AN11258

I2C secondary boot loader

Rev. 1 — 12 September 2012 | Application note

Document information

Info	Content
Keywords	LPC11xx, LPC17xx, SBL, I2C
Abstract	This application note describes how to implement a Secondary Boot Loader (SBL) using I2C Communication.

Revision history

Rev	Date	Description
1	20120912	Initial version.

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

This document describes functional requirement specifications, implementation and usage of the I2C Secondary Boot Loader (SBL) on NXP's LPC11xx and LPC17xx family.

1.1 Project overview

This project can run on LPC11xx and LPC17xx MCUs. In these MCUs, the primary boot loader resides in the boot block. The primary boot loader is executed every time the part is powered on or reset. It can execute the ISP command handler or the user application code which is stored in sectors of internal flash memory.

The SBL in this project refers to a user-defined application that provides the user with an option to update the user application firmware or execute the previously programmed user application firmware. It is placed from address 0x00 so that when the primary boot loader runs user application, it executes first.

1.2 System components

The system components are shown in Fig 1.

UART communication between the PC and the master MCU is added to give user control over the firmware upgrade process.

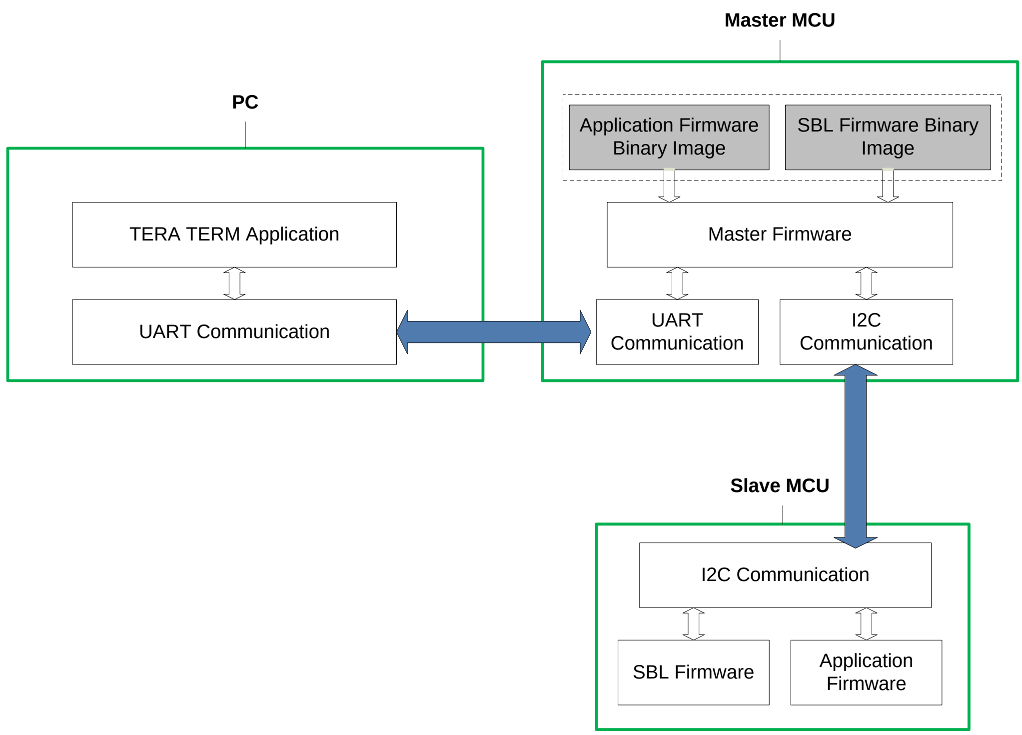

Fig 1. System components

The user interacts with master MCU using Tera Term. The master MCU supports three commands: Read user application firmware ID of slave MCU, Upgrade user application firmware, and Upgrade SBL firmware for slave MCU. Firmware data (user application firmware and SBL firmware) is stored in flash memory of master MCU, from the sector 1 to the last sector.

Once the master MCU receives a command, it communicates to the slave MCU to perform the relevant actions. All transmissions between master MCU and the slave MCU are done using I2C. The protocol in this case is covered in the section 3.

Slave MCU has an SBL located at the first sector of flash memory. At startup, SBL validates user code by checking the firmware version ID (the last 4 bytes of flash memory). If the ID is invalid (0xFFFFFFFF), the slave MCU enters SBL mode and waits for new firmware over I^{2}C-bus. If not, the MCU enters Application mode and executes the Reset Handler of user application firmware at address 0x1004.

In Application Mode, the slave still supports functions to read its user application firmware version ID and upgrade the user application firmware. Moreover, the firmware of SBL can also be upgraded. Using a special command, the user can ask the master MCU to communicate to the slave MCU in order to perform this function.

1.3 I2C communication

A typical I^{2}C-bus configuration is shown in Fig 2. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^{2}C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned.

The master device generates all of the serial clock pulses and the START / STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since a Repeated START condition is also the beginning of the next serial transfer, the I^2C-bus will not be released.

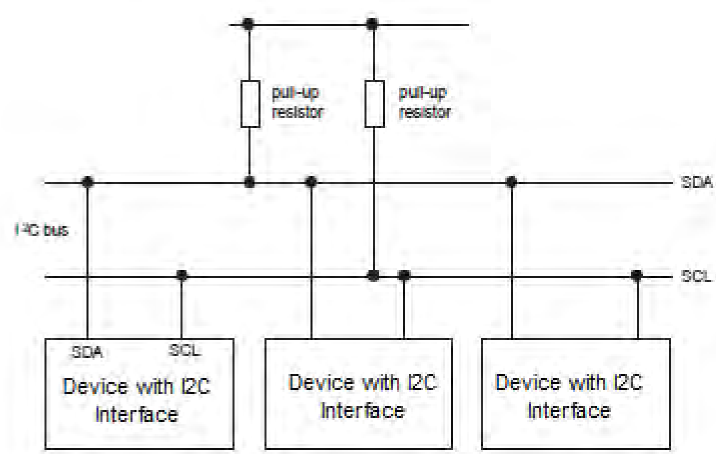

Fig 2. I^2C-bus configuration

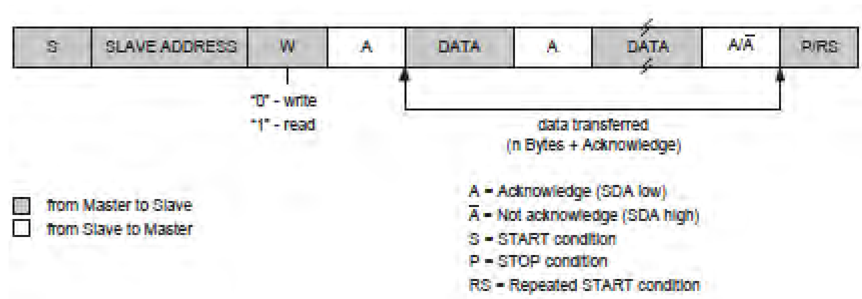

Fig 3. Format in the master transmitter mode (slave receiver mode)

