AN10866

LPC1700 secondary USB bootloader

Rev. 2 — 21 September 2010 Application note

Document information

Info	**Content**
Keywords	LPC1700, Secondary USB Bootloader, ISP, IAP
Abstract	This application note describes how to add a custom secondary USB bootloader to a LPC1700 series microcontroller.

Revision history

Rev	Date	Description
v.2	20100921	• Updated on how the USB Bootloader executes the user application. • Updated bootloader and user application location figure. • Added the CRP define statement for the “Asm” target options. • Removed the CRP binary/hex comparison section.
v.1	20090908	• Initial version.

Contact information

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

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

1. Introduction

NXP's LPC1700 microcontrollers provide the user a convenient way to update the flash contents in the field for bug fixes or product updates. This can be achieved using the following two methods:

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and UART0 serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.

A secondary bootloader is a piece of code which allows a user application code to be downloaded using alternative channels other than the standard UART0 used by the primary bootloader (on-chip). The primary bootloader is the firmware that resides in a microcontroller's boot ROM block and is executed on power-up and resets. After the boot ROM's execution the secondary bootloader is executed. The secondary bootloader in turn will then execute the end-user application.

This application note uses USB as an example for developing a secondary bootloader on a LPC1700 series microcontroller.

2. Requirements

2.1 Hardware requirements

The secondary USB Bootloader application has been developed and tested using a:

- Keil's MCB1700 development board
- MS Windows-based workstation with an available USB port.

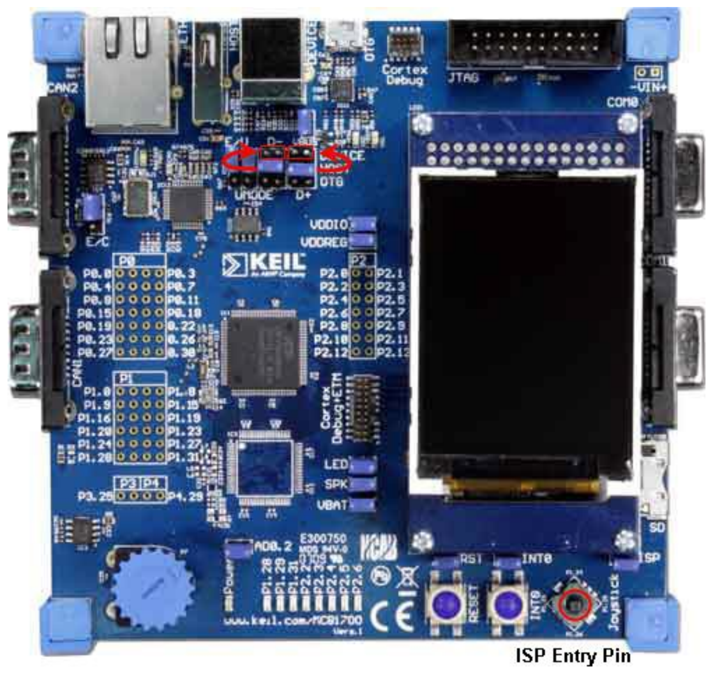


Fig 1. MCB1700 featuring a LPC176x series microcontroller

2.1.1 Programming

Programming the bootloader can be done using Keil's U-LINK JTAG/SWD module. However, as an additional option the LPC1700's ISP programming functionality can be used instead. To program using the ISP protocol an RS-232 serial cable is required and a free ISP programming tool is available from FlashMagic at: http://www.nxp.com/flashmagictool.com/

2.1.2 Jumper settings

In order to use the USB device functionality, ensure that the USB's D+ and D- jumpers are both set to the "DEVICE" header pins as shown in Fig 1.

2.1.3 USB cable

A USB-A to USB-B cable is required to power and connect the windows workstation to the MCB1700 development board.

2.1.4 Secondary Bootloader's entry mechanism

Having an entry pin grounded during reset will cause the secondary USB bootloader to enter its programming mode. For convenience, P1.20 is used by default because it is wired to the center button of the joy-stick as shown in Fig 1.

By keeping the MCB1700's joystick button pressed during a reset will cause the secondary USB bootloader to enter it programming mode[1].

2.2 Software requirements

An evaluation version of Keil's uVision3 is sufficient to compile the secondary USB bootloader and the sample user applications.

The secondary USB bootloader has been developed and tested using a MS windows-based workstation.

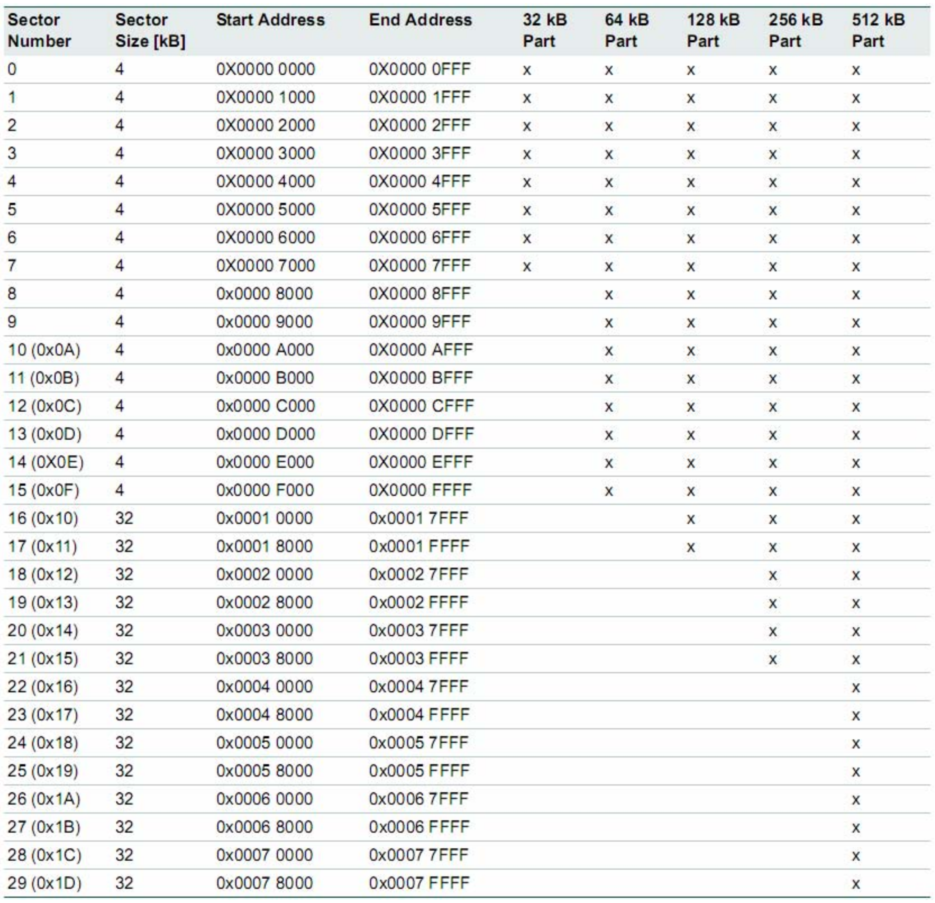
3. Flashing the LPC1700

3.1 Flash sectors

Depending on the LPC1700 part variant, the user has up to 512kB (Bytes) of on-chip flash available. This user flash space is divided up into sectors.

