SMBus Bootloader

1. Introduction

A bootloader enables field updates of application firmware. A System Management Bus (SMBus) bootloader enables firmware updates over the SMBus. The SMBus bootloader described in this application note is based on the Silicon Labs Modular Bootloader Framework. This framework is described in detail in the application note “AN533: Modular Bootloader Framework for Silicon Labs C8051Fxxx Microcontrollers”, which can be downloaded from here: http://www.silabs.com/products/mcu/Pages/ApplicationNotes.aspx

The following components are part of the firmware update setup:

- Target Bootloader Firmware
- Master Programmer Firmware
- Active Data Source Software
- Target Application Firmware

The firmware update setup is shown in Figure 1. Details about the steps involved in updating the firmware can be found in the Firmware Update Process Flow Diagram in application note AN533.

The code accompanying this application note is originally written for C8051F37x and C8051F39x devices, but can be ported to other devices in the Silicon Labs microcontroller range.

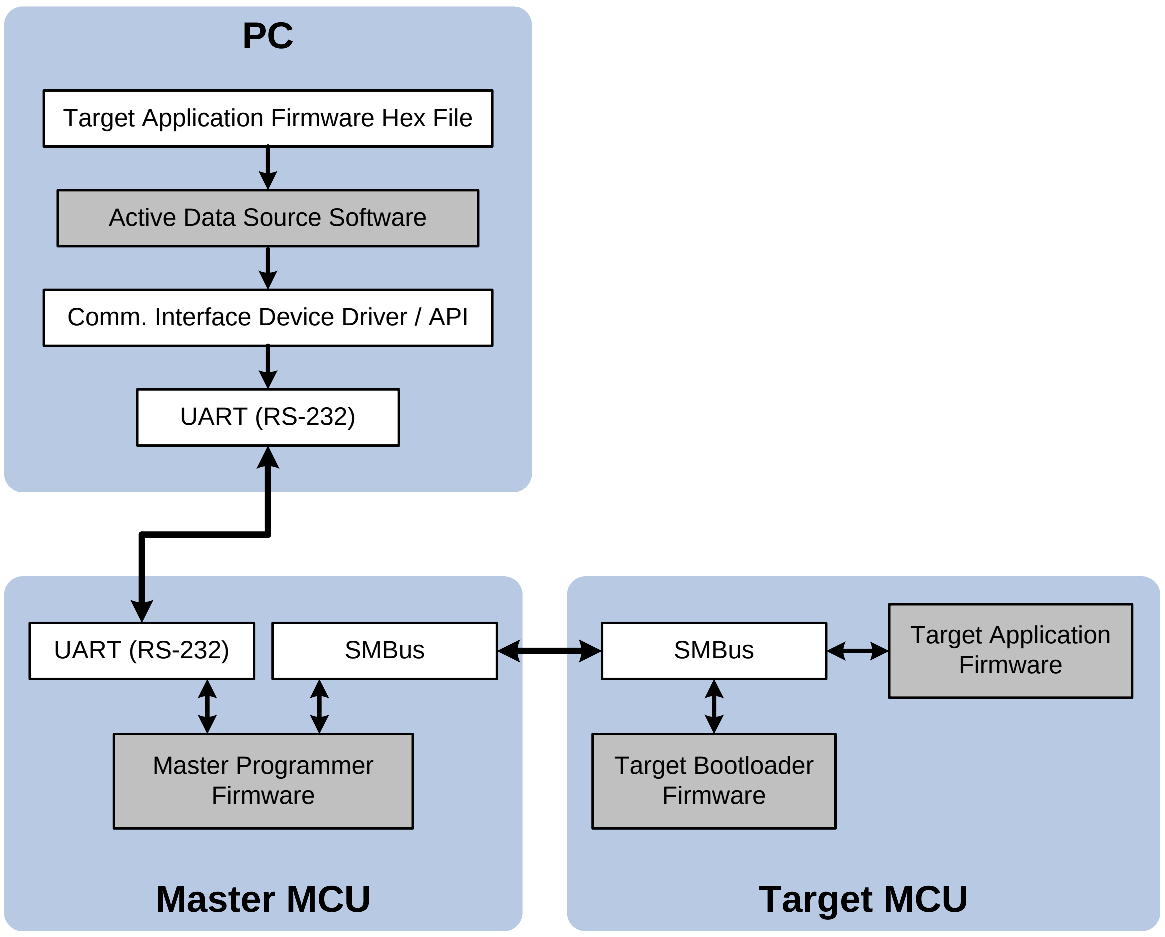

PC
Target Application Firmware Hex File
Active Data Source Software
Comm. Interface Device Driver / API
UART (RS-232)
UART (RS-232)
SMBus
Master Programmer Firmware
Master MCU
SMBus
Target Application Firmware
Target Bootloader Firmware
Target MCU

