

# Serial Bootloader for 56F82xx

by: William Jiang  
System & Applications Asia Pacific  
Microcontroller Solutions Group

## 1 Introduction

This document describes the static serial bootloader and dynamic serial bootloader as well as their usage requirements for applications. The static serial bootloader can reside in flash forever after downloading an application code, whereas the dynamic version will destroy itself after downloading an application code and can only be used once. This document describes the both types of the serial bootloader, and their usage requirements for applications.

It is designed under CW8.3 and it has about 2 KB. It is located in program flash from address 0x7C00 to 0x7FFF.

For the static serial bootloader, once a reset occurs, the bootloader starts to run, after a boot delay in seconds, even without receiving the application s-record, it will jump to the already programmed application code. If receiving the application s-record, it will program the application code to the on-chip flash and after completion, will jump to the application startup code.

## Contents

1	Introduction	1
2	Serial Bootloader process flow	3
3	Program flash usage	4
4	How to make an application downloadable with Static Serial Bootloader	5
5	How to Select a Target for the Supported Device	10
6	Serial communication	10
7	How to generate a S-record file	10
8	How to download the Bootloader code	11
9	How to download the Application code	12
10	Error code	14
11	Technical support	15

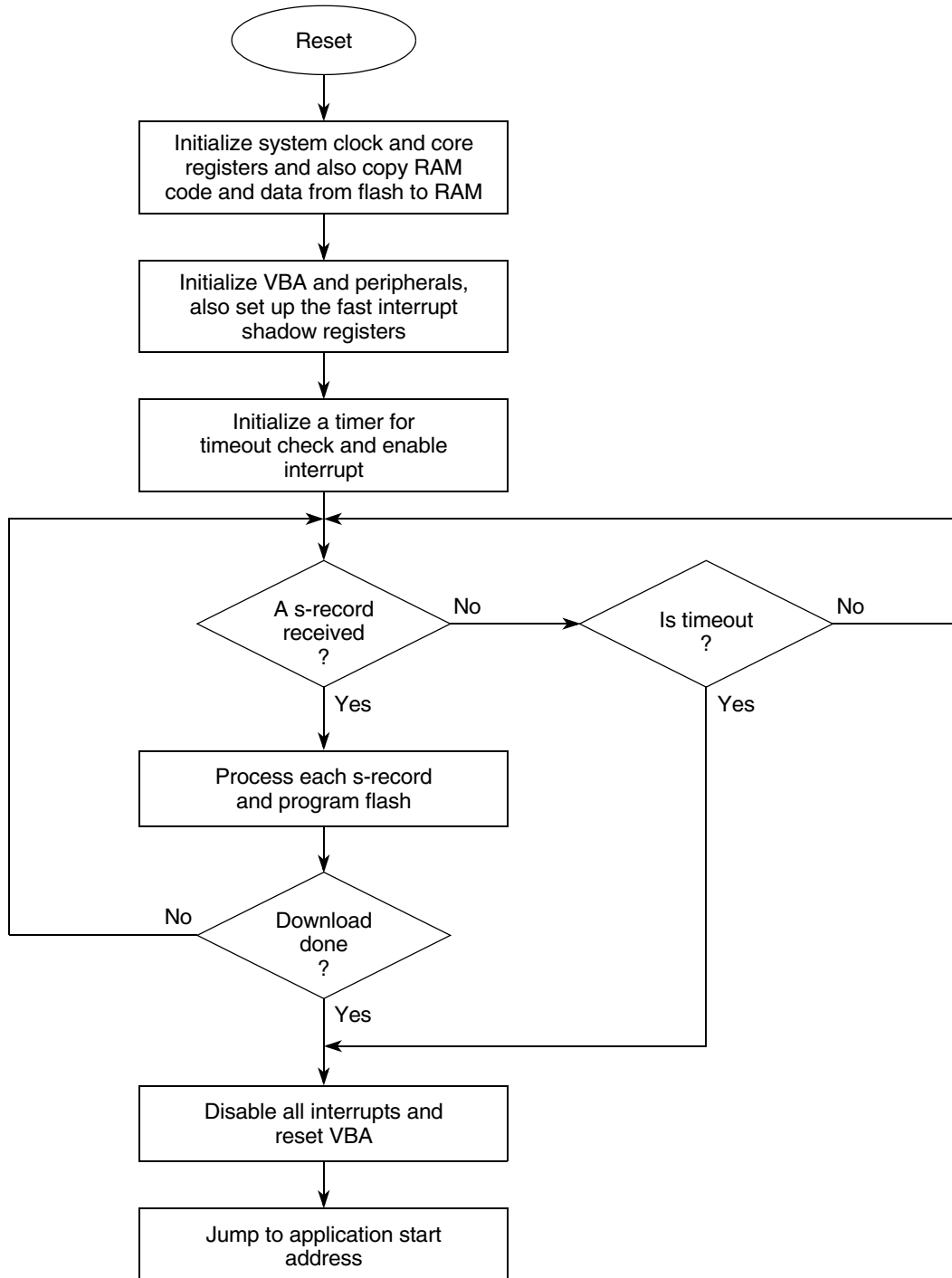
## Introduction

After a reset, the bootloader starts to run again and the above procedure repeats. In this case, the page erase method is used to erase the application flash areas and the bootloader code remains. To use the static serial bootloader, there are some usage requirements for applications described in this document.