In order to make any modifications (even if it is just one byte) to a particular sector, the entire sector must be first erased and then (re)written.

1. Depends on the CRP value.



Sector Number	Sector Size [kB]	Start Address	End Address	32 kB Part	64 kB Part	128 kB Part	256 kB Part	512 kB Part
0	4	0X0000 0000	0X0000 0FFF	x	x	x	x	x
1	4	0X0000 1000	0X0000 1FFF	x	x	x	x	x
2	4	0X0000 2000	0X0000 2FFF	x	x	x	x	x
3	4	0X0000 3000	0X0000 3FFF	x	x	x	x	x
4	4	0X0000 4000	0X0000 4FFF	x	x	x	x	x
5	4	0X0000 5000	0X0000 5FFF	x	x	x	x	x
6	4	0X0000 6000	0X0000 6FFF	x	x	x	x	x
7	4	0X0000 7000	0X0000 7FFF	x	x	x	x	x
8	4	0x0000 8000	0X0000 8FFF		x	x	x	x
9	4	0x0000 9000	0X0000 9FFF		x	x	x	x
10 (0x0A)	4	0x0000 A000	0X0000 AFFF		x	x	x	x
11 (0x0B)	4	0x0000 B000	0X0000 BFFF		x	x	x	x
12 (0x0C)	4	0x0000 C000	0X0000 CFFF		x	x	x	x
13 (0x0D)	4	0x0000 D000	0X0000 DFFF		x	x	x	x
14 (0X0E)	4	0x0000 E000	0X0000 EFFF		x	x	x	x
15 (0x0F)	4	0x0000 F000	0X0000 FFFF		x	x	x	x
16 (0x10)	32	0x0001 0000	0x0001 7FFF			x	x	x
17 (0x11)	32	0x0001 8000	0x0001 FFFF			x	x	x
18 (0x12)	32	0x0002 0000	0x0002 7FFF				x	x
19 (0x13)	32	0x0002 8000	0x0002 FFFF				x	x
20 (0x14)	32	0x0003 0000	0x0003 7FFF				x	x
21 (0x15)	32	0x0003 8000	0x0003 FFFF				x	x
22 (0x16)	32	0x0004 0000	0x0004 7FFF					x
23 (0x17)	32	0x0004 8000	0x0004 FFFF					x
24 (0x18)	32	0x0005 0000	0x0005 7FFF					x
25 (0x19)	32	0x0005 8000	0x0005 FFFF					x
26 (0x1A)	32	0x0006 0000	0x0006 7FFF					x
27 (0x1B)	32	0x0006 8000	0x0006 FFFF					x
28 (0x1C)	32	0x0007 0000	0x0007 7FFF					x
29 (0x1D)	32	0x0007 8000	0x0007 FFFF					x

(1) Flash space varies in size, depending on the part number

Fig 2. Flash sectors

Both, the secondary USB bootloader and the user application reside in flash. Therefore, for the secondary USB bootloader to flash the user application without modifying any of its own code, the user application should be flashed starting the next available sector.

Additionally, any flash sectors that do not contain any code may be used for data storage using the IAP commands.

3.2 ISP

ISP (In-system Programming) is supported in all LPC1700 series microcontrollers. ISP programming can be used to flash the microcontroller while it is in the end-user system. The ISP protocol consists of commands that are sent in ASCII format via the UART0 interface. To enter ISP mode, P2.10 needs to be low during a reset. The ISP routines are located in the primary bootloader. Therefore, during a reset, it is possible to enter ISP mode before entering the secondary bootloader.

A detailed description of the ISP commands can be found in the LPC1700 user manual.

3.3 IAP

IAP (In Application Programming) is a feature that allows a user application to erase and write to on-chip flash memory. In order for the secondary bootloader to flash the user application onto the on-chip flash it needs to utilize these IAP commands.

A detailed description of the IAP commands can be found in the LPC1700 user manual.

3.4 CRP

The LPC1700 has three levels of CRP (Code Read Protection) as shown in Table 1.

Table 1. Code Read Protection (CRP)

CRP Explanation	CRP Level	Volume Label
The user flash can be read or written.	No CRP	CRP DISABLE
The user flash content cannot be read but can be updated. The flash memory sectors are updated depending on the new firmware image	CRP1	CRP1 ENABLE
The user flash content cannot be read but can be updated. The entire user flash memory is erased before writing the new firmware image.	CRP2	CRP2 ENABLE
The user flash content cannot be read or updated. The USB bootloader ignores the Entry Mechanism (Update Entry Pin) and always executes the user application if present. If user application is not present then "Update" mode is entered.	CRP3	CRP3 ENABLE

CRP is used as a security feature so that the code stored in flash can’t be read using external tools, such as JTAG/SWD. The level of access restrictions depends on the CRP level selected.

Caution: If CRP3 is enabled, changes to the flash space can only be performed using IAP in the user application. Assuming a valid user application is present, the secondary USB bootloader will NOT enter ISP mode if CRP3 is selected.

4. USB communication

There are many USB Device Classes like DFU (Device Field Upgrade), HID (Human Interface Device), and MSCD (Mass Storage Class Device). Because the MSCD class presents easy integration with a PC‘s operating system. This class allows the embedded system’s flash memory space to be represented as a folder in a Windows/Linux environment, and the user can update the flash with a binary image using drag-and-drop (e.g., using Windows Explorer).

To make the LPC1700 appear in the host’s file system (as a removable disk), we need a FAT (File Allocation Table). In order to fully understand how the FAT file system works, the reader is advised to search the web for details on File Allocation Table and Storage Class Devices.