Figure 1. Firmware Update Setup

2. Target Bootloader Profile

The SMBus target bootloader firmware allows application firmware updates in the field over SMBus. The SMBus bootloader code builds under the Keil toolchain with a total size of 505 bytes. Because the bootloader and application have to be split on page boundaries, the bootloader takes up a total of 1 kB (= two flash pages) code space, with 512 bytes (= one flash page) of that total located on the last flash page, i.e., the one containing the lock byte. This means that the application firmware starts at address 0x0200 and ends one page short of the last flash page. Figure 2 shows the SMBus bootloader memory map. Figure 3 shows the code space utilization of the bootloader grouped by functional blocks.

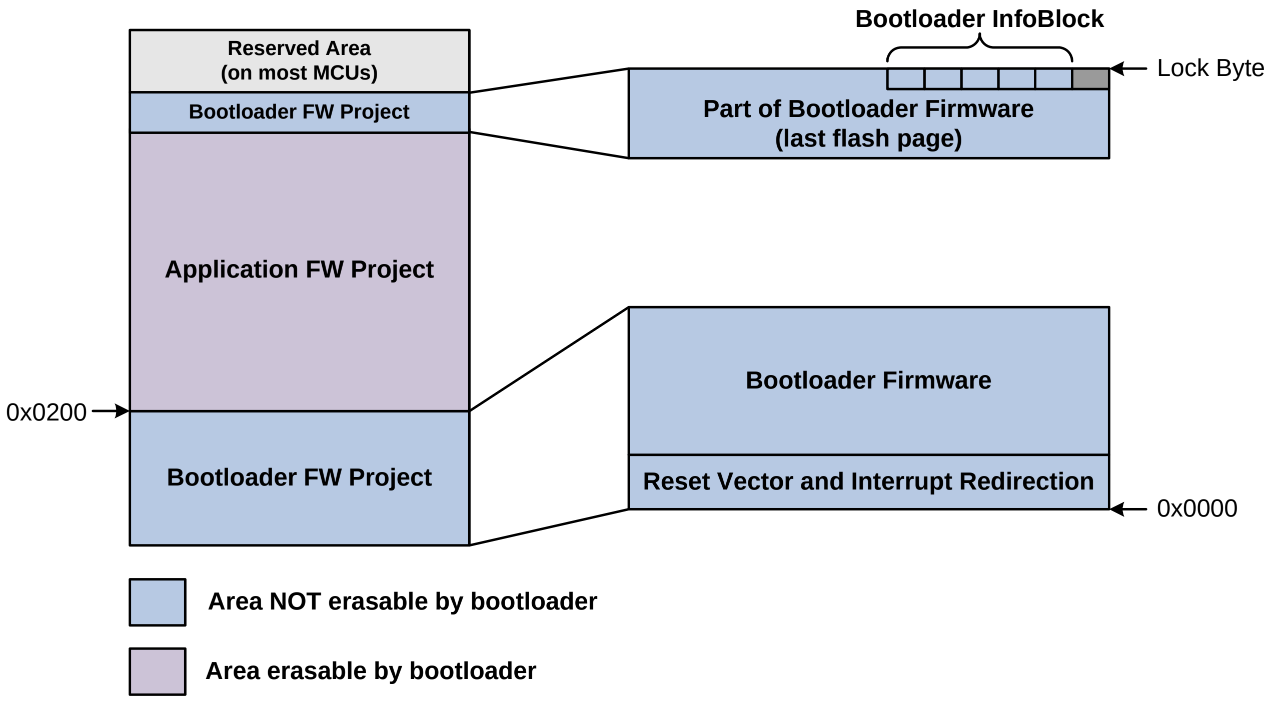

Bootloader InfoBlock
Lock Byte
Reserved Area (on most MCUs)
Bootloader FW Project
Part of Bootloader Firmware (last flash page)
Application FW Project
Bootloader Firmware
0x0200
Bootloader FW Project
Reset Vector and Interrupt Redirection
0x0000
Area NOT erasable by bootloader
Area erasable by bootloader

