AN10759

USB secondary ISP bootloader for LPC23xx

Rev. 01 — 16 October 2008 **Application note**

NXP
founded by Philips

Revision history

Rev	Date	Description
01	20081016	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

The NXP LPC2000 flash 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 user's application code to be downloaded using alternative channels other than the standard UART0 used by the primary bootloader (on-chip). Possible secondary bootloaders can be written for USB, Ethernet, SPI, SSP, CAN, I²C, and even I/Os. The secondary bootloader utilizes IAP as a method to update the user's application code.

This application note will use USB as an example for developing the secondary bootloader. The code was tested using every member of the LPC23xx family, containing a USB port, and the Keil MCB2300 evaluation board.

The following sections will present a guideline for development and implementation of the USB secondary bootloader design, configuration, and test.

2. Bootloader design considerations

As with any bootloader, the user needs to select an appropriate bootloader entry mechanism for the secondary bootloader. It can be through a dedicated hardware pin or through software handshake.

Upon power-up, the secondary ISP bootloader needs to check the entry mechanism. If valid, the secondary ISP bootloader will be executed. If the entry mechanism is not valid, then the code will jump to the user code address.

Within the secondary ISP bootloader, an agreed communication protocol with the outside world is required.

It is worth noting that even when the secondary bootloader is installed on the chip, the primary on-chip bootloader will always execute after reset or power up. However, the primary bootloader entry mechanism can be deactivated using the Code Read Protection feature found in the LPC2000 devices. In other words, by enabling CRP3, the default ISP entry mechanism (determined by checking P2.10 for the LPC23xx/24xx devices) will be bypassed, allowing user code, in this case the secondary bootloader, will be executed immediately after reset or power up.

2.1 Entry mechanism

2.1.1 Dedicated hardware pin

The secondary ISP bootloader checks the status of a GPIO pin to determine if the entry is valid. This is the easiest way since no post processing is needed.

2.1.2 Software handshake

In some applications, the system is required to share existing pins or limited pins, the secondary ISP bootloader will have to perform some form of handshaking to ensure necessary conditions are met before entering the bootloader.

2.2 Communication channel selection

In many applications, serial communication is preferred because it saves on I/O pins.

In this application note, the focus will be on USB serial communication ports.

All bootloaders have some form of Host-Slave relationship. Usually, the embedded device (in this case the microcontroller), will be initiating the communication. The Host can be a PC or another embedded system. With this relationship, the bootloader developer will need to consider the effort needed for developing and testing the Slave software interface.

In the LPC2000 family on-chip UART bootloader, the Host software is developed and supported by FlashMagic (http://www.nxp.com/redirect/flashmagictool.com/). Host software will be discussed further in Section 3.

2.3 Exit

Upon completion or timeout, the exit strategy is usually a reset or power cycle with the application removing the entry condition (before next power on).

3. Host/slave software design

In this section, we discuss the code structure of serial communication channels and the PC application software testing the serial communication.

Upon checking the entry condition, the microcontroller proceeds to talk to the Slave via the prefixed communication channel. In order for the Host to be developed, some form of slave application should be available to test the host software. In most of today's PCs, the available serial communication channels are UART/IrDA, USB, and Ethernet.

In this example, we will select USB as the communication channel, and we will check for a low level on a port pin, as the entry mechanism.

In addition to ensure proper testing, we have to prefix the crystal oscillator frequency, using the Keil MCB2300 board.

3.1 USB communication

There are many USB Device Classes like DFU (Device Field Upgrade), HID (Human Interface Device), and MSCD (Mass Storage Class Device). The MSCD presents easy integration with a PC's operating systems. 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 LPC23xx appear like a folder (or disk drive), 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, we implemented the FAT12 system, which supports up to 32 MB (volume size of the drive), making this useful for large embedded devices. To simplify things, the LPC23xx 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 USB Secondary Bootloader code is a modification of Keil's USB Mass Storage Class example (http://www.nxp.com/redirect/keil.com/336.asp). The source files for the complete project are provided for the reader to use and understand.