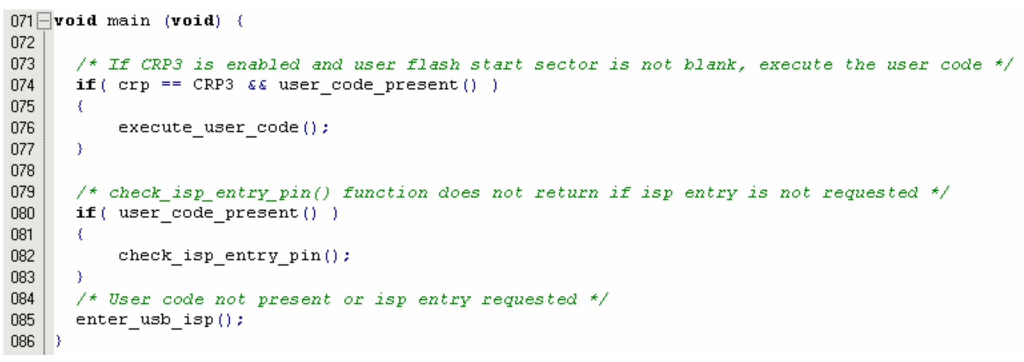
In our example code, a FAT12 file system is implemented. This file system supports up to 32 MB (volume size of the drive), making this useful for large embedded devices. To simplify things, the LPC1700 on-chip code Flash will appear as one single entity (file name: firmware.bin), solving any defragmentation problems. FAT12 is supported by Windows 9x to XP, Vista and Linux.

In this application note, we do not attempt to explain how the Mass Storage Class is implemented. This secondary USB Bootloader code is a modification of Keil's USB Mass Storage Class example (http://www.nxp.com/keil.com/336.asp). The source files for the complete project are provided for the reader to use and understand.

5. Secondary USB bootloader

5.1 Secondary bootloader entry

The boot sequence shown below is used when entering the secondary USB bootloader.



```
void main (void) {

  /* If CRP3 is enabled and user flash start sector is not blank, execute the user code */
  if( crp == CRP3 && user_code_present() )
  {
      execute_user_code();
  }

  /* check_isp_entry_pin() function does not return if isp entry is not requested */
  if( user_code_present() )
  {
      check_isp_entry_pin();
  }
  /* User code not present or isp entry requested */
  enter_usb_isp();
}
```

Fig 3. Secondary USB Bootloader boot sequence

5.1.1 Using an entry pin

The secondary USB bootloader will check the status of a GPIO pin to determine if it should enter into programming mode. This is the easiest way since no post processing is needed. By default the secondary bootloader uses P1.20.

5.1.2 Automatic secondary bootloader entry

If the secondary USB bootloader detects that no user application is present upon reset, it will automatically enter programming mode.

5.1.3 ISP entry disabled

If the secondary USB bootloader detects that a user application has already been installed and that CRP is set to level 3, then it will not enter ISP mode.

5.2 Bootloader size

Since the bootloader resides within user programmable flash, it is designed as small as possible. The larger the secondary USB bootloader is the less flash space is available to the user application.

By default, the USB bootloader has been designed to fit within the first two flash sectors (Sector 0-1) so that the user application can start from sector 2.

5.3 Code placement in flash

The secondary bootloader is placed at the starting address 0x0 so that it will be executed by the LPC1700 after reset. The USB communication within the bootloader utilizes interrupts and therefore its vector table as shown in Fig 4.

Flash programming is based on a sector-by-sector basis. This means that the code for the user application should not be stored in any of the same flash sectors as the secondary bootloader.

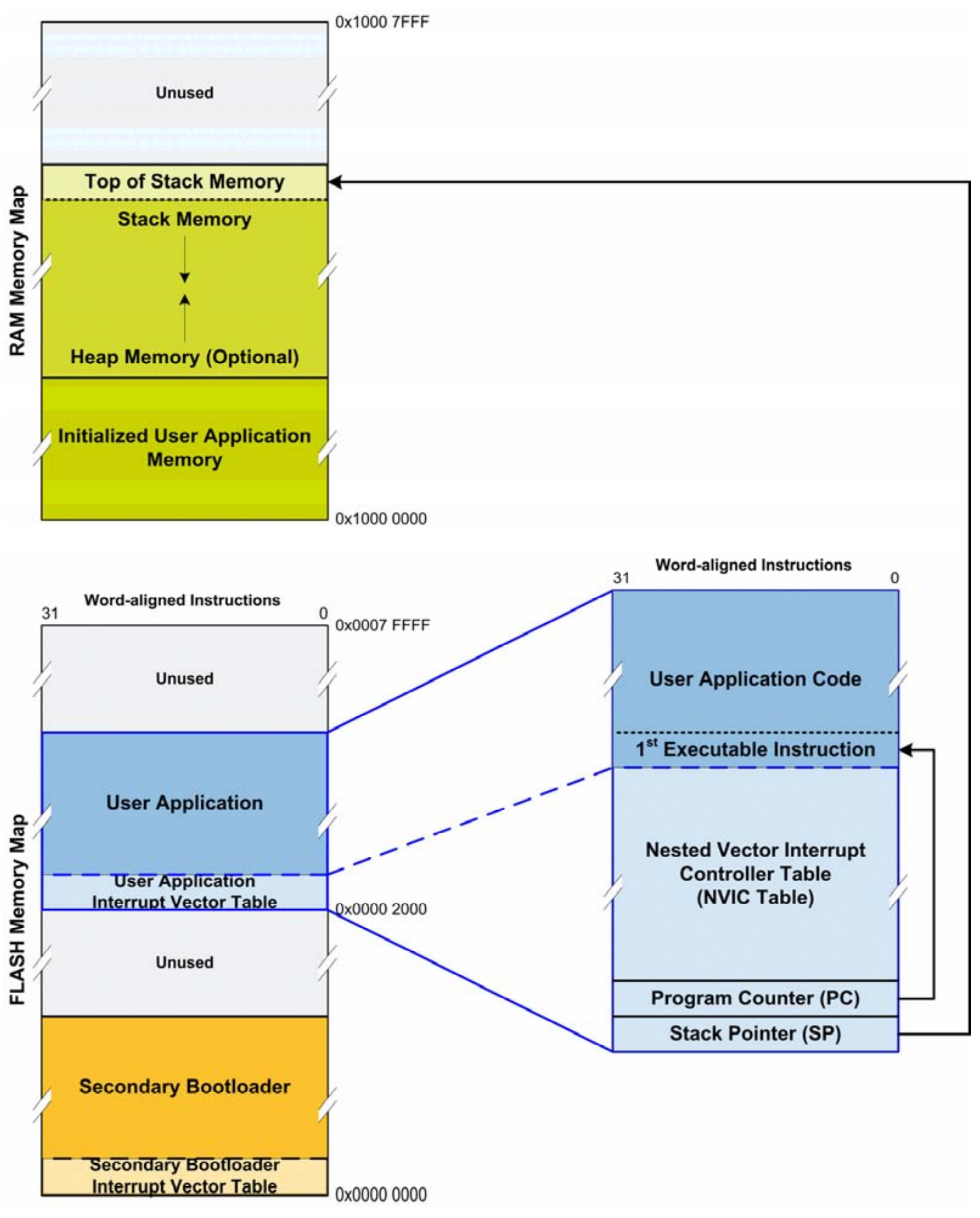
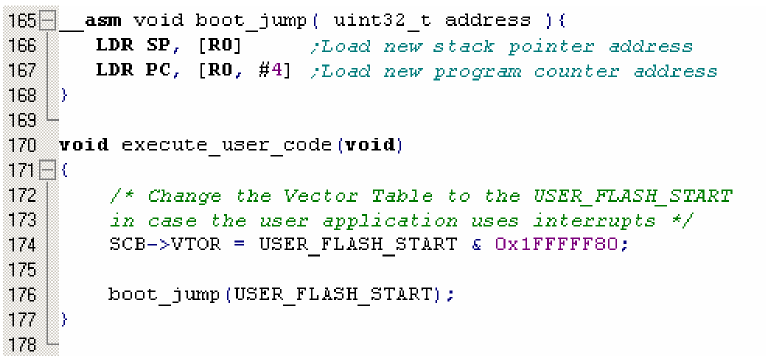