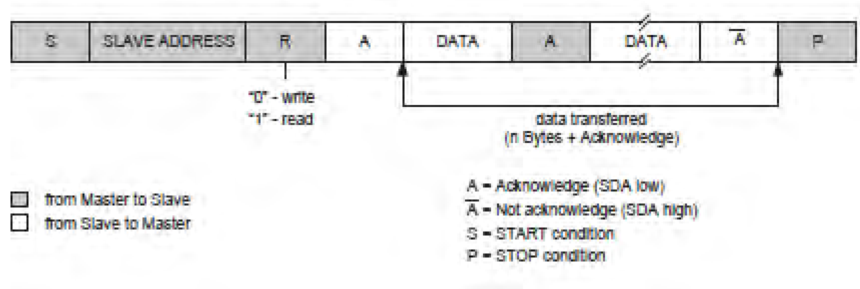

Fig 4. Format in the master receiver mode (slave transmitter mode)

2. System overview

2.1 Project configuration

I2C configuration:

I2C slave Address: 0x60

I2C Clock rate: 1 MHz

UART configuration:

Baud rate: 115200

Data: 8 bits

Stop bits: 1

Parity bit: None

Flow Control: None

2.2 Memory map

2.2.1 Memory map for SBL master MCU

The master's firmware resides on sector 0 of its internal flash memory. The remaining sectors are reserved to store firmware data. When the master MCU has to upgrade SBL firmware or user application firmware, it reads data from these sectors.

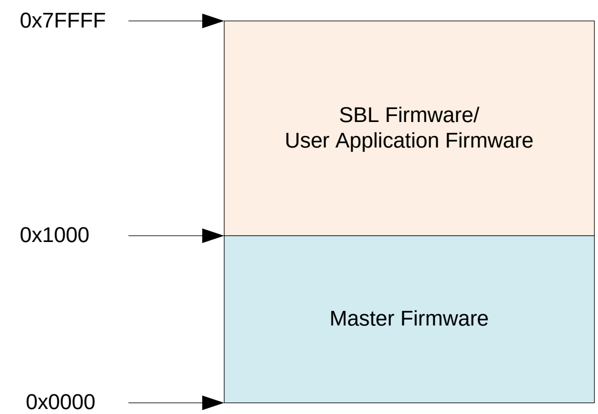

Fig 5. Memory Map for master MCU

2.2.2 Memory map for SBL slave MCU

The flash memory is divided into two regions. One region is for the placement of the user application firmware and the other region is for the placement of SBL. The SBL is placed at the first 4 kB of flash memory and runs first when the system resets.

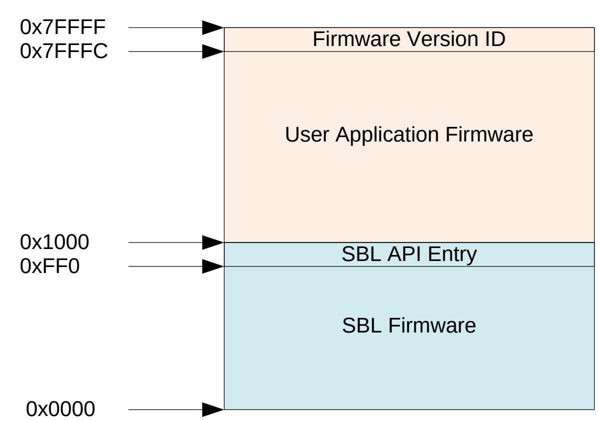

Fig 6. Memory map for slave MCU

As in Fig 6, the Firmware Version ID is stored in the last 4 bytes of flash memory, from 0x7FFFC if LPC17xx MCU is used as slave MCU and 0x7FFC if LPC11xx MCU is used as slave MCU. The SBL also provides an entry point for calling SBL APIs. The address of the entry is from 0xFF0.

3. Functional description

3.1 SBL mode

3.1.1 Read firmware version ID

Function ID: 0x31

Description:

- This function allows the SBL master MCU to read the user application firmware version ID from SBL slave MCU.

Implementation:

- The SBL master waits for a READY byte (0xAA) over I2C.
- The SBL master makes a request to read the user application firmware version ID by issuing a function ID to the SBL slave.
- Upon receiving the request, the SBL slave will prepare the user application firmware version ID information.
- The SBL master will then retrieve the aforesaid firmware version ID information by issuing an SBL read instruction.

Communication protocol:

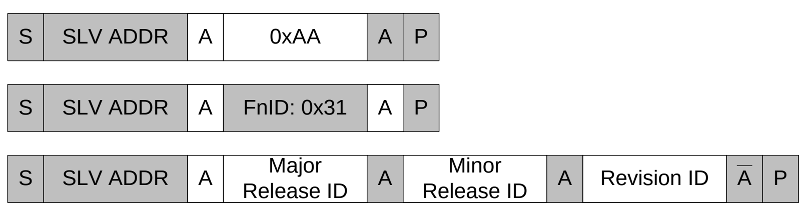

Fig 7. Read Firmware ID

Major release ID:

- Range of ID: 0 - 255

Minor release ID:

- Range of ID: 0 - 255

Revision ID:

- Range of ID: 0, a-f

3.1.2 User application firmware upgrade

Function ID: 0x32

Description:

- This function performs programming of the flash memory by the application program in a running system with an aim to upgrade the user application firmware.

Implementation:

- The SBL master waits for a READY byte (0xAA) over I2C.
- The SBL master makes a request to enter IAP mode by issuing a function ID to the SBL slave.

- The SBL slave erases the last sector of flash memory, which stores the Firmware Version ID information. Then it enters to IAP mode.
- The SBL master transfers firmware data to the SBL slave one block at a time via I2C. Each block of data is 1 kB in size.
- The SBL slave programs the flash one block at a time. The flash on the SBL slave is divided into sectors. Each sector has a size of 4 kB. The SBL slave will always prepare and erase the contents of a flash sector, when it receives first 1 kB block of data for every new sector.
  – Once the first block of data has been programmed, the SBL slave is ready to receive the next block of data.
  – When the second block of data is received, since the contents of entire flash sectors are already cleared, the SBL slave shall directly prepare and program the second block of data at a location that is 1 kB offset from the start of the sector.
- This process continues until the SBL master sends the last block of flash data.
- In order to inform the SBL slave that there are no more data, the SBL master would send Block num (MSB) and Block num (LSB) bytes having a value of 0x00.
- When the IAP process is completed, the SBL slave will execute Reset Handler of user application firmware at address 0x1004.

Communication protocol:

1. The SBL master follows this communication protocol to send the first 1 kB block of data.

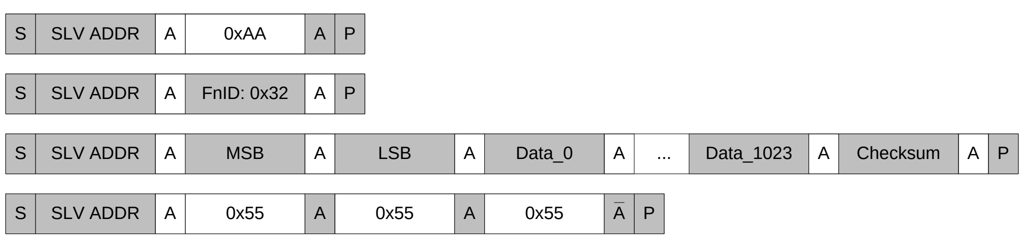

Fig 8. Upgrade the first block of user application firmware

Function ID:

- The function ID used to perform IAP is 0x32. The SBL master is required to send the function ID as the first byte so that the SBL slave can enter IAP mode.