Figure 2. SMBus Target Bootloader Memory Map

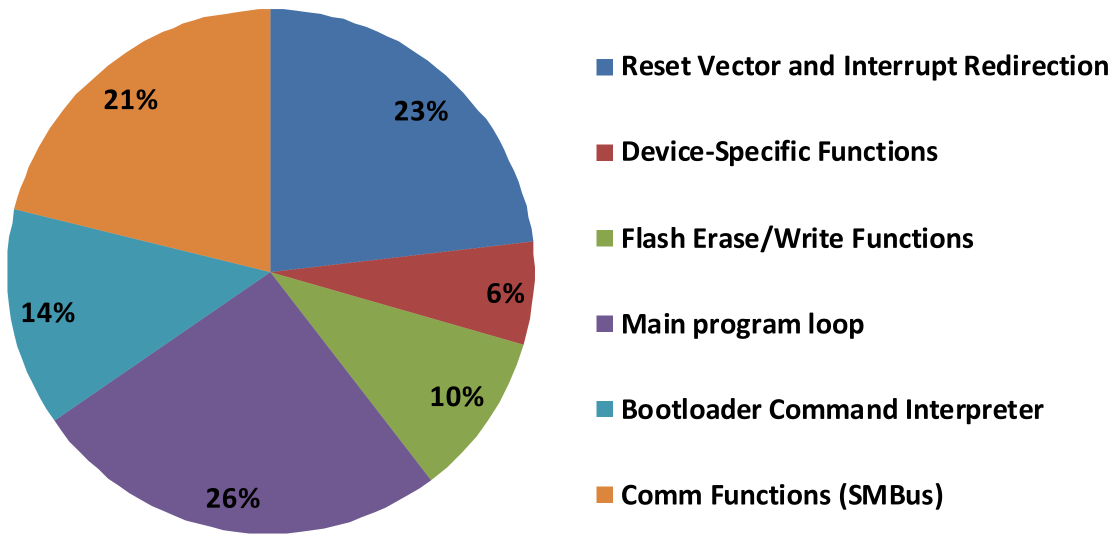


Figure 3. SMBus Target Bootloader Code Space Utilization Profile

2.1. Configurable Options

The target bootloader has the following parameters that can be configured. These parameters are located in two header files as grouped in Table 1 and Table 2.

Table 1. Fxxx_Target_Config.h

Parameter	Options
TGT_PRODUCT_CODE[1]	Any 8-bit value
TGT_BL_TYPE[2]	8-bit value: **0x84**
TGT_FLASH_PAGE_SIZE[3]	Number of bytes per flash page: **512**
TGT_FLASH_PAGE_SIZE_CODE[4]	8-bit value: **9**
APP_FW_START_ADDR[5]	16-bit value: **0x0200**
APP_FW_END_ADDR[6]	16-bit value: **0x3DFF**

Notes:

1. This can be used to identify a product line among many different products.
2. This denotes that the BL uses Silicon Labs-defined SMBus bootloader protocol (see **Fxxx_BL132_SMB0_Interface.h**).
3. Should be changed based on the MCU data sheet.
4. 2^n encoding: 2^9 = 512 bytes.
5. Starting address of App FW.
6. Ending address of App FW (includes App InfoBlock and Signature bytes).

Table 2. Fxxx_TargetBL_Config.h

Parameter	Options
TGT_BL_FW_INFOBLOCK_LENGTH[1]	8-bit value: **16**
TGT_BL_FW_VERSION_LOW and TGT_BL_FW_VERSION_HIGH[2]	8-bit values: **0 and 1**
TGT_BL_BUF_SIZE[3]	Max number of bytes in bootloader receive buffer: **40**
TGT_BL_BUF_SIZE_CODE[3]	8-bit value: **5**

Notes:

1. See Table 1 in AN533.
2. BL v1.0: Low = 0 and High = 1.
3. 2^n encoding: 2^5 = 32 bytes, plus packet header (6 bytes) for a total of 40 bytes.

3. Target Application Profile

The target application firmware needs to fit within the allocated application area in flash memory. The application firmware memory map is shown in Figure 4.

Note: The application firmware starts at address 0x0200 in this example.