Fig 4. Bootloader and user application in flash

For efficient use of flash space, the user application should be flashed into the next available empty sector after the bootloader. Note that the user application will also contain its own interrupt vector table as shown in Fig 4.

5.3.1 User application execution

If the center joystick button is not depressed, the secondary bootloader will start the execution of the user application. Execution of the user application is performed by updating the stack pointer (SP) and program counter (PC) registers. The first two 32-bit words of a binary image contain new SP and PC values that are to be loaded into their respective registers. See Fig 4. The SP points to the new location where the user application has allocated the top of its stack (the stack grows downwards in memory). The PC on the other hand contains the location of the first executable instruction in the user application. From here on the CPU will continue normal execution and initializations specified on the user application.



```
__asm void boot_jump( uint32_t address ){
   LDR SP, [R0]      ;Load new stack pointer address
   LDR PC, [R0, #4]  ;Load new program counter address
}

void execute_user_code(void)
{
    /* Change the Vector Table to the USER_FLASH_START
    in case the user application uses interrupts */
    SCB->VTOR = USER_FLASH_START & 0x1FFFFF80;

    boot_jump(USER_FLASH_START);
}
```

Fig 5. Executing user application

By reading the first two word-aligned values from the start of the user sector and copying them into their respective registers, the CPU will automatically start proper execution of the user application.

By default, the bootloader uses 2 flash sectors. Therefore, to utilize the remaining flash, the secondary bootloader will look for the user application at 0x0000 2000. See Fig 2.

5.3.2 Updating the vector table offset register

The secondary USB bootloader will change the Cortex-M3 NVIC's Vector Offset Register in case the user application uses interrupts. See Fig 5.

By modifying the VTOR register, it will allow the NVIC to use the NVIC table of the user application instead that of the secondary bootloader's. If the user application wishes to use its vector table in SRAM, it is up to the user application to create that vector table.

See ARM's technical reference manual on the Cortex M3 for more details.

5.4 Code read protection

The secondary USB bootloader supports all levels of CRP.

When the secondary bootloader is active, it will indicate which CRP level is active (if any) to the user. The volume name can be seen from the "My Computer" window.