For the dynamic serial bootloader, if an application code is not already loaded, then after reset, the bootloader starts to run, after a boot delay in seconds, even without receiving the application s-record, it will jump to the already programmed application code. If receiving the application s-record, it will program the application code to the on-chip flash and after completion, will jump to the application startup code. After a reset, the application will be started to run immediately. In this case, the mass erase method is used. So the bootloader code is erased and will never be alive after reset. There is no usage requirement for applications to use the dynamic serial bootloader. So the serial bootloader that is factory programmed is the dynamic serial bootloader.

## 2 Serial Bootloader process flow

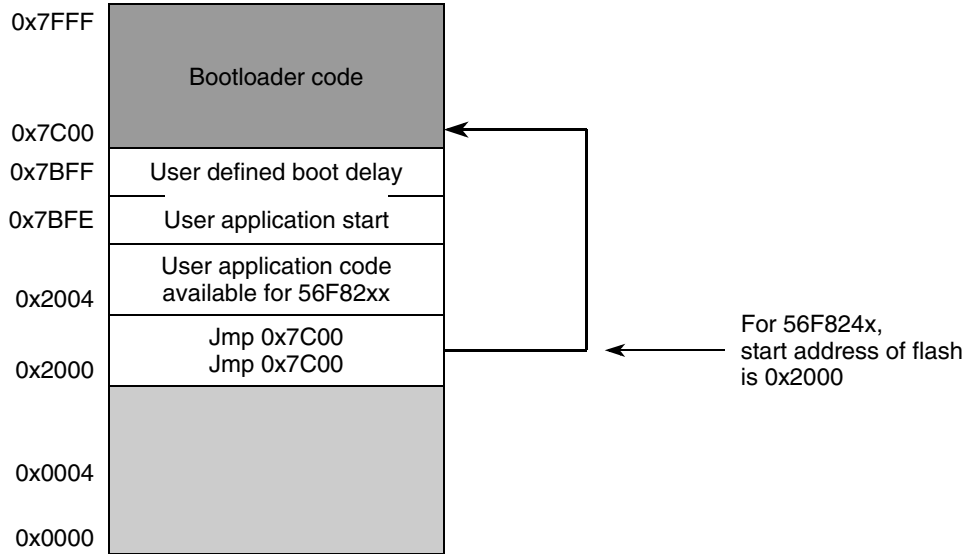
The following [Figure 1](#) shows the main process flow of the serial bootloader.



**Figure 1. Bootloader process flow**

### 3 Program flash usage

The program flash usage by the serial bootloader is shown in [Figure 2](#) and [Figure 3](#).



**Figure 2. Program Flash Usage for 56F824x Serial Bootloader**

**NOTE**

- Do not overwrite the memory range from 0x7C00 to 0x7FFF if the static serial bootloader is used, because this area is occupied by the serial bootloader.
- For 56F824x, the lower memory area from 0x0000 to 0x1FFF is reserved, both hardware reset and COP reset vectors are located at 0x2000 to 0x2003. Therefore, two “jump to bootloader” instructions must be placed at these locations. However, for 56F825x, these locations can be used for user application code because both reset vectors are located at 0x0000 to 0x0003.