Block num (MSB & LSB)

- Represents the block number that the SBL master is sending.

  E.g. For block number 1: Block num (MSB) – 0x00, Block num (LSB) – 0x01

  For block number 257: Block num (MSB) – 0x01, Block num (LSB) – 0x01

1024 bytes of data

- Represents the 1 kB block of data that is used for programming the flash on the SBL slave.

Checksum byte

- Checksum is used to check the integrity of the SBL transfer.

- Checksum is initialized as 0.
- Computation of Checksum:

  Checksum = 0 - Block num (MSB) - Block num (LSB) - Data_0 - ... - Data_1023

When the SBL master completes the transmission of the block of data, the SBL master will read continuously the status of the Secondary Boot Loader. When 3 bytes of 0x55 are received, the SBL master understands that the SBL slave has received data successfully. If not, the SBL master will resend the block.

SBL status

- If the three SBL status bytes received are '0x55', this means that programming is successful for the 1 kB of data.
- If the three SBL status bytes received are '0xFF', this means that there are errors during programming of the 1 kB of data.

2. The SBL master follows this communication protocol to send the second and subsequent blocks of data.

S	SLV ADDR	A	MSB	A	LSB	A	Data_0	A	...	Data_1023	A	Checksum	A	P
S	SLV ADDR	A	0x55	A	0x55	A	0x55	$\overline{A}$	P					

Fig 9. Upgrade the second and subsequent blocks of user application firmware

Fn ID

- The function ID is not required here since the SBL slave is now in IAP mode.
- The absence of function ID is the only difference in the SBL communication protocol between the transmission of the first and subsequent blocks of data.

3. The SBL master follows this communication protocol to inform the SBL slave that there are no more data to send from the SBL master.

S	SLV ADDR	A	0x00	A	0x00	A	0x00	A	P

Fig 10. Inform that no more data of user application firmware

Block num (MSB) and (LSB)

- Block num (MSB): 0x00
- Block num (LSB): 0x00

Checksum byte

- Generated using Command ID, Block num (MSB) and (LSB)
- Computation: Checksum = 0 - Block num (MSB) - Block num (LSB)

3.1.3 SBL firmware upgrade

Function ID: 0x33

Description:

- This function performs programming of Sector 0 by the application program in a running system.

Implementation:

- The SBL master waits for a READY byte over I2C.
- The SBL master makes a request to program sector 0 of the flash memory by issuing a function ID (0x33) to the SBL slave.
- Following next, the SBL master transfers firmware data to the SBL slave one block at a time via I2C. Each block of data is 1 kB in size.
- The SBL slave will always prepare and erase the contents of a flash sector when it receives first 1kB of data.
- The SBL slave programs the flash one block at a time. (Each sector of the flash memory contains four blocks of 1kB of memory)
- After the SBL master sends the last block of flash data, it would send Block Num (MSB) and Block Num (LSB) bytes having a value of 0x00 to inform the SBL slave MCU that there are no more data.
- When the IAP process is completed, the SBL slave will reset the system.

Communication protocol:

The I2C communication protocol for programming the SBL region on flash memory is similar to the I2C communication protocol for programming the User Application region.

S	SLV ADDR	A	0xAA	A	P											
S	SLV ADDR	A	FnID: 0x33	A	P											
S	SLV ADDR	A	FnID: 0x33	A	MSB	A	LSB	A	Data_0	A	...	Data_1023	A	Checksum	A	P
S	SLV ADDR	A	0x55	A	0x55	A	0x55	$\overline{A}$	P							

Fig 11. Upgrade the first block of SBL firmware

S	SLV ADDR	A	FnID: 0x33	A	MSB	A	LSB	A	Data_0	A	...	Data_1023	A	Checksum	A	P
S	SLV ADDR	A	0x55	A	0x55	A	0x55	$\overline{A}$	P							

Fig 12. Upgrade the second and subsequent blocks of SBL firmware

3.2 Fail-safe implementation

The fail-safe mechanism is to ensure that in the event of loss of power during In-Application Programming, the MCU is able to re-start and complete the IAP process when the power resumes.

Implementation:

There are three important implementations in achieving the fail-safe mechanism for IAP.

1. Have a main function that decides if the flow of execution should go to the User Application or to the In-Application Programming Process.

2. Create and embed a user application firmware ID at an address located at the end of the flash memory.
   – The Firmware ID is used as a determinant to identify if the User Application Code is intact. If the check is successful, it will enter the User Application. Otherwise, it enters in IAP mode to program the user application firmware.
   – This Firmware ID is located at the end of the flash memory to ensure that this data will the last to be programmed during IAP process. In the event when the Firmware ID is invalid (0xFFFFFFFF), this means that the User Application Code has defects. One reason for this is that the previous IAP process is not successful.
   – In the main function, the MCU will always try to enter IAP mode if the Firmware ID is not valid.
3. The first task into the IAP mode is to erase the Firmware ID the last sector to commence the IAP process.