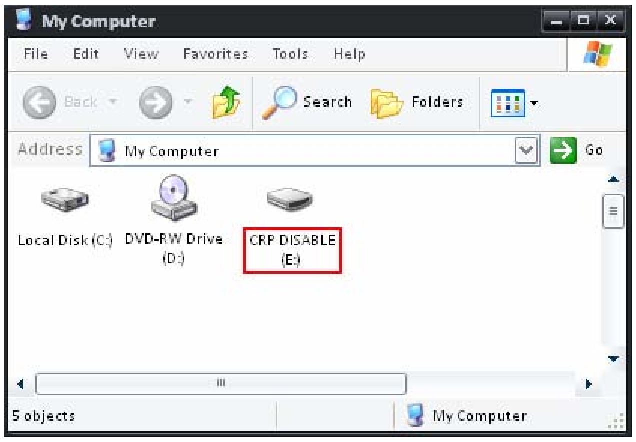


Fig 6. USB Bootloader MSCD – CRP Disabled

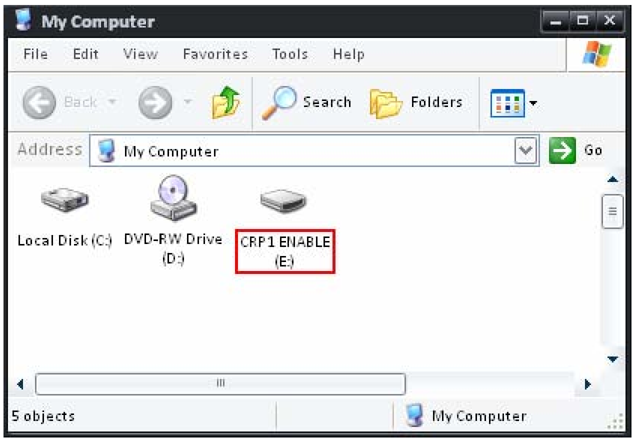


Fig 7. USB Bootloader MSCD – CRP1 Enabled

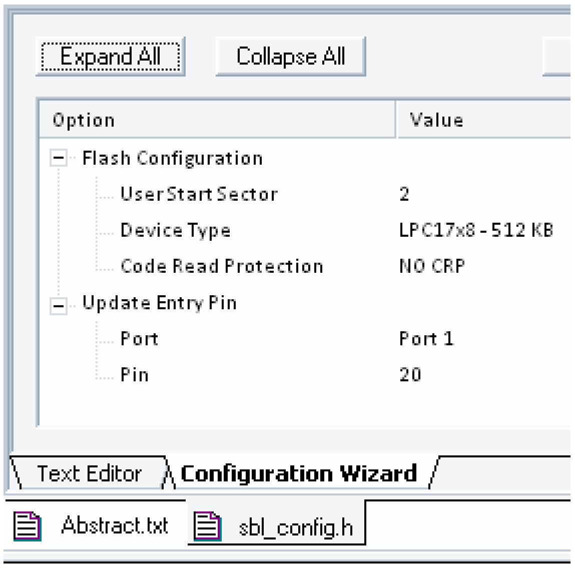
Opening the `firmware.bin` located on the MSC drive while CRP is enabled will show a file containing only NULL values. If CRP is disabled, then the binary file will contain the contents of the flash sector starting from the user application.

5.5 Bootloader code modifications

The secondary bootloader has several modification options available.

5.5.1 Flash configuration

The bootloader's flash configurations can be found in the `sbl_config.h` file. When using Keil's "Configuration Wizard" these configuration options can be easily changed.



Option	Value
Flash Configuration	
User Start Sector	2
Device Type	LPC17x8 - 512 KB
Code Read Protection	NO CRP
Update Entry Pin	
Port	Port 1
Pin	20

(1) "Configuration Wizard" view of "sbl_config.h"

Fig 8. Secondary bootloader modification options

Table 2. Secondary bootloader flash configuration options

Flash Configuration Options	Range of values	Default value
User Start Sector	0 - 29	2
Device Type	8, 64, 128, 265, 512 KB	512 KB
Code Read Protection	NO CRP, CRP1, CRP2, CRP3	NO CRP

Table 3. Secondary bootloader update entry pin options

Update Entry Pin Options	Range of values	Default value
Port	Port 0, Port 1, Port 2, Port 3, Port 4	Port 1
Pin	0 - 31	20

5.5.1.1 User start sector

This field specifies at which sector number the user application should start from. Unless the bootloader is modified such that it requires more code space than the first two sectors (Sector 0-1) this field does not need to be modified. The starting address for sector 2 on the LPC1700 is 0x0000 2000.

5.5.1.2 Device type

The device type refers to a particular part LPC1700 part number. This field specifies how much flash is available on that device. This should be changed to reflect the intended part number for which the bootloader is built for.

5.5.1.3 Code read protection

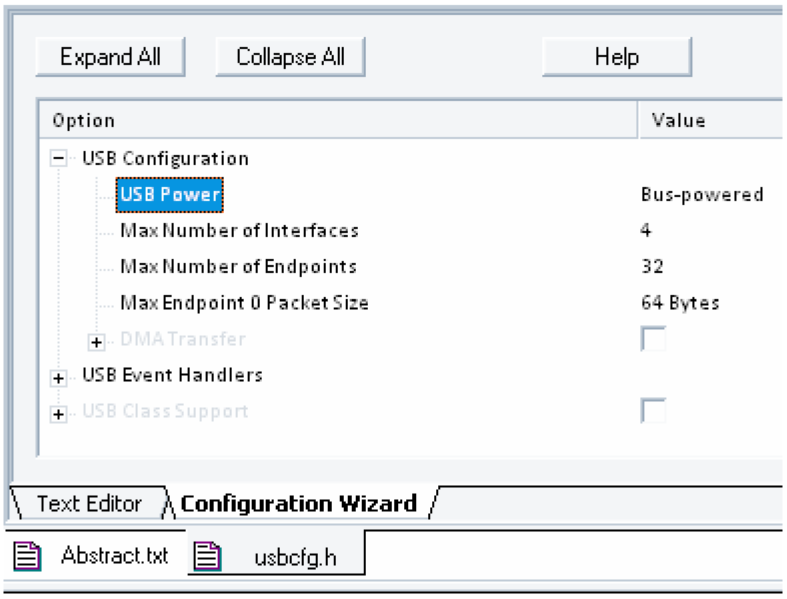
Changing the code read protection field will enable/disable the LPC1700 CRP feature.

5.5.1.4 Entry pin

The “Port” and “Pin” fields specify which I/O pin is to be checked for ISP entry. The defaults are “Port 1” and “20” respectively.

5.5.2 USB configuration

The USB configurations can be found in the `usbcfg.h` file. When using Keil’s “Configuration Wizard” these configuration options can be easily changed.



Option	Value
USB Configuration	
USB Power	Bus-powered
Max Number of Interfaces	4
Max Number of Endpoints	32
Max Endpoint 0 Packet Size	64 Bytes
DMA Transfer	☐
USB Event Handlers	
USB Class Support	☐

Fig 9. USB configuration

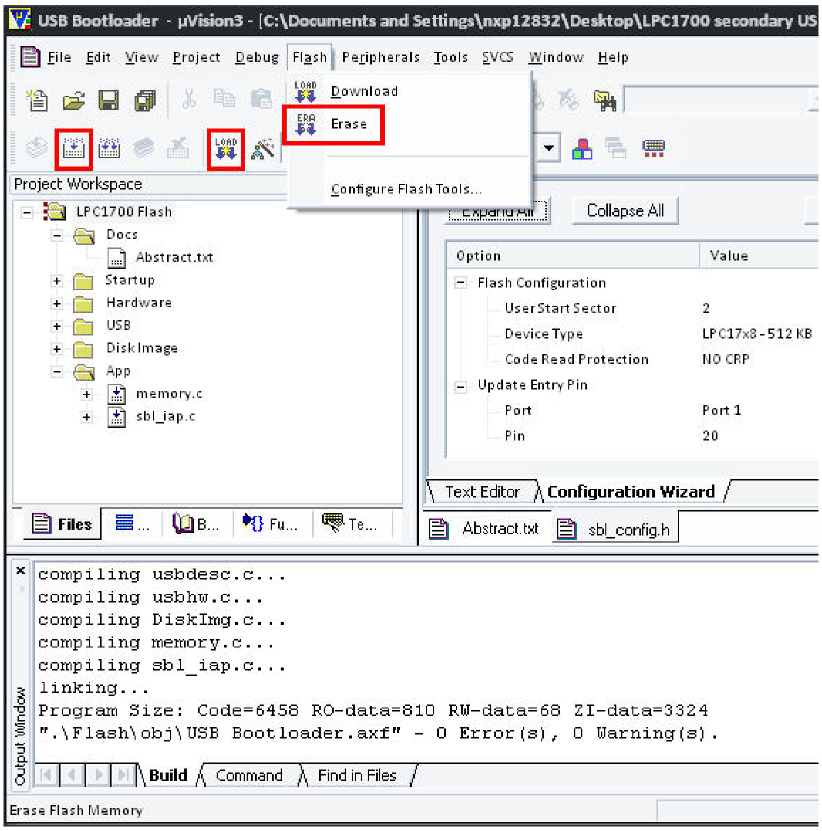
The default values shown are sufficient for the secondary USB bootloader example. Specific USB configuration properties are outside of the scope of this Application Note.