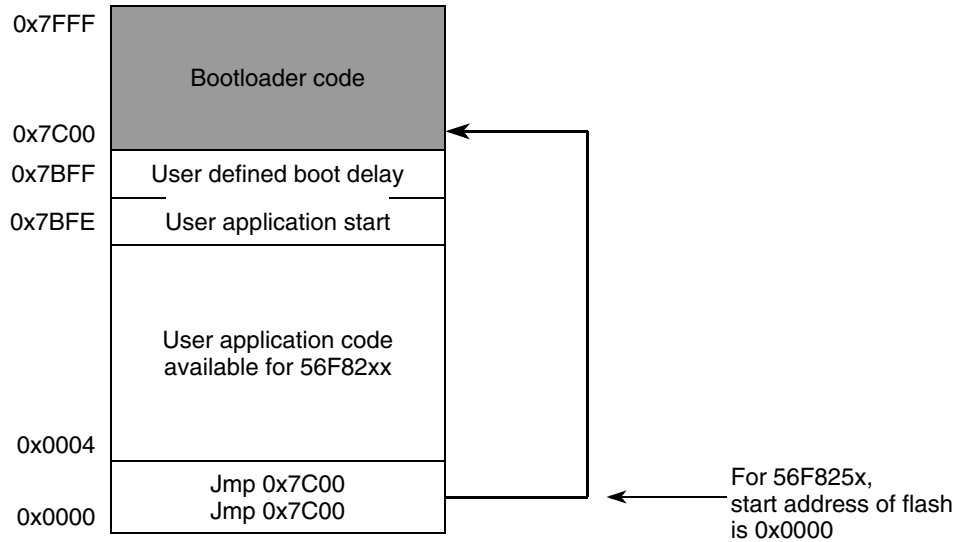


Figure 3. Program Flash Usage for 56F825x Serial Bootloader

## 4 How to make an application downloadable with Static Serial Bootloader

For the application to be loaded by the static serial bootloader the following requirements are necessary:

**If the bootloader needs to start immediately after reset, then a “jump to 0x7C00” instruction must be written into the reset and cop vector. Otherwise, there must be a jump to bootloader in the user code by calling the following instruction:**

```
asm{
    jmp     BOOTLOADER_ADDR
}
```

Where BOOTLOADER\_ADDR is defined **0x7C00**.

Assuming that the ProcessorExpert is used, follow the steps below to make the necessary modification:

1. Generate the application code with ProcessorExpert by clicking “Processor Expert” menu and clicking on “Generate Code” item
2. Open the CPU Component Inspector by clicking “Processor Expert” of the application project window followed by double-clicking the “Cpu:MC56F82xx” icon under “CPUs” group (see [Figure 4](#)), where “MC56F82xx” stands for the specific device part

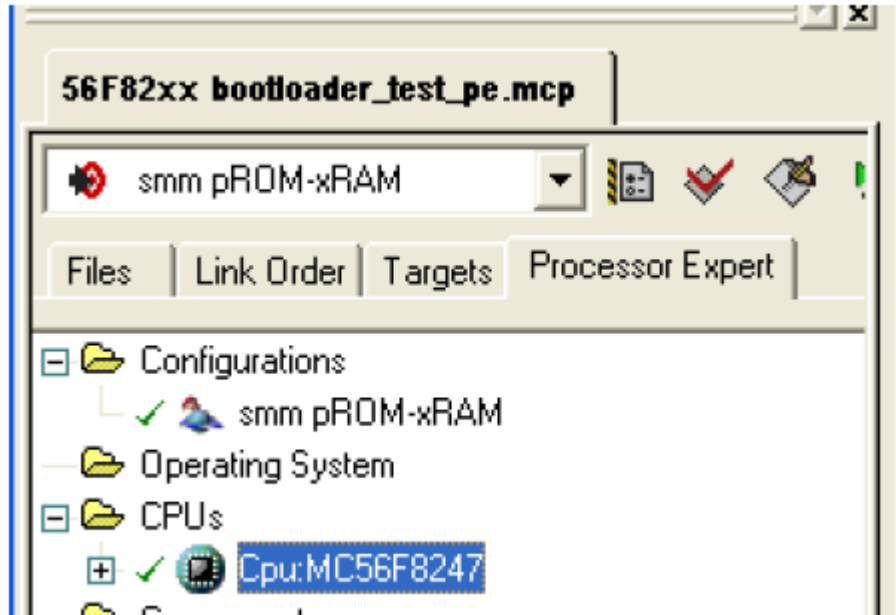


Figure 4. ProcessorExpert tab

3. Click “Build Options” in the CPU Component Inspector and then select “Generate linker file” to “no” (refer to the Figure 5). With this setting, the linker command file will not be regenerated and the changes can be done to the linker command file without being overwritten by the ProcessorExpert.

