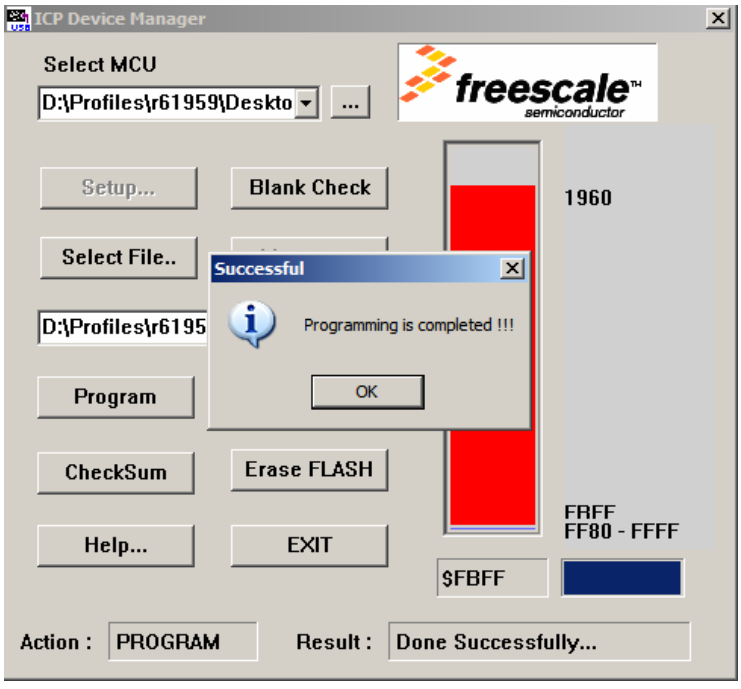

Figure 21. Programming Flash

If the flash memory can not be programmed, check the following items:

- The S-record file loaded is correct
- Flash memory is blank before programming
- Flash memory clock setting is correct
- Flash is not secured or protected
- COP is disabled

3.3.5 Code Verification

The bootloader can verify the code in the MCU after it is programmed. Click the Verify button for verification. If the code is programmed correctly, verification succeeds as shown in Figure 22.

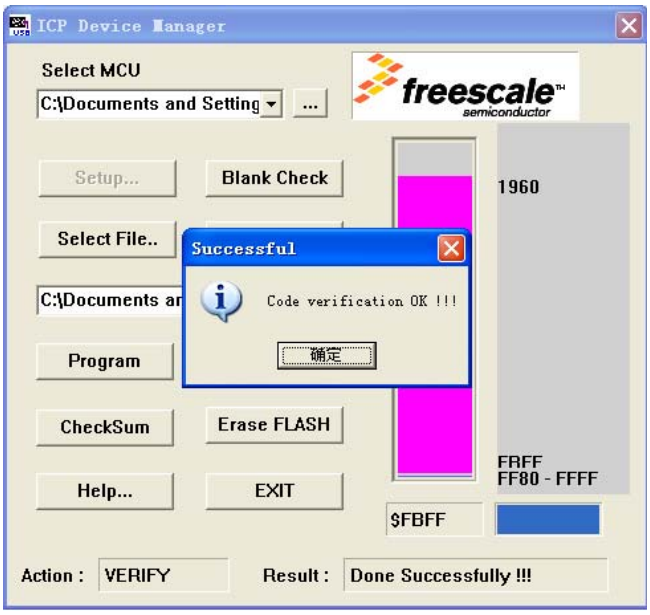

Figure 22. Code Verification

4 Conclusions

This application note has covered the following topics:

- Background information — Introduces what the USB bootloader is and its benefits.
- Bootloader overview — The bootloader startup process, initialization and PC GUI tool are explained.
- Tutorial for bootloader implementation — Step by step instructions on how to use bootloader.
- An example project.

Appendix A Example Project

An example project is provided for reference. This project is based on the MC9S08JM60 demonstration board and CodeWarrior for the HC(S)08 V5.1 with the MC9S08JM60 service pack.

Demonstration procedure is:

1. Download the project into the MC9S08JM60 demonstration board
2. Power on the board with the PTG0 button pressed to enter bootloader mode
3. After the driver is finished installing, it is ready for use.

Please find the project in the AN3561SW for this application note.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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

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

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

Document Number: AN3561
Rev. 0
11/2007

freescale™
semiconductor