4. Implementation

4.1 Top-level flow diagram

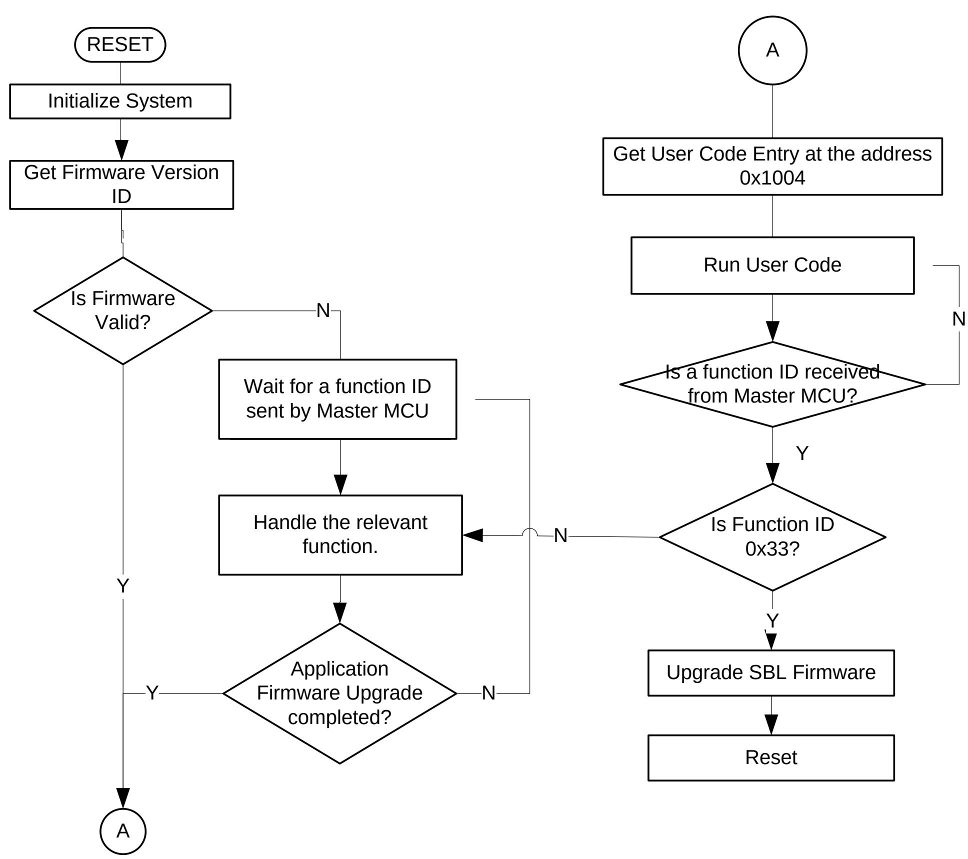

Fig 13. Top-level flow diagram of slave MCU

4.2 SBL mode

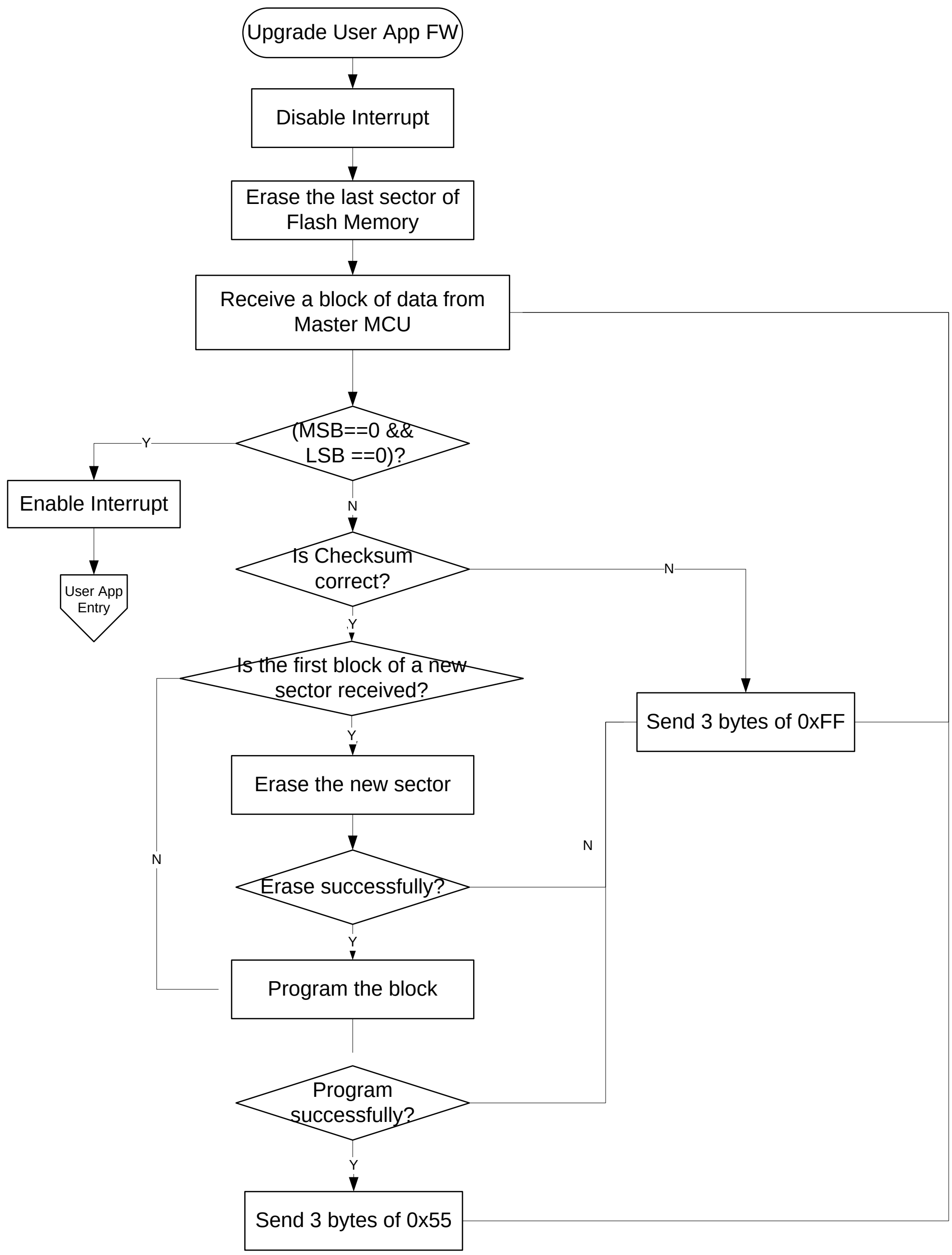

Fig 14. Flow diagram for firmware upgrade

4.3 Application mode

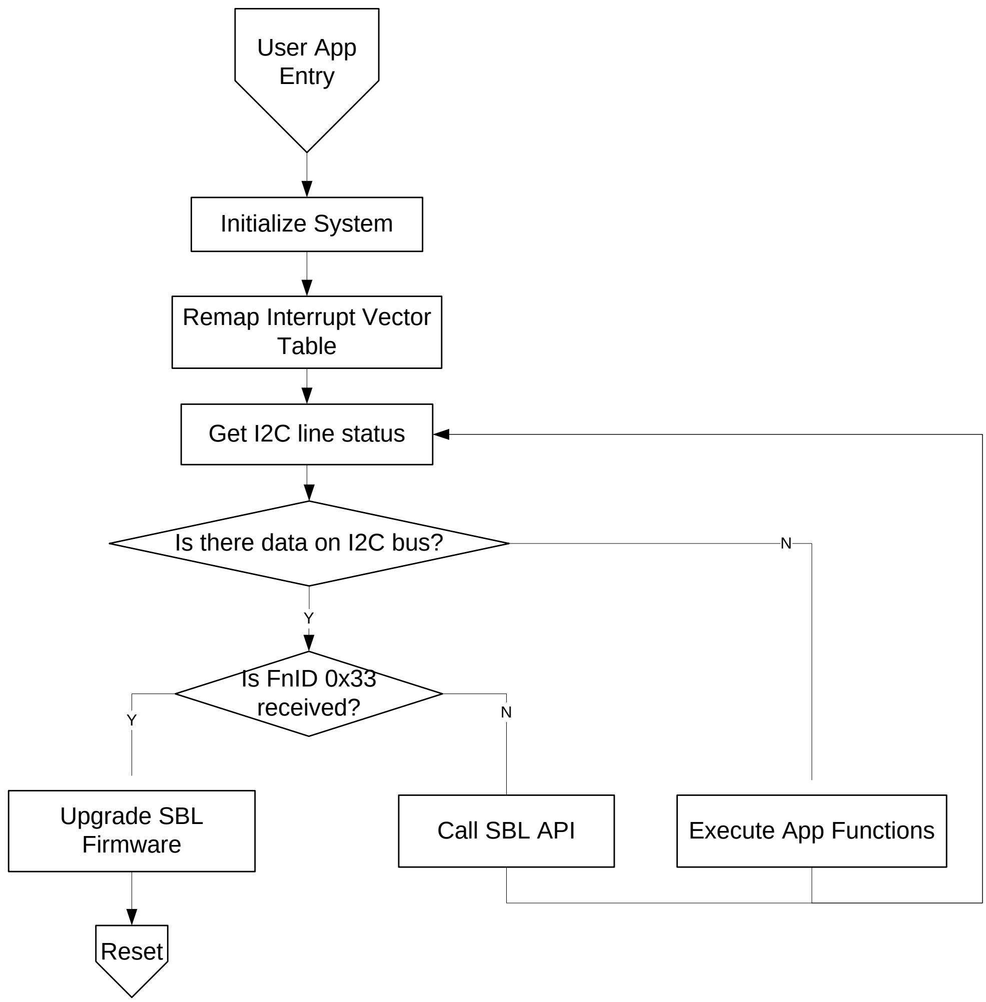

Fig 15. Flow diagram for user application

The SBL function could be called in the following way using C.

```
#define SBL_LOCATION 0xFF1
typedef void (*sbl_api)(uint32_t API_Code, uint8_t* pParam);
```

Create function pointer and call it:

```
sbl_api entry;
entry = (sbl_api) (SBL_LOCATION);
(entry)(API_Code, Params);
```

The parameters will be passed as shown in Table 1:

Table 1. SBL APIs

API_Code	Params	Description
0x01	None	Enter SBL Mode.
0x02	None	Exit SBL Mode.
0x03	Function ID (Size = 1 byte)	Handle command received over I^2C-bus.

4.4 Scatter loading description file

To support In-Application Programming, a scatter loading description file is used to specify the memory map of the image to the linker, in order to control the grouping and placement of the image components.

This is important, because the new image is placed at its intended location and not any other location that can corrupt the codes used to perform IAP.

4.4.1 Scatter loading description file for SBL

The SBL resides on the sector 0 of the slave MCU. It is divided into two regions: one for storing the firmware itself, the other is used to public the SBL API for User Application.

Table 2. Scatter loading description file for SBL (LPC17xx)

Load Region	Execution Region	Input Sections
0x00000000 - 0x00000FEF	0x00000000 – 0x00000FEF	Interrupt Vector Table, Code and RO data.
	0x10000000 – 0x10007BFF	RW data, ZI data.
	0x2007C000 – 0x20083FFF	RW data, ZI data.
0x00000FF0 - 0x00000FFF	0x00000FF0 - 0x00000FFF	SBL API.

Table 3. Scatter loading description file for SBL (LPC11xx)

Load Region	Execution Region	Input Sections
0x00000000 - 0x00000FEF	0x00000000 - 0x00000FEF	Interrupt Vector Table, Code and RO data.
	0x10000000 – 0x10001BFF	RW data, ZI data.
0x00000FF0 - 0x00000FFF	0x00000FF0 - 0x00000FFF	SBL API.

The last 1 kB of internal RAM, starting from 0x10007C00 for the LPC17xx and from 0x10001C00 for the LPC11xx, is used to store 1 kB of firmware data sent by the master MCU while upgrading firmware for the User Application.