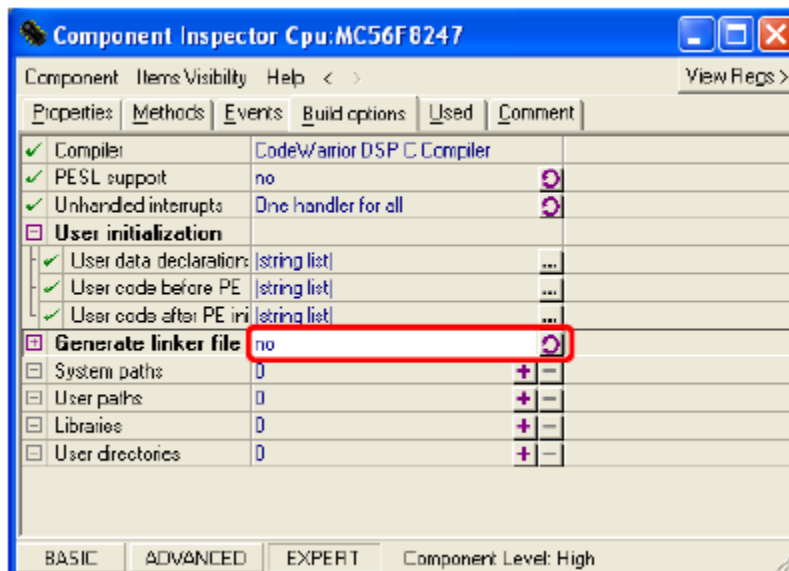


Figure 5. CPU Component Inspector Setting for Generating linker file

4. Open the linker command file and modify it:
  - comment lines related to interrupt\_vectorsboot.text section by following Figure 6, so that the code is not generated there:

```

.p_flash_ROM_data (RX) : ORIGIN = 0x00000001, LENGTH = 0x00000FFF

#Other memory segments
.p_internal_RAM (RVX) : ORIGIN = 0x00008000, LENGTH = 0x1000

# Serial bootloader configuration section
.xBootCfg (RVX) : ORIGIN = 0x00007BFE, LENGTH = 0x2

}

#KEEP_SECTION { interrupt_vectorsboot.text } # comment this line
KEEP_SECTION { interrupt_vectorsboot.text }

SECTIONS {

    .interrupt_vectorsboot :
    {
        F_vector_addr = .;
        * interrupt_vectorsboot.area
    } > .p_interruptsboot

    * (interrupt_vectorsboot.text) # comment this line

    .interrupt_vectors :
    {
        * interrupt_vectors
        * (interrupt_vectors.text)
    } > .p_Interrupts

```

Figure 6. Linker command file modification

5. Write the jump to bootloader code into reset and cop vectors. You have the following one of the two options.
  - Option 1: open the vector.c file and modify the first two lines in interrupt\_vector section to jump to bootloader code:

```

// Define bootloader address
#define BOOTLOADER_ADDR 0x7C00

volatile asm void _vect(void);
#pragma define_section interrupt_vectors "interrupt_vectors.text" RX
#pragma section interrupt_vectors begin
volatile asm void _vect(void) {
    JMP BOOTLOADER_ADDR /* Interrupt no. 0 (Used) - ivINT_Reset */
    JMP BOOTLOADER_ADDR /* Interrupt no. 1 (Used) - ivINT_COPReset */
    JSR Cpu_Interrupt /* Interrupt no. 2 (Unused) - ivINT_Illegal_Instruction */
    JSR Cpu_Interrupt /* Interrupt no. 3 (Unused) - ivINT_SW3 */
    JSR Cpu_Interrupt /* Interrupt no. 4 (Unused) - ivINT_HWStackOverflow */
}

```

Figure 7. Vector.c Modification for Reset and COP Vectors

- Option 2: comment these two lines in vector.c as shown in Figure 8 below:

```
#pragma define_section interrupt_vectors "interrupt_vectors.text" RX
#pragma section interrupt_vectors begin
volatile asm void _vect(void) {
// JMP _EntryPoint          /* Interrupt no. 0 (Used) - ivINT_Reset */
// JMP _EntryPoint          /* Interrupt no. 1 (Used) - ivINT_COPReset */
JSR Cpu_Interrupt          /* Interrupt no. 2 (Unused) - ivINT_Illegal_Instruction */
JSR Cpu_Interrupt          /* Interrupt no. 3 (Unused) - ivINT_SW3 */
JSR Cpu_Interrupt          /* Interrupt no. 4 (Unused) - ivINT_HWStackOverflow */
}
```

Figure 8. Remove Reset and COP Vectors in Vector.c

Also modify the linker command file to put two “JMP to 0x7C00” instructions in .p\_Interrupts section as below:

```
//KEEP_SECTION { interrupt_vectorsboot.text }
KEEP_SECTION { interrupt_vectors.text }

SECTIONS {

//      .interrupt_vectorsboot :
//      {
//          F_vector_addr = .;
//          # interrupt vectors boot area
//          * (interrupt_vectorsboot.text)
//      } > .p_Interruptsboot

.interrupt_vectors :
{
# interrupt vectors
WRITEH(0xE154);          # opcode in hardware reset vector to jump to 0x7C00
WRITEH(0X7C00);
WRITEH(0xE154);          # opcode in COP reset vector to jump to 0x7C00
WRITEH(0X7C00);
* (interrupt_vectors.text)
} > .p_Interrupts
```

Figure 9. Add Opcode in linker command file

**NOTE**

The ProcessorExpert must not overwrite the vector.c file in the subsequent code generation. Otherwise, modify it as mentioned above.

