Freescale Semiconductor
Application Note

Document Number: AN4252
Rev. 0, 4/2011

USB HID bootloader for the MC9S08JM60

by: Derek Lau
System and Solution Engineering, Microcontroller Solutions Group
Hong Kong
China

Contents

1 Introduction

In-circuit programming (ICP) is a process for a microcontroller (MCU) to be programmed or re-programmed on the printed circuit board. It allows the user code to be changed during product development and production, and after sold.

The MC9S08JM60 (JM60) is a member of the low-cost, high-performance HCS08 family of 8-bit microcontroller units. It has a 60 KBytes of embedded flash memory that can be programmed or erased without special voltage input. The JM60 has a USB 2.0 full-speed module, making it suitable for in-circuit programming via a USB interface. The using of the human input device class (HID) allows the use of standard HID drivers provided by most operating systems.

2 System overview

The system includes:

- JM60 bootloader firmware
- JM60 keyboard demo user program
- JM60 mouse demo user program
- Windows PC demo software (executive file only)

The bootloader is a small program put into the JM60 that complies with the USB HID class, which receives commands and data from the host to program and erase the flash memory of the JM60. The keyboard and mouse demo user programs are two examples showing how user programs can be programed and re-programed into the JM60 by the bootloader via the PC demo software. The firmware and demo user program was tested under the hardware platform of the DEMOJM board. The PC demo software was tested under 32-bit Windows 7 and XP systems.

3 Flash utilization

The bootloader is put into the highest flash area of the JM60. The user program and the vectors are put into the flash area below the bootloader flash. The following figure shows the flash memory map.

<table>
<tr><td>$10B0 ↓ $17FF</td><td>Flash 1
1872 Bytes</td><td>Flash 1 (user program or data)
1872 Bytes</td></tr>
<tr><td>$1800 ↓ $195F</td><td>USB RAM
256Bytes</td><td>USB RAM
256Bytes</td></tr>
<tr><td>$1960 ↓ $F7C3</td><td rowspan="3">Flash 2
58,960 Bytes</td><td>Flash 2A (user program or data)
56932 Bytes</td></tr>
<tr><td>$F7C4 ↓ $F7FF</td><td>Redirection Vectors
60 Bytes</td></tr>
<tr><td>$F800 ↓ $FFAF</td><td>Bootloader
1968 Bytes</td></tr>
<tr><td>$FFB0 ↓ $FFBF</td><td>Non volatile registers
16 Bytes</td><td>Non volatile registers
16 Bytes</td></tr>
<tr><td>$FFC0 ↓ $FFFF</td><td>FLASH Vectors
64 Bytes</td><td>Bootloader
64 Bytes</td></tr>
</table>

Figure 1. Memory map

Only continuous protection of blocks starting from the highest address is allowed. Therefore, the bootloader code is put into the flash address of 0xF800 to 0xFFAF and 0xFFC0-0xFFFF which is protected from erasing by writing 0xF6 to the flash protection register. The user program can be put into the flash 1 and flash 2A areas. The protection of the bootloader code ensures that only the user program is changed while the bootloader will never be accidentally erased during user program upgrade. When any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. The bootloader code therefore has enabled the vector redirection by programming the vector redirection disable (FNORED) bit of the nonvolatile location (NVOPT) register located at address 0xFFBF to zero. Since the flash address from 0xF800 is protected and vector redirection is enabled, all of the interrupt vectors (memory locations 0xFFC4-0xFFFD) except the reset vector are redirected to 0xF7C4-0xF7FD. The user program starting address is put into 0xF7FE-0xF7FF. The bootloader program will jump to the address pointed by 0xF7FE-0xF7FF to run the user program in normal user mode. If the address is blank, bootloader mode is run.

4 Bootloader commands

The bootloader commands and data are sent by the PC software "bootloader.exe" to JM60 through USB HID class protocol. The bootloader command format is shown in the following table with 0xA5 as an identifier for starting of command followed by the command, arguments and data.

Table 1. Bootloader command format

Offset	Field	Size (byte)	Description
0	Command_Start	1	Command identifier (0xA5)
1	Command	1	Command
2-63	Arguments and data	4-62	Arguments and data

4.1 Block erase command

The Block_Erase command enables erasing of one flash block. The argument contains any address within the flash block to be erased. The device returns the status after receiving and executing the command.