4.4.2 Scatter loading description file for user application

The User Application resides in flash memory, from sector 1 to the last sector of the slave MCU. It is also divided into 2 regions; 1 for storing firmware data itself and the other for storing Firmware Version ID.

Table 4. Scatter loading description file for user application (LPC17xx)

Load Region	Execution Region	Input Sections
0x00001000 - 0x0007FFFB	0x00001000 – 0x0007FFFB	Interrupt Vector Table, Code and RO data.
	0x10000000 – 0x10007BFF	RW data, ZI data.
	0x2007C000 – 0x20083FFF	RW data, ZI data.
0x0007FFFC- 0x0007FFFF	0x00000FF0 - 0x00000FFF	Firmware Version ID

Table 5. Scatter loading description file for user application (LPC11xx)

Load Region	Execution Region	Input Sections
0x00001000 - 0x00007FFB	0x00000000 – 0x00007FFB	Interrupt Vector Table, Code and RO data.
	0x10000000 – 0x10001BFF	RW data, ZI data.
0x00007FFC- 0x00007FFF	0x00007FFC- 0x00007FFF	Firmware Version ID

The last 1 kB of internal RAM is also used to store 1 kB firmware data sent by master MCU while upgrading firmware for SBL.

5. How to run

This project is tested on Keil's MCB1700 with LPC1768 version 1 and Keil's MCB1000 with LPC1114. It requires two boards: one master and one slave.

5.1 Directory contents

The folder structure of the project is described in Fig 16.

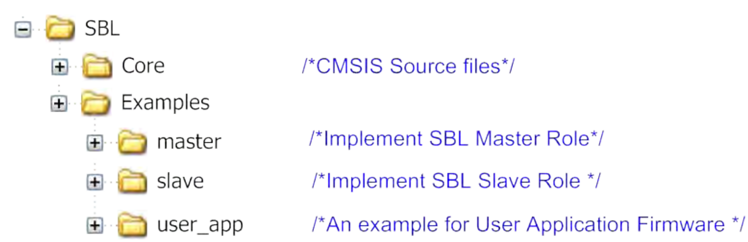

Fig 16. Project directory contents

In each example folder, there are four sub-folders:

- Folder "Keil": includes Keil project file, scatter file and debug file. After compiling, output files would be placed in a sub-folder of this folder. The name of the sub-folder is the same to the name of selected target.
- Folder "drivers": includes source code for controlling peripherals including I2C, UART, and GPIO.
- Folder "sbl": includes SBL source code (Master/slave role).
- Folder "main": includes main program.

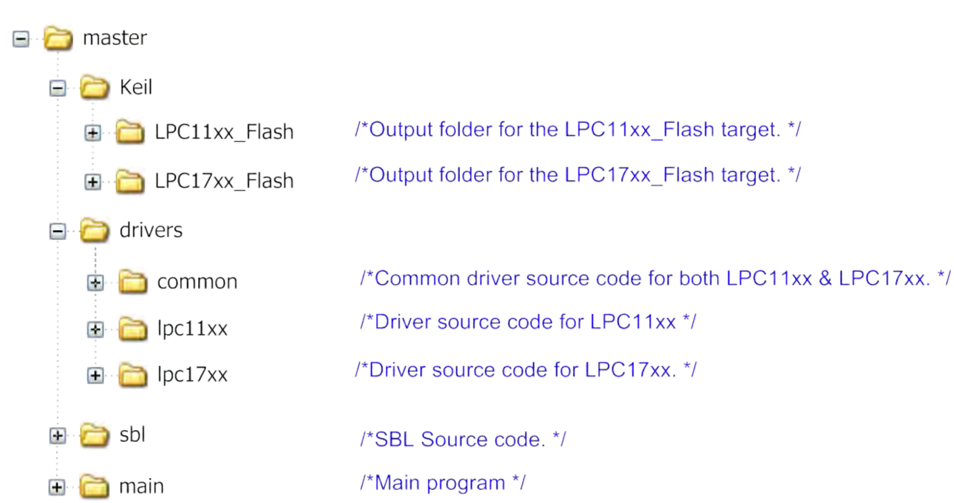

Fig 17. Example directory contents

5.2 Hardware configuration

5.2.1 MCB1700

These jumpers must be configured as:

- VDDIO: ON
- VDDREGS: ON
- Remaining jumpers: OFF

5.2.2 MCB1000

These jumpers must be configured as following:

- J2 – VDD: ON

5.2.3 I2C connection

I2C pins:

Function	MCB1700	MCB1000
SCL	P0.28	P0.4
SDA	P0.27	P0.5

Common ground must be connected together between two boards.