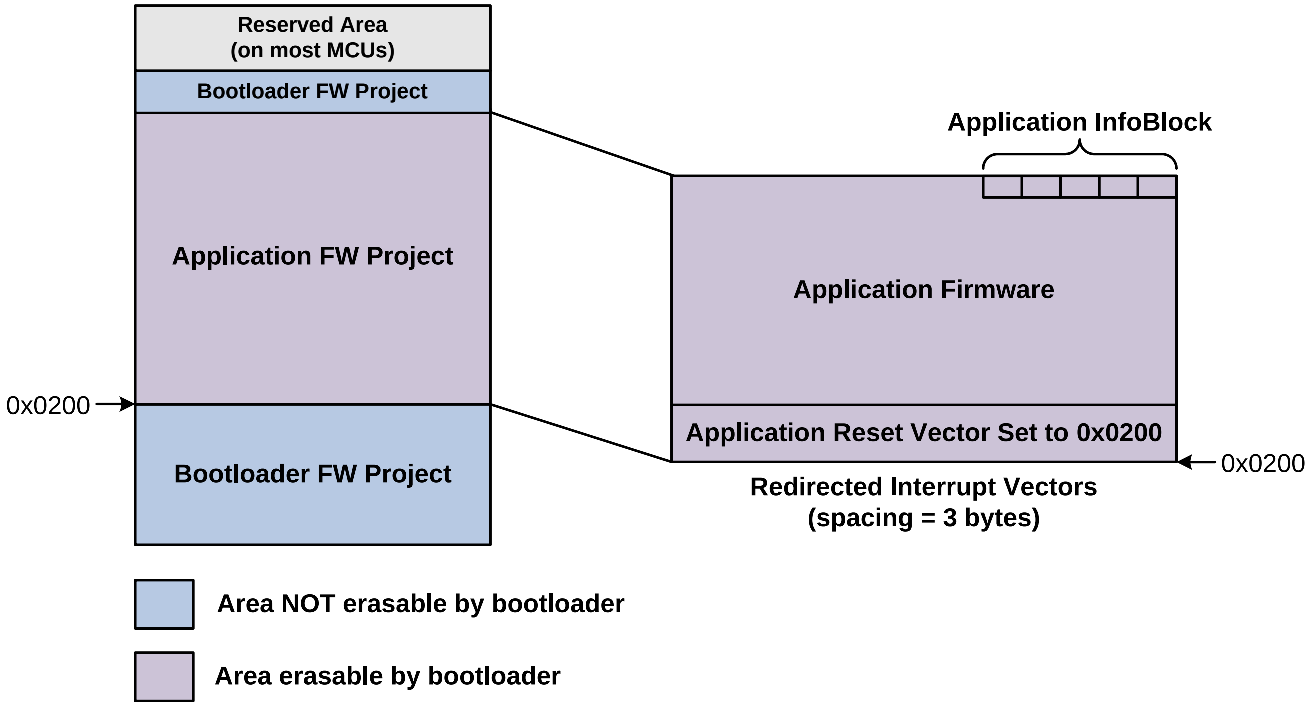

Reserved Area (on most MCUs)
Bootloader FW Project
Application FW Project
0x0200
Bootloader FW Project
Application InfoBlock
Application Firmware
Application Reset Vector Set to 0x0200
0x0200
Redirected Interrupt Vectors (spacing = 3 bytes)
Area NOT erasable by bootloader
Area erasable by bootloader

Figure 4. Target Application Memory Map

3.1. Target Application Template

A target application template is included for easy integration with existing application code or to use as a starting point. This template includes the following files:

- F390_Blinky.c
- STARTUP.A51
- F39x_InfoBlock.c

3.2. Configurable Options

The application firmware should always keep its version number updated in the Application InfoBlock whenever a new version is built so that the application hex file includes this information. The active data source software can interpret this information from the hex.

Table 3. F39x_InfoBlock.c

Parameter	Options
TGT_APP_FW_VERSION_LOW and TGT_APP_FW_VERSION_HIGH[1]	8-bit values: **0 and 1**
TGT_APP_FW_INFOBLOCK_LENGTH[2]	8-bit value: **7**
Notes: **1.** App v1.0: Low = 0 and High = 1. **2.** See Table 5 in AN533.	

3.3. Making an Application Bootloader Aware

A series of simple steps can be used to make an existing application firmware project “bootloader aware”, i.e., allow it to co-exist with the bootloader. These steps are described in detail in AN533. The following is a summary of the changes needed when using the Keil toolchain for the C8051F37x or C8051F39x MCU families:

1. Add **STARTUP.A51** to the application firmware project and build list; this changes the reset vector from 0x0000 to 0x0200.
2. Add these options to the **compiler** command line of the project:

```
INTVECTOR(0x200) INTERVAL(3)
```