Table 2. Block erase command

Command	Arguments	Description
Block_Erase (0x01)	Erase_Address	Any address within the block to be erased(4 bytes)

Table 3. Block erase example

From	Data	Description
Host	OUT [A5 01 00 DE 00 XX XX XX ... XX]	Block erase DE00 to DFFF
Device	IN [A5 01 XX XX]	Device returns status of block erase success

4.2 Bytes program command

The Bytes_Program command is with arguments of starting address, number of bytes for program, and data to be programmed. The device returns the status after receiving and executing the command.

Table 4. Bytes program command

Command	Arguments	Description
Bytes_Program (0x11)	Start_Address	Starting address (4 bytes)
	Len_Data	Number of bytes to be programmed (2 bytes)
	Data	Data to be programmed (1-56 bytes)

Table 5. Bytes program example

From	Data	Description
Host	OUT [A5 11 00 DE XX XX 38 00 D0 D1 D2 D3 D4... D55]	Bytes program 56 bytes of data from DE00 to DE37
Device	IN [A5 01 XX XX]	Device returns status of program success

4.3 Status

The device returns four bytes of data after completing the Block_Erase or Block_Program command, which includes one byte of 0xA5 as the identifier, one byte of status, and two reserved bytes.

Table 6. In data command

Status	Description
0x01	Command success
0xFF	Command failure

5 Demo

The demo shows the procedures of programming the bootloader code into the JM60 using codewarrior, and programming and re-programming of keyboard and mouse user programs through the PC software “bootloader.exe”. The following table shows how the pins of the JM60 are used in the demo. The figure below shows the JM60 Demo Board (DEMOJM).

Table 7. JM60 pins usages

Pins	Description
PTE2	Caps Lock LED control for the keyboard demo.
PTE3	Num Lock LED control for the keyboard demo.
PTF0	Scroll Lock LED control for the keyboard demo.
PTG0	Press the button and plug-in the USB cable will cause JM60 to enter bootloader mode. Page Up key for the keyboard demo.
PTG1	Page Down key for the keyboard demo.