One option of interest however is the “USB Power” setting. For “USB Power” there are two options: bus-powered and self-powered. The MCB1700 is bus-powered since it derives its power from the bus. If the user’s application has its own power source (such as a battery), then the self-powered option should be selected.

5.6 Using the USB bootloader

5.6.1 Installing the secondary USB bootloader

- Open the secondary USB bootloader sample project using Keil’s uVision.
- If necessary, make any desired code changes.
- Build the project.
- Erase the LPC1700.
- Flash the LPC1700 with the bootloader.
- Reset the LPC1700.



Option	Value
Flash Configuration	
User Start Sector	2
Device Type	LPC17x8 - 512 KB
Code Read Protection	NO CRP
Update Entry Pin	
Port	Port 1
Pin	20

```
compiling usbdesc.c...
compiling usbhw.c...
compiling DiskImg.c...
compiling memory.c...
compiling sbl_iap.c...
linking...
Program Size: Code=6458 RO-data=810 RW-data=68 ZI-data=3324
".\Flash\obj\USB Bootloader.axf" - 0 Error(s), 0 Warning(s).
```

Fig 10. Flashing the secondary USB bootloader using Keil's uVision

Because the device was fully erased, the secondary bootloader will determine that there is no valid user code programmed on the device. As such, it will enter its ISP mode at startup.

5.6.2 Flashing the user application

- Create the binary file of the user application (See Section 6).
- Enter the secondary USB bootloader's ISP mode.
- Open the newly created drive created in the "My Computer" window.

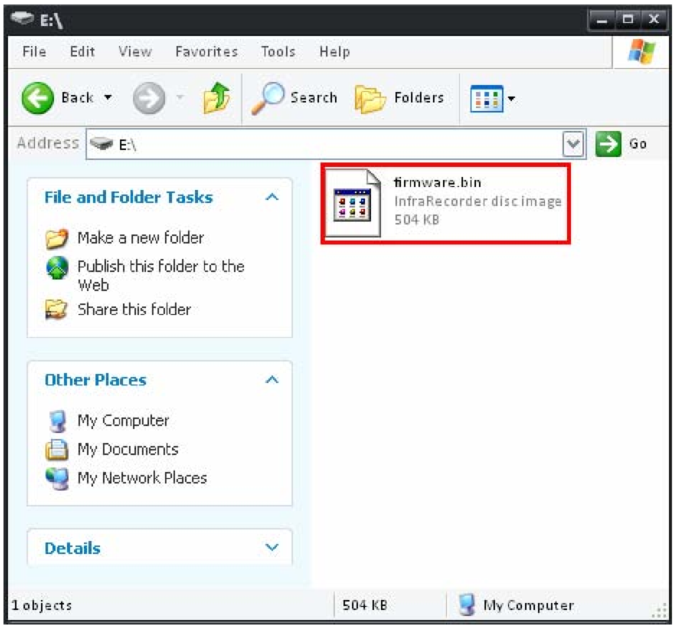


Fig 11. Firmware.bin in the USB bootloader's MSCD

- Delete `firmware.bin`
- Drag the newly created binary image of the user application onto the drive.
- Reset the LPC1700

Unless the entry pin (P1.20 by default) is grounded, the LPC1700 will now boot into the user application.

6. User application

This section briefly describes the necessary steps needed in order to make the user application usable by the bootloader.