3. Add these options to the **linker** command line of the project:

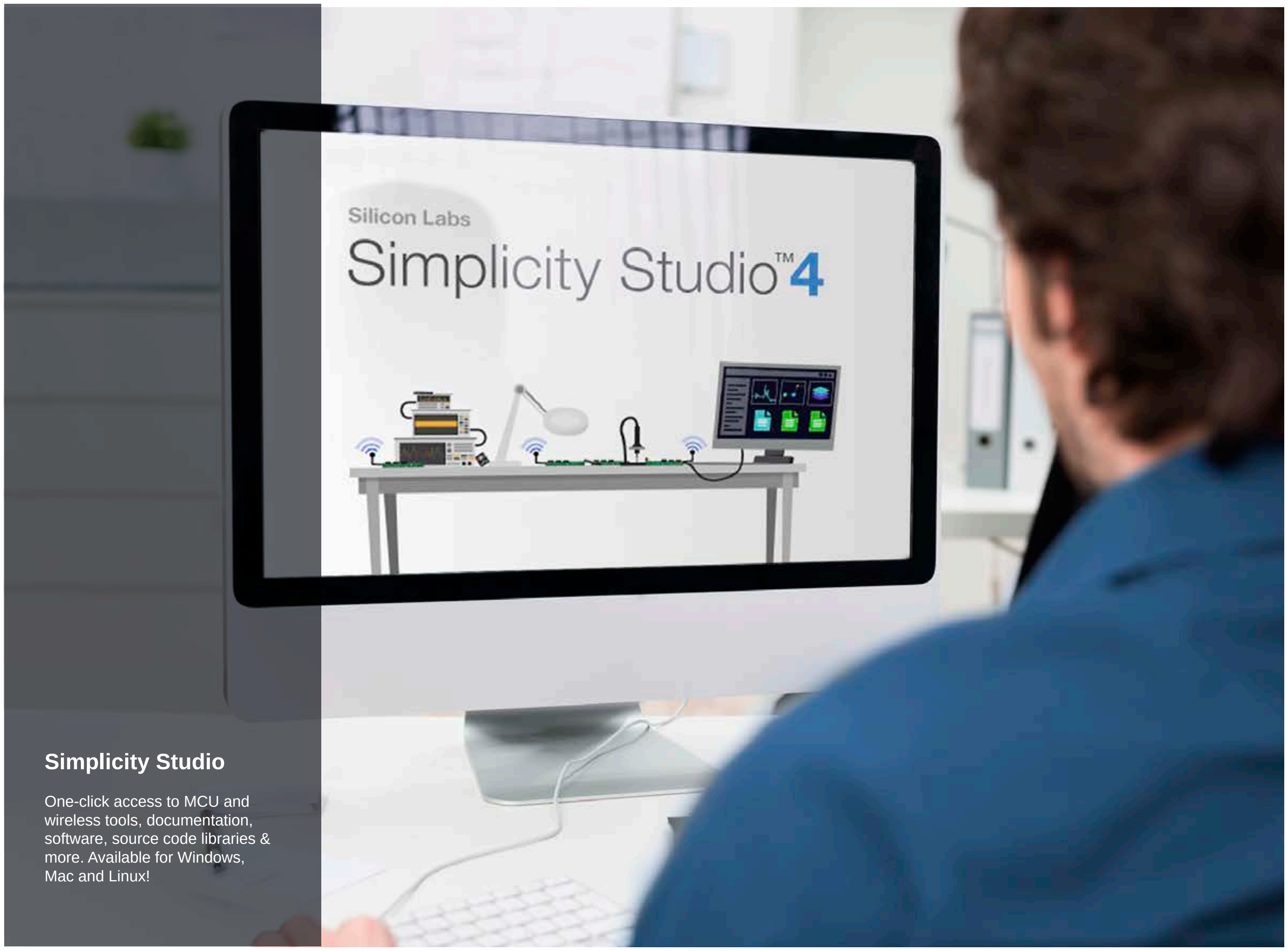
```
CODE(0x200-0x3DFF, ?CO?F39X_INFOBLOCK(0x3DF5))
```

4. Add **F39x_InfoBlock.c** to the application project and build list.
5. (Optional) Add code to the application to recognize the TGT_Enter_BL_Mode command and take appropriate action.
6. Check hardware design to allow the use of a GPIO pin as a fail-safe trigger to enter bootload mode. In the SMBus bootloader example, port pin P1.0 is used for this purpose. To disable or change this, see **Fxxx_TargetBL_Main.c** in the Target Bootloader firmware project. If this is disabled, then the application has to provide some other way of entering bootload mode.

4. Master Programmer and Data Source Examples

A master programmer example that runs on the C8051F390 MCU is included with the SMBus bootloader source code. This master programmer example can update the firmware on any Silicon Labs MCU with a SMBus interface that implements the SMBus protocol detailed in the above sections. This example code can be used as-is or can be used as a reference to implement this functionality on another MCU. The master programmer includes code to communicate with the active data source PC software via the UART.

The Silicon Labs MCU Serial Bootloader Data Source software included with the modular bootloader framework is an example of an active data source software. This is described in application note “AN533: Modular Bootloader Framework for Silicon Labs C8051Fxxx Microcontrollers”. The software source code is included with SMBus bootloader source code.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com