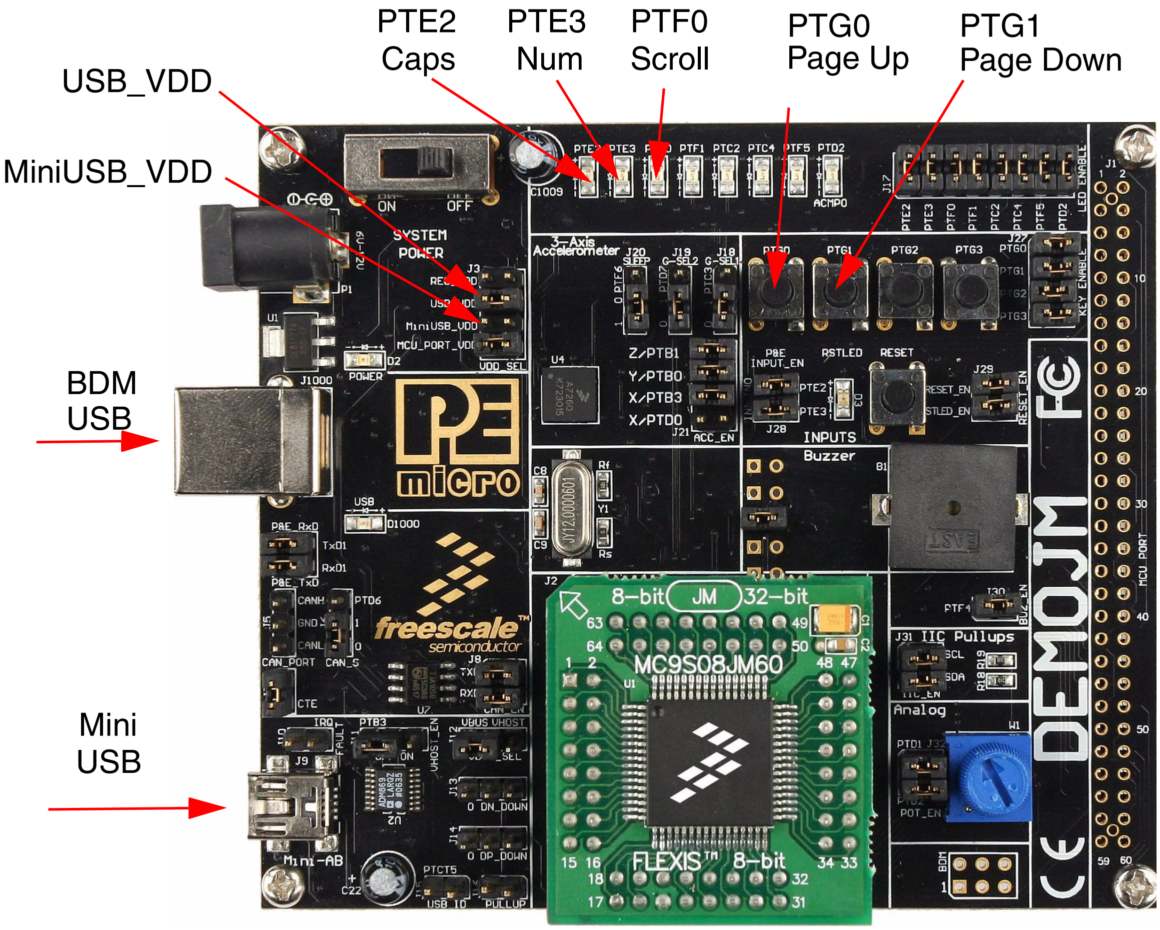

Figure 2. DemoJM board

5.1 Programming bootloader into JM60

The bootloader code can be programmed into the JM60 using Freescale Codewarrior Development Studio.

1. Connect the USB_VDD jumper to select the V_{DD} from the BDM USB port.
2. Connect a USB cable from the PC to the DBM USB port and install the driver if prompted.
3. Launch Codewarrior version 6.3.
4. Click <File>, then select <Open> to open the project file "hid_bootloader.mcp" under the directory "jm60_hid_bootloader_code".
5. Click <Project>, then select <Debug> to automatically download and program the s-record of the bootloader code into JM60.
6. Close debugger and remove the USB cable.

The JM60 has been programmed with the bootloader code. It always runs in bootloader mode when no user program is put into it.

5.2 Programming USB keyboard demo

1. Connect the mini_USB jumper to select the V_{DD} from the mini-USB port.
2. Connect a USB cable from the PC to the mini-USB port.

3. Run the file "HID_Bootloader.exe" in the directory of "jm60_hid_bootloader".

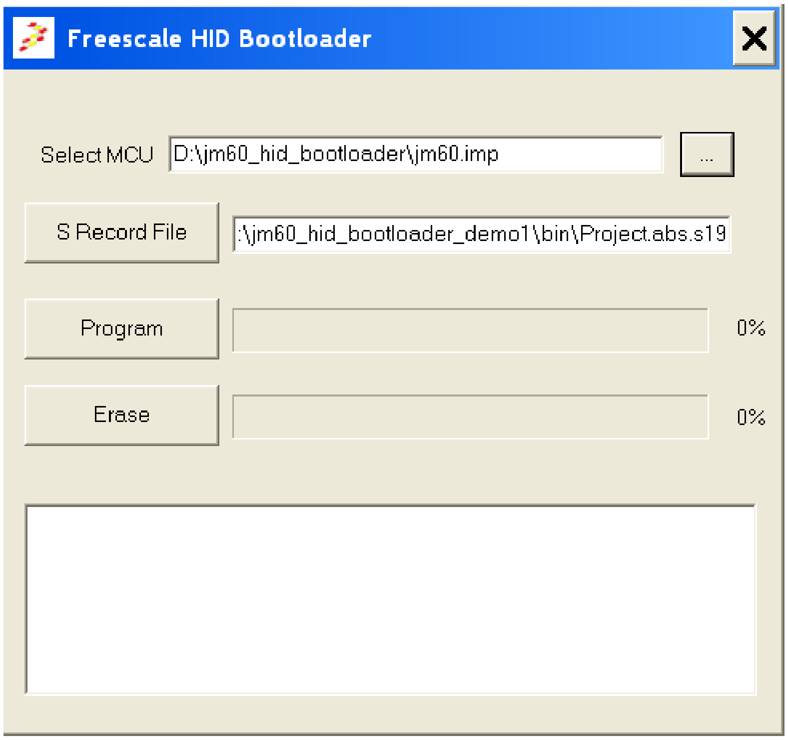

Figure 3. HID bootloader PC program

4. In <Select MCU>, select the file "jm60.imp" in the directory of "jm60_hid_bootloader".
5. In <S Record File>, select the file "Project.abs.s19" in the directory "jm60_hid_bootloader_demo1\bin" (customers can choose their own s-record files to program)
6. Click Program to program the USB keyboard demo user program into the JM60.
7. Unplug and replug the USB cable.