At the address 0x7BFE in program flash, the application entry point must be written and at address 0x7BFF in the program flash, the bootloader delay variable must be written.

Define a new segment .xBootCfg and a new section .ApplicationConfiguration in the linker command file:



```
.xBootCfg (RWX) : ORIGIN = 0x00007BFE, LENGTH = 0x2 #

.ApplicationConfiguration :
{
    # Store the application entry point
    WRITEH(F_EntryPoint);

    #Bootloader start delay in seconds
    WRITEH(10);

} > .xBootCfg
```

**Figure 10. Add .xBootCfg in linker command file**

The VBA register has to be set correctly before the application enables interrupts.

If the interrupts are used in the application, the VBA register has to be restored to the proper value so that the chip can jump to the correct vector when an interrupt occurs.

In the application's main routine, the following sequence must be added before calling PE\_low\_level\_init():

```
extern unsigned int _vba;
INTC_VBA = (unsigned int)&_vba >>8; // restore VBA register to point to user defined vector table
```

**Figure 11. Add VBA register initialization code in main()**

Where **\_vba** variable is defined in the linker command file as below:

```
.interrupt_vectors :
{
    # interrupt vectors
    F_vba = .;
    * (interrupt_vectors.text)
} > .p_Interrupts
```

**Figure 12. Add \_vba definition in linker command file**

## 5 How to Select a Target for the Supported Device

There are two targets in the bootloader project: 56F825x bootloader and 56F824x bootloader. To select the bootloader for MC56F824x, select “56F824x bootloader” target setting as shown below:

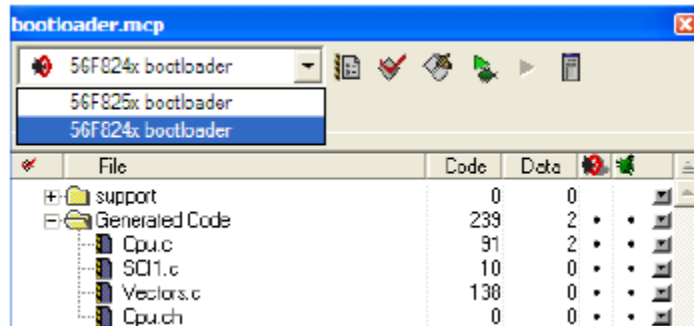


Figure 13. Select a bootloader target

To select the bootloader for MC56F825x, select “56F825x bootloader” target setting accordingly. In addition, the dynamic version of the bootloader is selected by default. If it is necessary to use the static bootloader, remove the comment in the following line in the C/C++ Preprocessor panel of the specified target setting:

```
#define DYNAMIC_BOOT
```

## 6 Serial communication

SCI0 port is used by the serial bootloader. By default, the baud rate of the serial communication is set to 115200 bps and the serial communication protocol is configured as XON/XOFF protocol. To change the default baud rate, simply define SCI\_BAUDRATE\_9600 macro in C/C++ Preprocessor panel of the target setting dialog. In addition, GPIOC2 pin is configured as TXD0 to transmit chars from 56F82xx to the external host, and GPIOC3 pin is configured as RXD0 to receive chars from the external host. To change the default SCI0 pins, either SCI\_GPIOC7\_C3 or SCI\_GPIOC2\_F8 can be defined so that GPIOC7/GPIOC3 or GPIOC2/GPIOF8 are used.

## 7 How to generate a S-record file

Follow [Figure 14](#) below to configure the M56800E Linker options. Do not check “Generate Byte Addresses”. The recommended **Max Record Length** is 40. EOL character must be DOS.

After configuration, click on OK button and then select “Project” menu followed by clicking “Make” item to rebuild the project.

The generate s-record file to be used by the bootloader is the combined p and x s-record file without “.p” or “.x” in the extension name (.s).