There must be a pull-up resistor on SDA and a pull-up resistor on SCL line. Refer to Fig 2 “I2C Configuration” for more detail.

5.3 Software configuration

5.3.1 Tera Term setting

Serial display configuration

- 115200 bps
- 8 data bit
- No parity
- 1 stop bit
- No flow control

5.4 Steps to run

Step 1: Open SBL master project in **SBL\Examples\master** folder.

- Select the target relevant to the master board.
- Determine the size of User application firmware using SBL_FW_DATA_BLOCK_NUM macro in file “sbl.h”.
- Build SBL master project.
- Burn hex file into master board. (If run on ROM mode)

Step 2: Open SBL slave Project in **SBL\Examples\slave** folder and User Application project in **SBL\Examples\user_app** folder.

- Select the target which relevant to the slave board.
- Build these projects.
- Erase all sectors on flash memory of slave board.

- Burn the hex file of SBL slave project into slave board.
- Burn the hex file of User Application project to master board.

Step 3: Connect UART0 on master board to COM port on your computer.

Step 4: Configure hardware, connect master board and slave board as mentioned in section 5.2.

Step 5: Configure serial display as mentioned in section 5.3.

Step 6: Reset slave board. The LED at P2.2 is lighted on to notify that the system is in SBL mode.

Step 7: Reset the master board. The welcome message is shown.

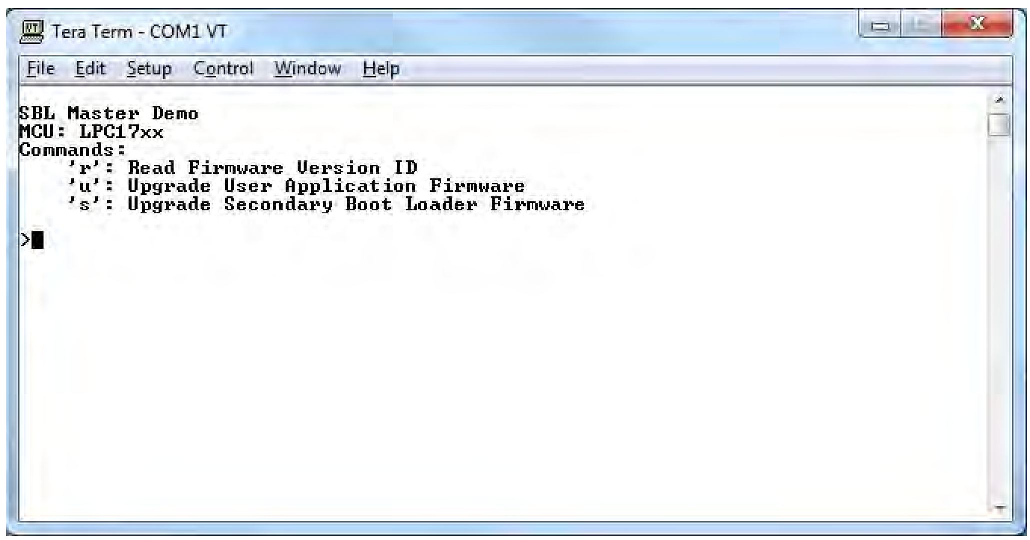

Fig 18. Terminal output when master board resets

- Press ‘r’ to read the user application firmware version ID from slave. If this is the first time upgrading firmware, 0xFF will be displayed for all version details.
- Press ‘u’ to start upgrading slave firmware. The master gets firmware data from sector 1 of its flash memory, and then sends to the slave. The firmware data size is determined by the SBL_FW_DATA_BLOCK_NUM macro.