3.2 USB Secondary bootloader Configuration

In this section, we will introduce the key files the developer will need to change in order to adapt this USB secondary ISP bootloader for their application.

3.2.1 Setup file (sbl_config.h)

This file configures the secondary bootloader. The user should change this according to meet their requirements. See section 4 to see how the changes can be made using a Configuration Wizard.

Some definitions and explanation:

USER_START_SECTOR – Starting sector where the user code will reside. Since the secondary ISP bootloader occupies 2 x 4 kB sectors, the user code will start at sector 2.

MAX_USER_SECTOR – This parameter is device dependent. In a 512 kB device, it will be 27 sectors. Modify this according to the microcontroller's flash memory (see Fig 1).

CRP – Code Read Protection. This parameter allows select the desired CRP level. Choosing CRP3, the primary bootloader's entry mechanism check will be bypassed. See LPC23XX User manual (UM10211) and Table 1 for CRP details.

ISP_ENTRY_GPIO_REG – In this example, we define Port 0 (0xE0028000).
ISP_ENTRY_PIN - In this example, we have chosen P0.6 hence this is 6.

3.2.2 Interrupt handling code

In this example, the Vector Interrupt Controller (VIC) has been set up. Please refer to the Startup.s, *USB_Init()* and *USB_ISR()* functions in usbhw.c source file for more details.

3.3 USB Secondary bootloader Modes of Operation

Since this is an add-on function, the user must first program the blank LPC23xx device with this USB Secondary ISP bootloader before it can be used. This can be done via JTAG or the UART ISP.

On connecting to the PC running Windows, the LPC23xx Microcontroller memory is automatically recognized by the PC, which will load a generic Mass Storage driver.

This USB Secondary ISP bootloader operates in two modes:

1. Execution mode running the user application.

2. Flash update mode is selected by tying a GPIO pin low**.**

Sector Number	Sector Size [kB]	Address Range	128 kB Part	256 kB Part	512 kB Part
0	4	0X0000 0000 - 0X0000 0FFF	x	x	x
1	4	0X0000 1000 - 0X0000 1FFF	x	x	x
2	4	0X0000 2000 - 0X0000 2FFF	x	x	x
3	4	0X0000 3000 - 0X0000 3FFF	x	x	x
4	4	0X0000 4000 - 0X0000 4FFF	x	x	x
5	4	0X0000 5000 - 0X0000 5FFF	x	x	x
6	4	0X0000 6000 - 0X0000 6FFF	x	x	x
7	4	0X0000 7000 - 0X0000 7FFF	x	x	x
8	32	0x0000 8000 - 0X0000 FFFF	x	x	x
9	32	0x0001 0000 - 0X0001 7FFF	x	x	x
10 (0x0A)	32	0x0001 8000 - 0X0001 FFFF	x	x	x
11 (0x0B)	32	0x0002 0000 - 0X0002 7FFF		x	x
12 (0x0C)	32	0x0002 8000 - 0X0002 FFFF		x	x
13 (0x0D)	32	0x0003 0000 - 0X0003 7FFF		x	x
14 (0X0E)	32	0x0003 8000 - 0X0003 FFFF		x	x
15 (0x0F)	32	0x0004 0000 - 0X0004 7FFF			x
16 (0x10)	32	0x0004 8000 - 0X0004 FFFF			x
17 (0x11)	32	0x0005 0000 - 0X0005 7FFF			x
18 (0x12)	32	0x0005 8000 - 0X0005 FFFF			x
19 (0x13)	32	0x0006 0000 - 0X0006 7FFF			x
20 (0x14)	32	0x0006 8000 - 0X0006 FFFF			x
21 (0x15)	32	0x0007 0000 - 0X0007 7FFF			x
22 (0x16)	4	0x0007 8000 - 0X0007 8FFF			x
23 (0x17)	4	0x0007 9000 - 0X0007 9FFF			x
24 (0x18)	4	0x0007 A000 - 0X0007 AFFF			x
25 (0x19)	4	0x0007 B000 - 0X0007 BFFF			x
26 (0x1A)	4	0x0007 C000 - 0X0007 CFFF			x
27 (0x1B)	4	0x0007 D000 - 0X0007 DFFF			x