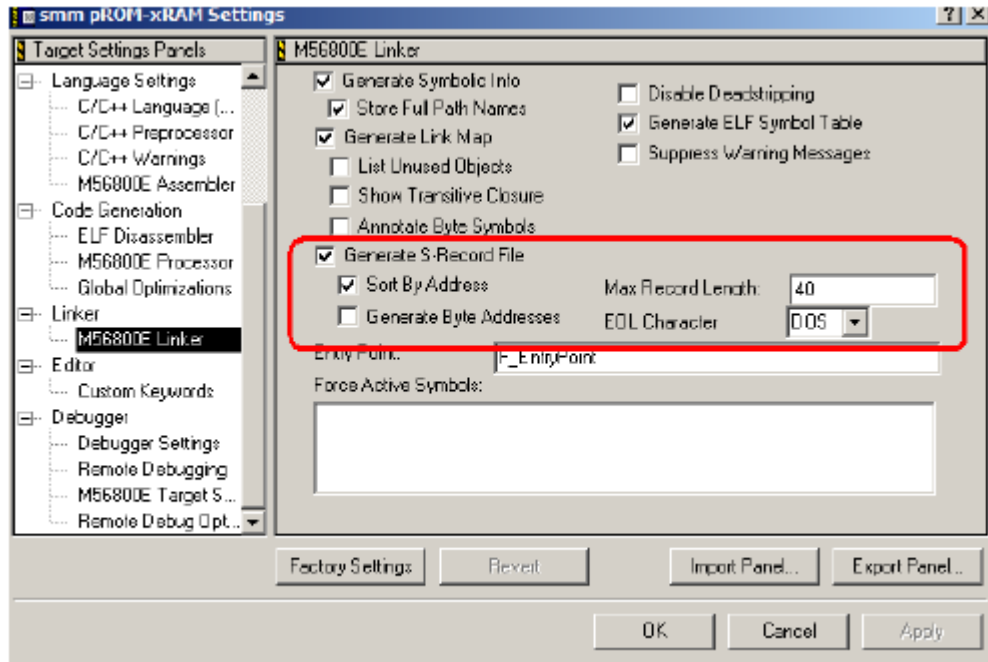


Figure 14. M56800E Linker Settings for Generating S-Record File

## 8 How to download the Bootloader code

There are two connection options: one is the “56800E Local Hardware Connection”, the other is “56800E Local USBTAP Connection”. The first one requires the parallel interface to the PC, and the other one requires USB connection to the PC.

The default setting for the bootloader is “56800E Local USBTAP Connection”. Follow the [Figure 15](#) below to select which one is suitable to your development environment and then click on OK button. Then, select “Project” menu followed by clicking “Debug” menu item to download the bootloader code to the device.