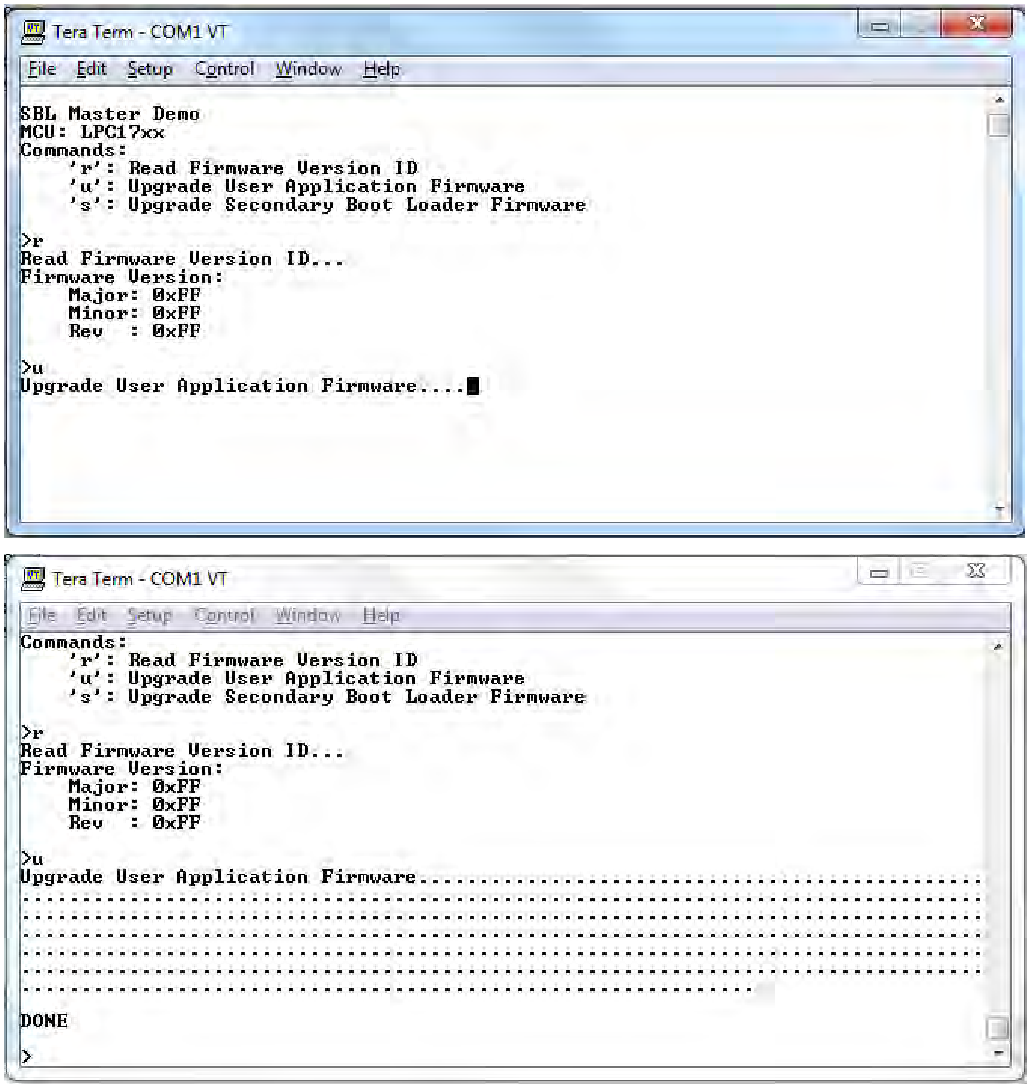

Fig 19. Upgrade firmware

- After upgrading ends, the user code is executed. LED at P2.2 is OFF. LEDs at P2.3~P2.6 blink.
- Press ‘r’ to display version of the new firmware.

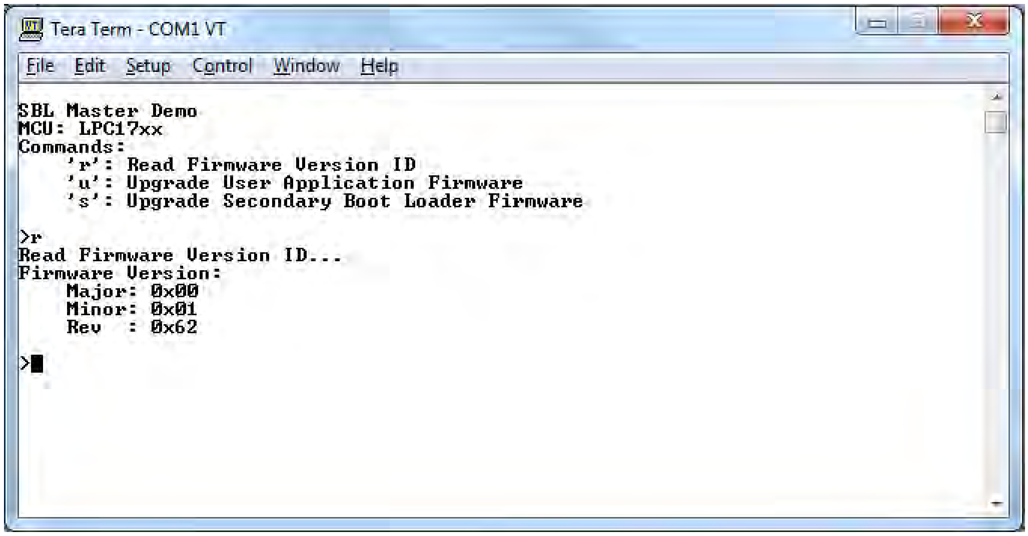

Fig 20. Read new firmware version ID

- To upgrade SBL firmware:
  – First, download the binary firmware file from the SBL slave project (stored in the path **SBL\Examples\slave\Keil\<Target Name>**) to sector 1 of flash memory in master MCU:
    - Open SBL slave Project.
    - Open Target Option Dialog, select Utilities Tab.
    - Choose External Tool for Flash Programming as below.

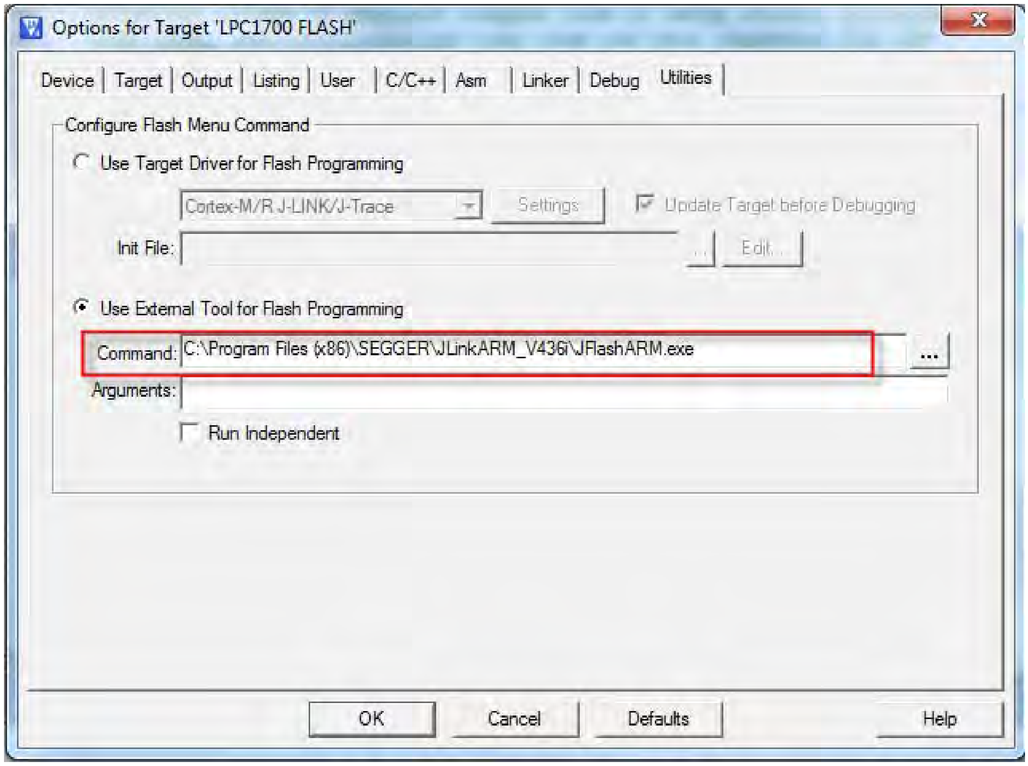

Fig 21. Select Flash Programming Tool for SBL Project. (1st step)

- Select Flash→ Download. The J-Flash ARM Application is opened.
- Select File→ Open Project; browse to file “LPC1768.jflash” if the master is LPC17xx MCU. If it is LPC11xx MCU, please select “LPC1114.jflash”.