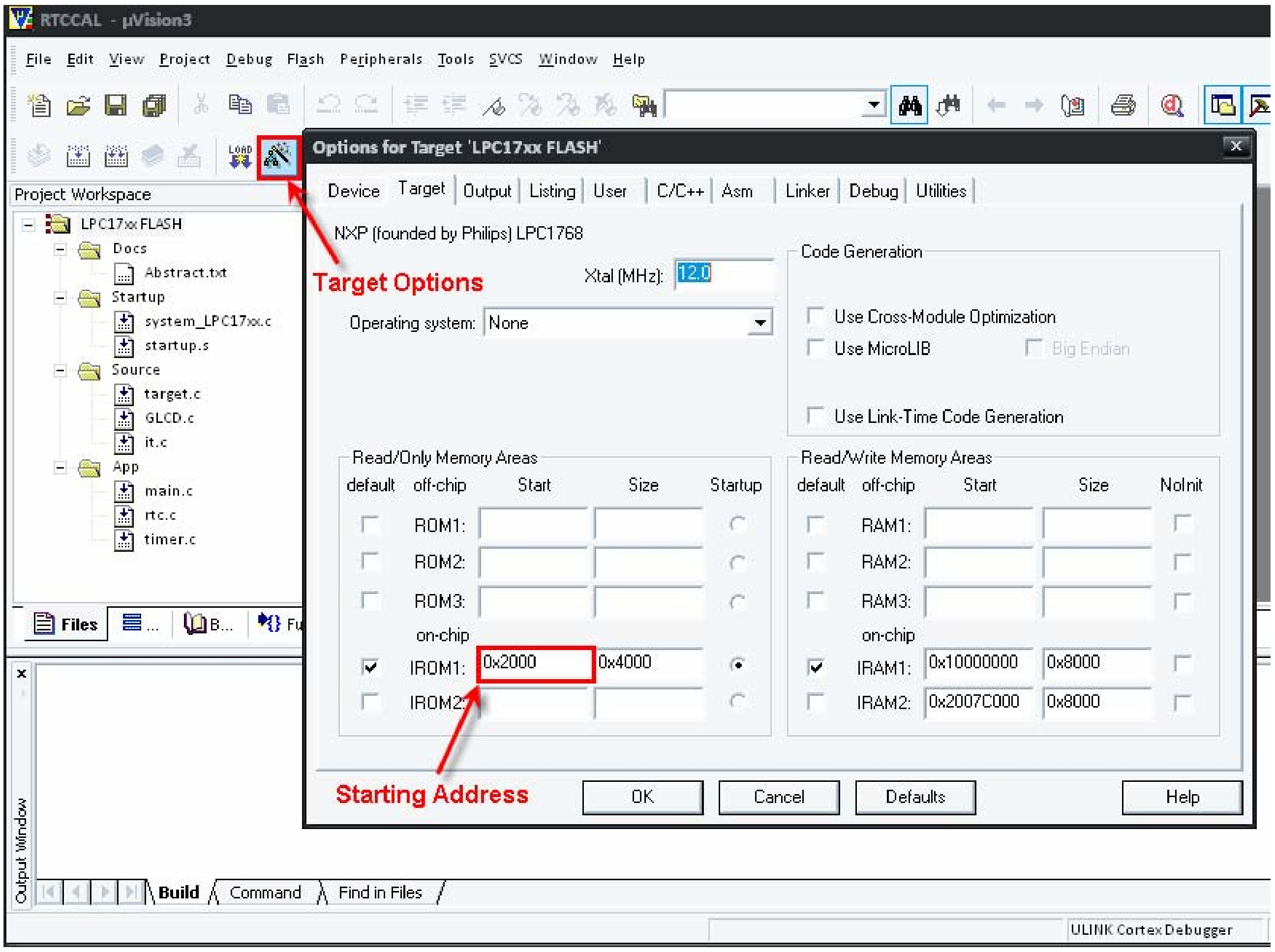
6.1 User code entry

To execute the user application the secondary USB bootloader will load the new SP and PC values into their respective registers, allowing the CPU to execute the new code correctly. Therefore, the user application must be built so that it can run from that starting address. By default, this address is 0x0000 2000.

See Fig 4 for an illustration on how the user application is stored relative to the secondary USB bootloader.

6.1.1 Updating the starting address

- Open the user application's uVision project.
- Open the "Target Options"
- Change the starting address to "0x2000"



NXP (founded by Philips) LPC1768

Xtal (MHz): 12.0

Operating system: None

Code Generation

- [ ] Use Cross-Module Optimization
- [ ] Use MicroLIB
- [ ] Big Endian
- [ ] Use Link-Time Code Generation

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x2000	0x4000	●
☐	IROM2:			○

Read/Write Memory Areas

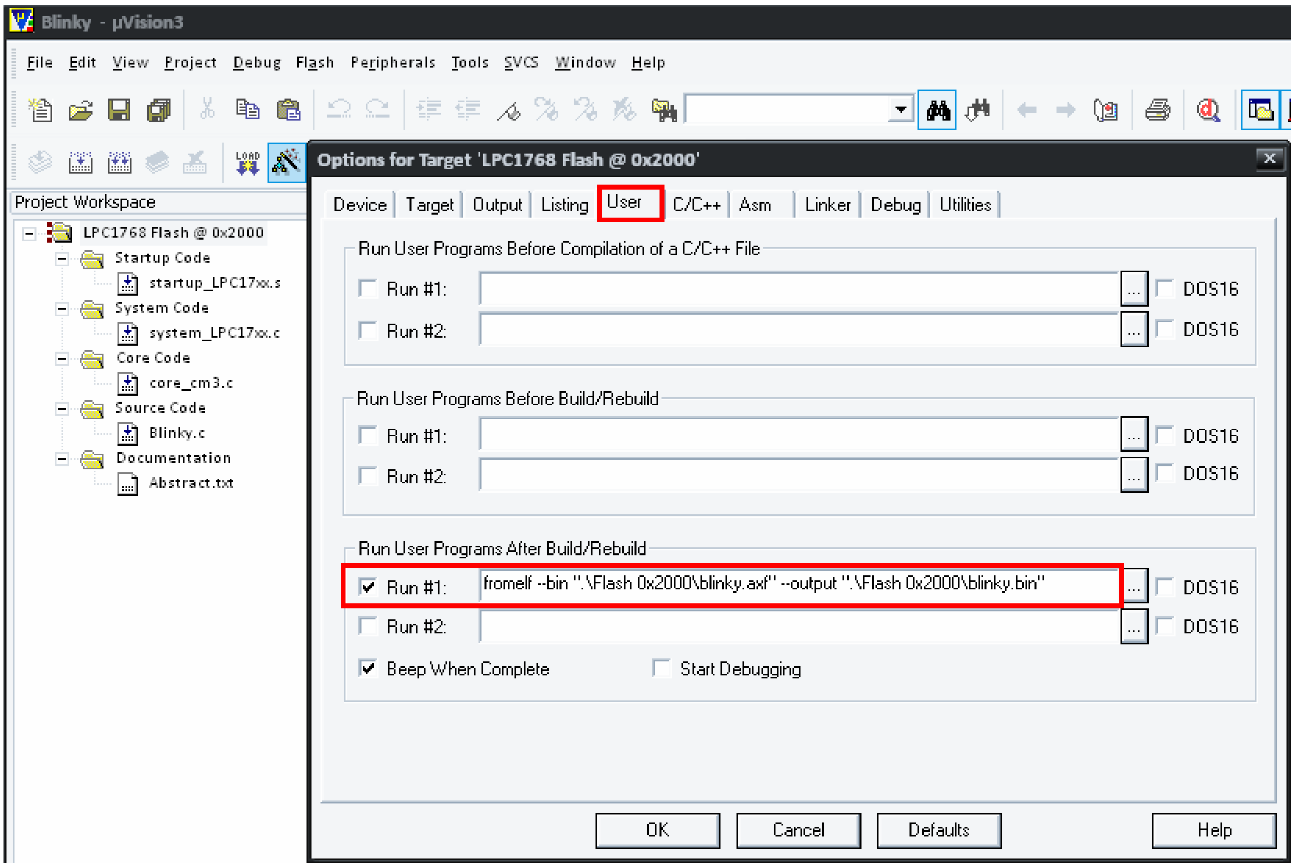
default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x10000000	0x8000	☐
☐	IRAM2:	0x2007C000	0x8000	☐

Fig 12. Configuring user application's starting address

6.1.2 Creating the binary file

By default, Keil's uVision does not create the binary which is compatible with the secondary USB bootloader. In order to create a compliant binary, use an external command line driven tool that is included with Keil's toolchain. Depending on your project configuration, you may have to change the path for the tool's command line arguments.