Now the USB keyboard demo user program has been put into the JM60. The user program runs if PTG0 is not pressed. The system functions as a USB keyboard. PTG0 and PTG1 function as the Page up and Page Down keys while the LEDs connected to the PTE2, PTE3, PTF0 function as the Caps Lock, Num Lock and Scroll Lock indicators.

5.3 Programming USB mouse demo

Since the JM60 has been programmed with a user program, the bootloader program runs if PTG0 is pressed during JM60 is power up.

1. Unplug the USB cable.
2. Press PTG0 and plug-in the USB cable
3. In S Record File, select the file "Project.abs.s19" in the directory "jm60_hid_bootloader_demo2\bin".
4. Click <Erase>
5. Click <Program> to program the USB mouse demo user program into the JM60.
6. Unplug and replug the USB cable.

Now the JM60 has been programed with the JM60 mouse demo user program. The system emulates a USB mouse making the cursor move from left to right and from right to left.

6 Customization

The major differences between a normal user program and an ICP user program are the flash areas, vector table location and the startup address. This section shows how a normal user program is modified to be used under ICP.

6.1 ROM segment

The user program can only be put into the flash area of 0x1000-0x17FF and 0x1960-0xF7FF. The ROM segment defined at the file "project.prm" may be modified as below:

```
ROM READ_ONLY 0x1960 TO 0xF7FF
```

6.2 Reset vector and startup position

To work with our bootloader code, the interrupt vector table of an ICP user program is located at 0xF7C4-0xF7FC and the startup address is located at 0xF7FE-0xF7FF. The following code is an example to locate the reset vector and the startup address for an ICP user program.

```
#define BOOTLOADER_START_ADDR 0xF800
void (* volatile const _Usr_Vector[])()@(BOOTLOADER_START_ADDR-0x100+0xC4)={
RTC_ISR, // Int.no.29 RTC (at F7C4)
I2C_ISR, // Int.no.28 I2C (at F7C6)
ACMP_ISR, // Int.no.27 ACMP (at F7C8)
ADC_ISR, // Int.no.26 ADC (at F7CA)
KBI_ISR, // Int.no.25 KBI (at F7CC)
Dummy_ISR, // Int.no.24 SCI2 Transmit (at F7CE)
Dummy_ISR, // Int.no.23 SCI2 Receive (at F7D0)
Dummy_ISR, // Int.no.22 SCI2 Error (at F7D2)
SCI1Tx_ISR, // Int.no.21 SCI1 Transmit (at F7D4)
SCI1Rx_ISR, // Int.no.20 SCI1 Receive (at F7D6)
SCI1Err_ISR, // Int.no.19 SCI1 error (at F7D8)
Dummy_ISR, // Int.no.18 TPM2 Overflow (at F7DA)
Dummy_ISR, // Int.no.17 TPM2 CH1 (at F7DC)
Dummy_ISR, // Int.no.16 TPM2 CH0 (at F7DE)
Dummy_ISR, // Int.no.15 TPM1 Overflow (at F7E0)
Dummy_ISR, // Int.no.14 TPM1 CH5 (at F7E2)
Dummy_ISR, // Int.no.13 TPM1 CH4 (at F7E4)
Dummy_ISR, // Int.no.12 TPM1 CH3 (at F7E6)
Dummy_ISR, // Int.no.11 TPM1 CH2 (at F7E8)
TPM1CH1_ISR, // Int.no.10 TPM1 CH1 (at F7EA)
TPM1CH0_ISR, // Int.no.9 TPM1 CH0 (at F7EC)
Dummy_ISR, // Int.no.8 Reserved (at F7EE)
USB_ISR, // Int.no.7 USB Statue (at F7F0)
Dummy_ISR, // Int.no.6 SPI2 (at F7F2)
Dummy_ISR, // Int.no.5 SPI1 (at F7F4)
Dummy_ISR, // Int.no.4 Loss of lock (at F7F6)
Dummy_ISR, // Int.no.3 LVI (at F7F8)
Dummy_ISR, // Int.no.2 IRQ (at F7FA)
USB_ISR, // Int.no.1 SWI (at F7FC)
_Startup, // startup address (at F7FF)
};
```

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor
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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4252
Rev. 0, 4/2011