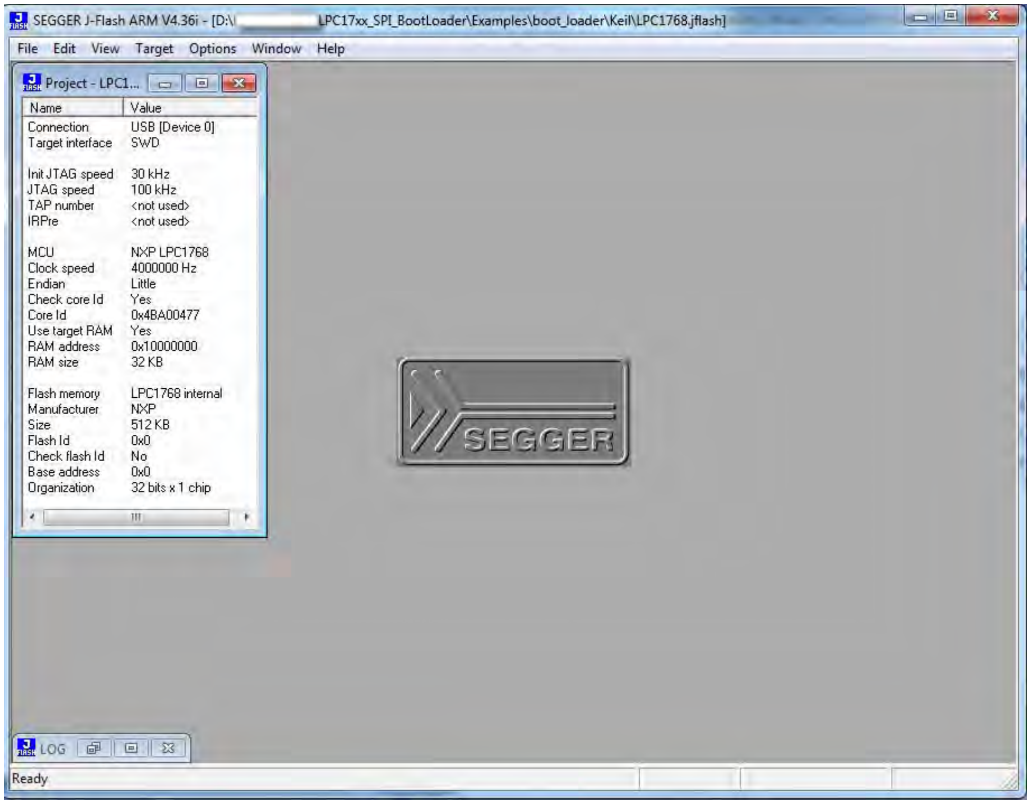

Fig 22. Select Flash Programming Tool for SBL Project. (2nd step)

- Select File→Open data file, browse to sbl.bin file in the relevant output folder of SBL Project. Set the Start Address as 0x1000.

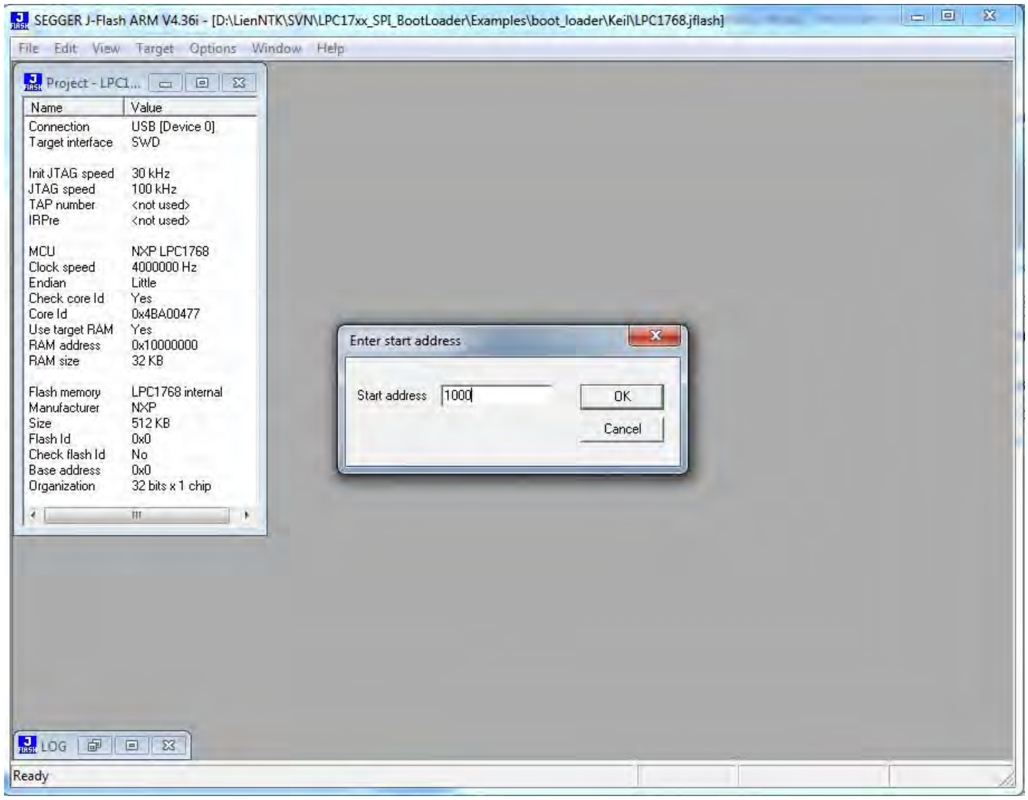

Fig 23. Select Flash Programming Tool for SBL Project. (3rd step)

– Select Target→Program to program the master board.

– Reset the master board.

- Press ‘s’ to upgrade firmware for SBL.

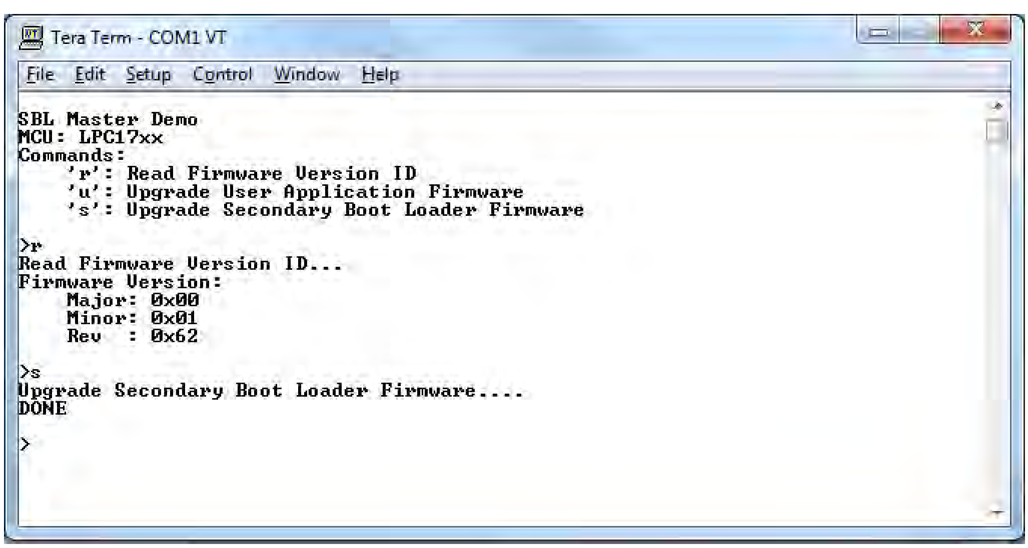

Fig 24. Upgrade SBL firmware

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. List of figures

8. List of tables

9. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12 September 2012
Document identifier: AN11258