- Select the "User" tab in the "Target Options"
- Check the "Run #1:" from the "Run User Programs After/Rebuild"
- Type in the command to run the `fromelf` tool which can be seen in Fig 13.



Options for Target 'LPC1768 Flash @ 0x2000'

Run User Programs Before Compilation of a C/C++ File

- [ ] Run #1: — [ ] DOS16
- [ ] Run #2: — [ ] DOS16

Run User Programs Before Build/Rebuild

- [ ] Run #1: — [ ] DOS16
- [ ] Run #2: — [ ] DOS16

Run User Programs After Build/Rebuild

- [✓] Run #1: `fromelf --bin ".\Flash 0x2000\blinky.axf" --output ".\Flash 0x2000\blinky.bin"` — [ ] DOS16
- [ ] Run #2: — [ ] DOS16
- [✓] Beep When Complete
- [ ] Start Debugging

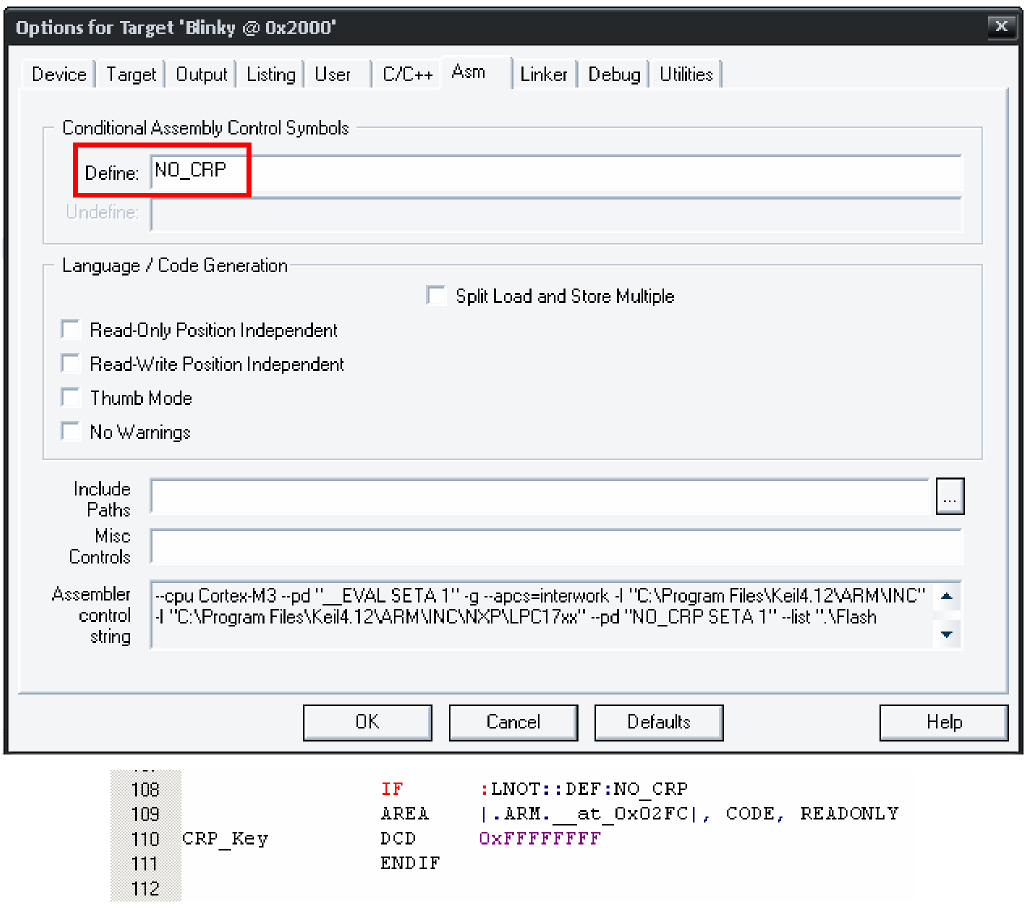
Fig 13. Configuring “fromelf” to create the required binary file

6.2 Interrupt vectors

The secondary USB bootloader will automatically adjust the NVIC’s vector table offset register (VTOR) to point to same starting address at which it branches to for user code execution. This is done so that there isn’t any extra modifications required to the user application other than changing its entry point.

6.3 NO_CRP

The secondary USB bootloader already handles the CRP configuration (See Section 5.5.1). Therefore, if your user application uses the `system_LPC17xx.s` startup file from one of Keil’s application examples, some modifications may be recommended.



Define: NO_CRP

- [ ] Split Load and Store Multiple
- [ ] Read-Only Position Independent
- [ ] Read-Write Position Independent
- [ ] Thumb Mode
- [ ] No Warnings

Assembler control string:

```
--cpu Cortex-M3 --pd "__EVAL SETA 1" -g --apcs=interwork -I "C:\Program Files\Keil4.12\ARM\INC"
-I "C:\Program Files\Keil4.12\ARM\INC\NXP\LPC17xx" --pd "NO_CRP SETA 1" --list ".\Flash
```

```
                    IF      :LNOT::DEF:NO_CRP
                    AREA    |.ARM.__at_0x02FC|, CODE, READONLY
CRP_Key             DCD     0xFFFFFFFF
                    ENDIF
```

Fig 14. Disable the CRP allocation in the “startup_LPC17xx.s” file

In the “Asm” tab of the target options define “NO_CRP”. This will prevent the assembler and linker to generate an application binary that attempts to have a flashing utility to write into flash address 0x02FC which is reserved for the CRP feature.

Writing into 0x02FC should be handled in the secondary bootloader.

7. Conclusion

By using a secondary bootloader it is possible to conveniently perform in-application software updates without using additional development hardware, such as JTAG/SWD. In this case this secondary bootloader uses USB to transfer the binaries to the LPC1700, but other channels such as Ethernet and I^{2}C are also possible.

This application note serves as a reference on how use, create, and modify a secondary USB bootloader. The secondary USB bootloader has been designed as a standalone project that contains all of its dependant source files. The user applications used for this demonstration are modified code examples that come included with Keil’s uVision toolchain.

8. Legal information

8.1 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.

8.2 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.

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 accepts 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 national authorities.

8.3 Trademarks

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

9. Contents

Date of release: 21 September 2010
Document identifier: AN10866