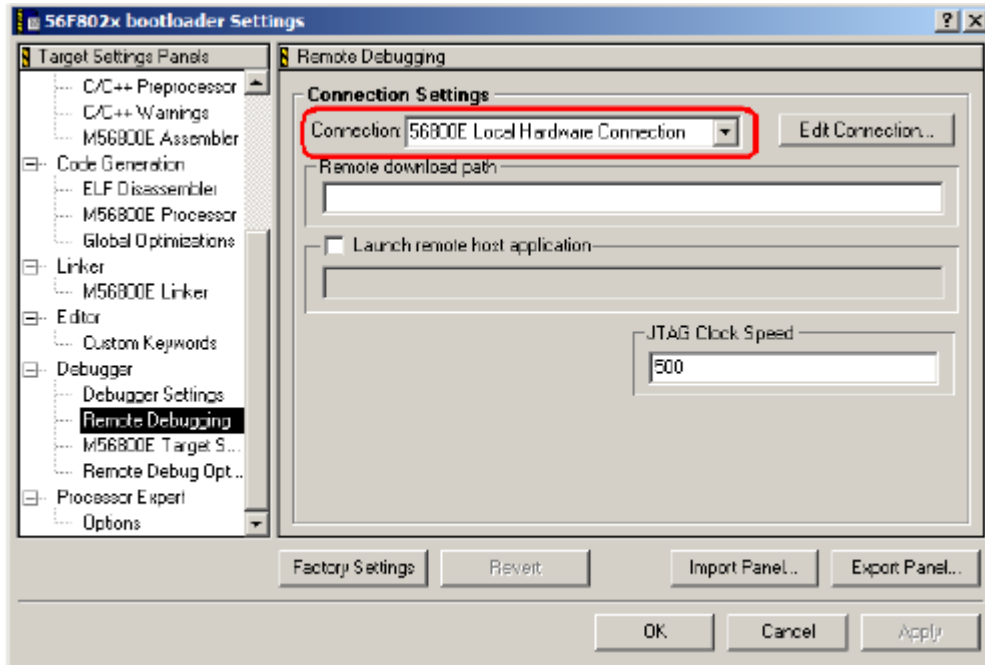


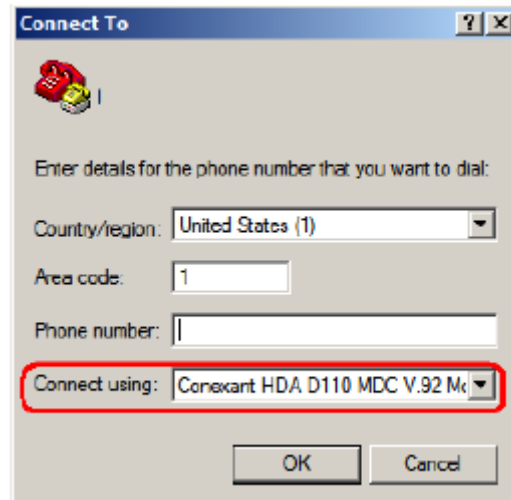
Figure 15. Remote Debugging Settings

## 9 How to download the Application code

Most serial terminal programs can be used to download an S-Record file from a host to a 56F82xx device via the Serial Bootloader. For example, HyperTerminal can be used in Windows platform.

Follow steps below to download the application code with HyperTerminal:

1. Click Windows Start menu
2. Select “**Programs->Accessories->Communication->HyperTerminal**” item
3. Enter a name for the connection and click on “OK” button. It will display the following window (see [Figure 16](#)) where a COM must be selected in “Connect using” dropdown list.



**Figure 16. Connect To dialog**

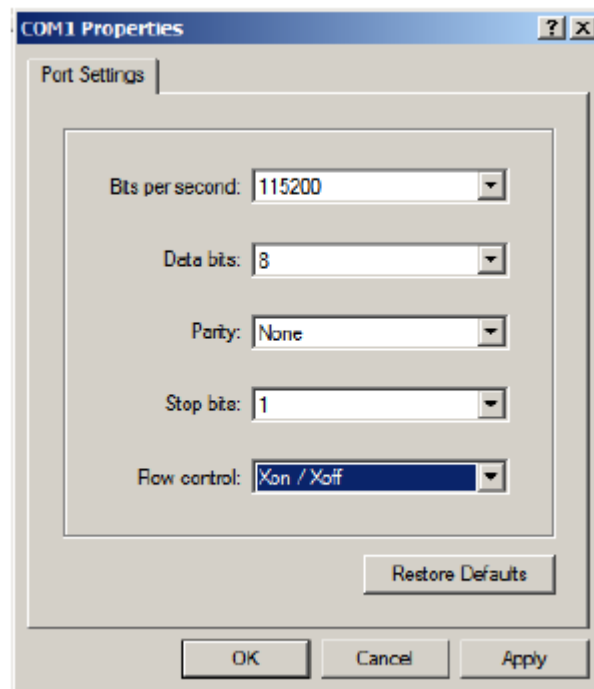
4. Configure COM Properties as shown in [Figure 17](#).

The COM properties must be configured as follows:

**Baud rate:** 115200 bps

**8N1:** 8 data bits, no parity, 1 stop bit character format

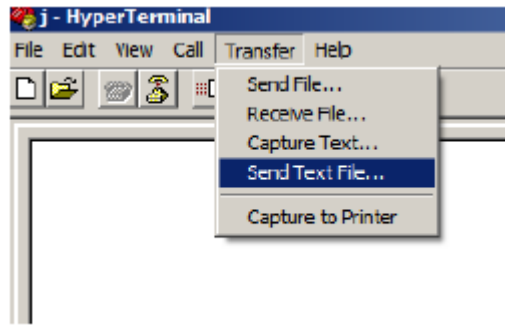
**Flow Control Protocol:** Xon/Xoff



**Figure 17. COM Settings**

**How to download the Application code**

5. Select “Send Text File” when the bootloader startup banner appears:

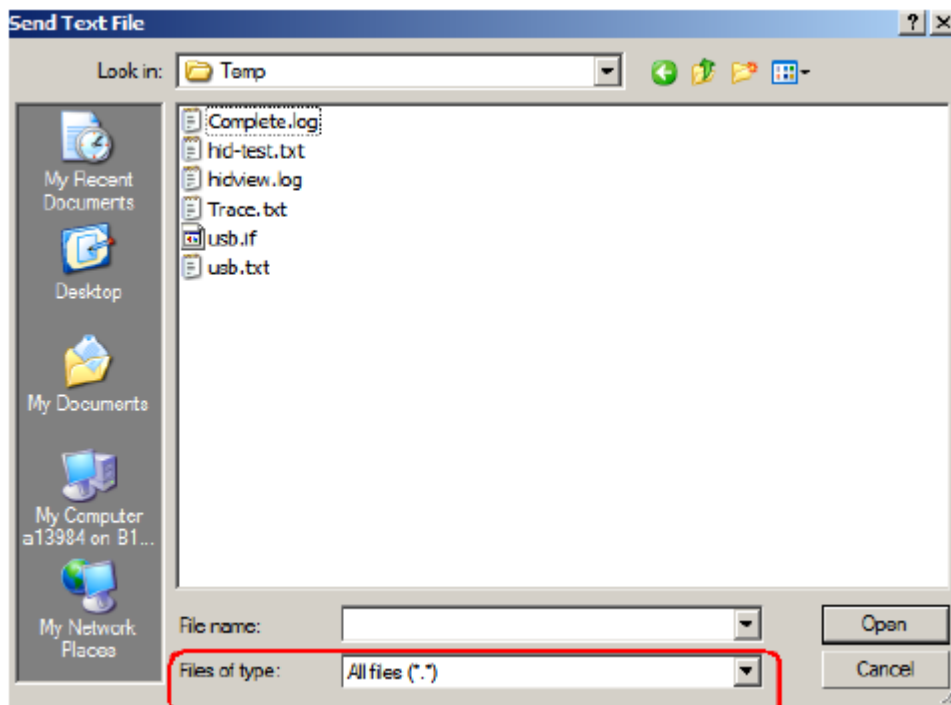


**Figure 18. Send Text File.... Menu item**

6. Choose the application s-record file in Send Text File dialog.

**NOTE**

Change the file type to All files(\*.\*) as below:



**Figure 19. Choose a s-record file**

## 10 Error code

“x” means don’t care.

**Table 1. Error code description**

Error codes	Meaning	Possible Reasons	What to do
xx01	s-record checksum error	s-record file format is incorrect or serial connection does not support XON/XOFF protocol.	Refer to <a href="#">Section 7, “How to generate a S-record file</a> and use the serial connection supporting XON/XOFF protocol
xx02	s-record type error	s-record file format is incorrect or serial connection does not support XON/XOFF protocol.	Refer to <a href="#">Section 7, “How to generate a S-record file</a> and use the serial connection supporting XON/XOFF protocol
xx10	Flash program error	The application entry point written in 0x7BFE is different from the real application start address; or the low voltage of the power supply to the chip.	Make sure flash location in 0x7BFE contains correct application start address; Check the power supply to see if the voltage is below 2.7 V.
xx20	wrong code/data start address	s-record file format is incorrect	Refer to <a href="#">Section 7, “How to generate a S-record file</a>
xx40	SCI receiving error	The host transmits chars too fast than can be handled by the bootloader, which is normally caused by a breakpoint in the bootloader during debugging.	Remove all breakpoints during the bootloader debugging; Lower the baud rate of bootloader and the host terminal.

## 11 Technical support

For any questions/problems, please go to <http://www.freescale.com/support> to raise the request or send the request to [support@freescale.com](mailto:support@freescale.com) or call hotlines below:

Region	Country	Language	Phone Number	Office hours	Time Zone	TimeZone Offset W (S)
North and South America						
	USA	English	1 800 521 6274	8AM-6PM	CT	GMT-6 (-5)
	Mexico	Spanish	00 1800 514 3392	8AM-6PM	CT	GMT-6 (-5)
	Other	Spanish	+52 33 3283-0770	8AM-6PM	CT	GMT-6 (-5)
Europe						
	France	French	+33 1 69 35 48 48	9AM-5PM	CET	GMT+1 (+2)

## Technical support

Germany	German	+49 89 92103 559	9AM-5PM	CET	GMT+1 (+2)
UK	English	+44 1296 380 456	9AM-5PM	CET	GMT+1 (+2)
Asia					
China	Chinese	800 990 8188	9AM-5PM	HKG	GMT+8 (+9)
Hong Kong	English, Chinese	2666 8080	9AM-5PM	HKG	GMT+8 (+9)
India	English	000 800 852 1155	9AM-5PM	HKG	GMT+8 (+9)
Asia Pacific (all others)	English, Chinese	+800 2666 8080	9AM-5PM	HKG	GMT+8 (+9)
Japan	Japanese	120 191 014	8AM-5PM	TKY	GMT+9 (+10)

### Abbreviations:

GMT: Greenwich Mean Time

CT: Central Time

CET: Central European Time

HKG: Hong Kong

TKY: Tokyo

W: Winter

S: Summer (Daylight Saving Time)



**How to Reach Us:****Home Page:**

[www.freescale.com](http://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](http://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](http://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](mailto: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](mailto:support.asia@freescale.com)

**For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
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.

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>.

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. 2011. All rights reserved.