Fig 1. Flash sectors in LPC23xx

3.3.1.1 Execution

This mode is selected when the user defined Update Entry Pin level is high.

In addition, the code checks if User application is programmed in the internal flash memory. The first sector in the user application space is used to detect the presence of a

user application. If this sector is not blank, then the user application is executed. If the device is blank, the Update mode is always activated.

3.3.1.2 Update

The user application programmed in the internal flash memory is updated via USB. This mode is selected by tying the Update Entry Pin low.

In the Update mode, the USB bootloader shows up as a USB mass storage device when the MCB2300 is plugged into a PC USB port. The entire user flash contents are mapped to a file named "firmware.bin". For example, for a 512 kB device, the maximum user flash space is 496 kB.

The commands supported in Windows/Linux are Copy and Delete.

3.3.1.3 Code read protection

The USB secondary ISP bootloader can be configured to enable the code read protection feature of the LPC23xx microcontrollers. The volume label on the mass storage device indicates the Code Read Protection status. See Setup File (sbl_config.h) for details.

Table 1. Code Read Protection (CRP)

CRP explanation	CRP status	Volume label
The user flash can be read or written.	No CRP	CRP DISABLD
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 ENABLD
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 ENABLD
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 ENABLD

3.3.1.4 User code starting location

The user application needs to be modified to execute from address 0x2000 because the first two sectors are occupied by the USB secondary bootloader. In the zip file, the "User Code Sample Blinky" folder contains an example linked to execute from the address 0x2000.

3.4 Testing the USB mass storage driver with PC

The following LEDs on MCB2300 are used to display the status:

- P2.0: LED_RD is on when USB Device is performing Read access
- P2.1: LED_WR is on when USB Device is performing Write access

- P2.4: LED_CFG is on when USB Device is configured
- P2.5: LED_SUSP is on when USB Device is suspended

Please, check that both the D+ and D- jumpers are in the lower position. The UMODE jumper should be in the upper position if the SoftConnect feature is enabled, and in the lower position if SoftConnect is not used.

For this test, it is necessary to program the Secondary bootloader (Memory.uv2 project output) into the flash using JTAG or FlashMagic.

With the Update Entry Pin (in our example code, we assigned P0.6) set low, plugging the MCB2300 board to the PC, we'll see a 512 kB device with no CRP show up as:

Fig 2.   Blank device with no CRP
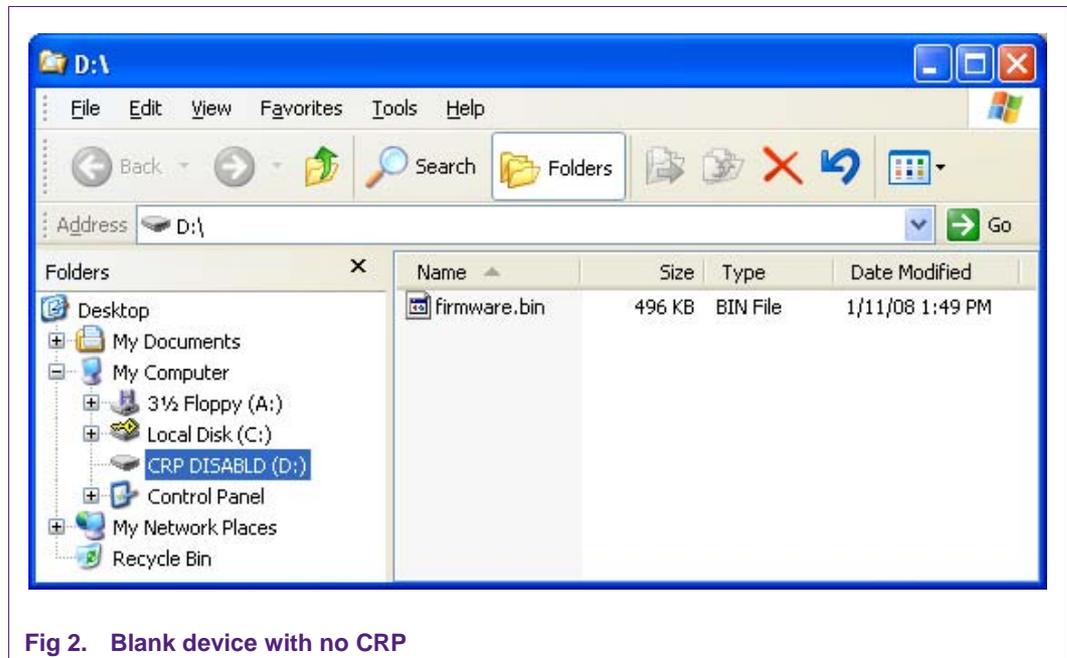
To update the user application, delete the file "firmware.bin",
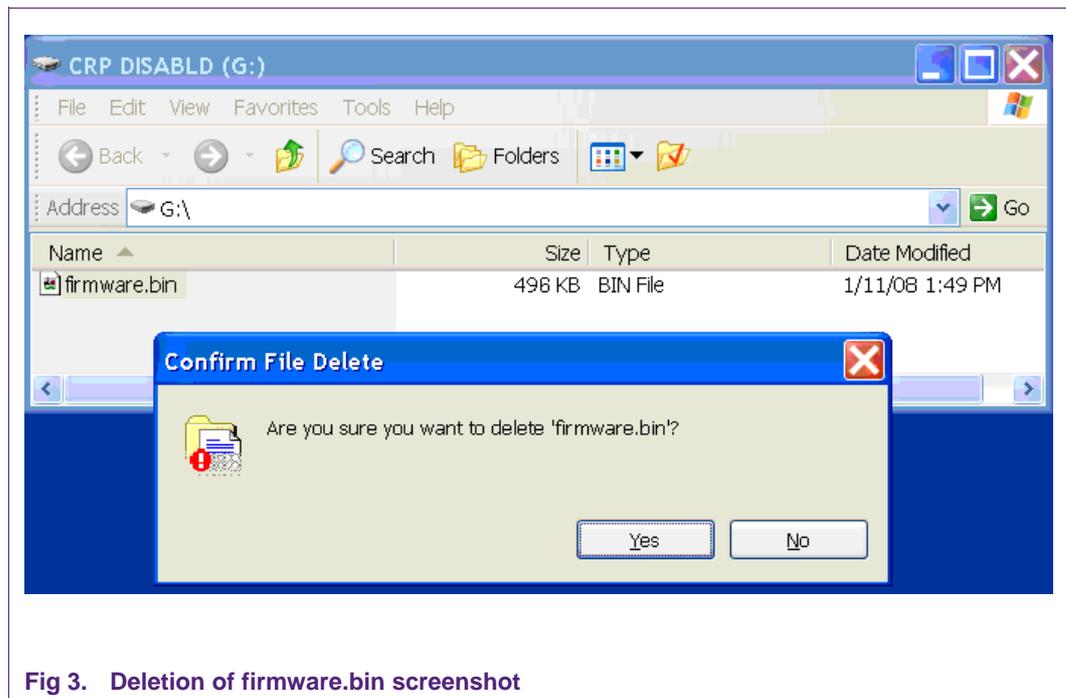
Fig 3.   Deletion of firmware.bin screenshot
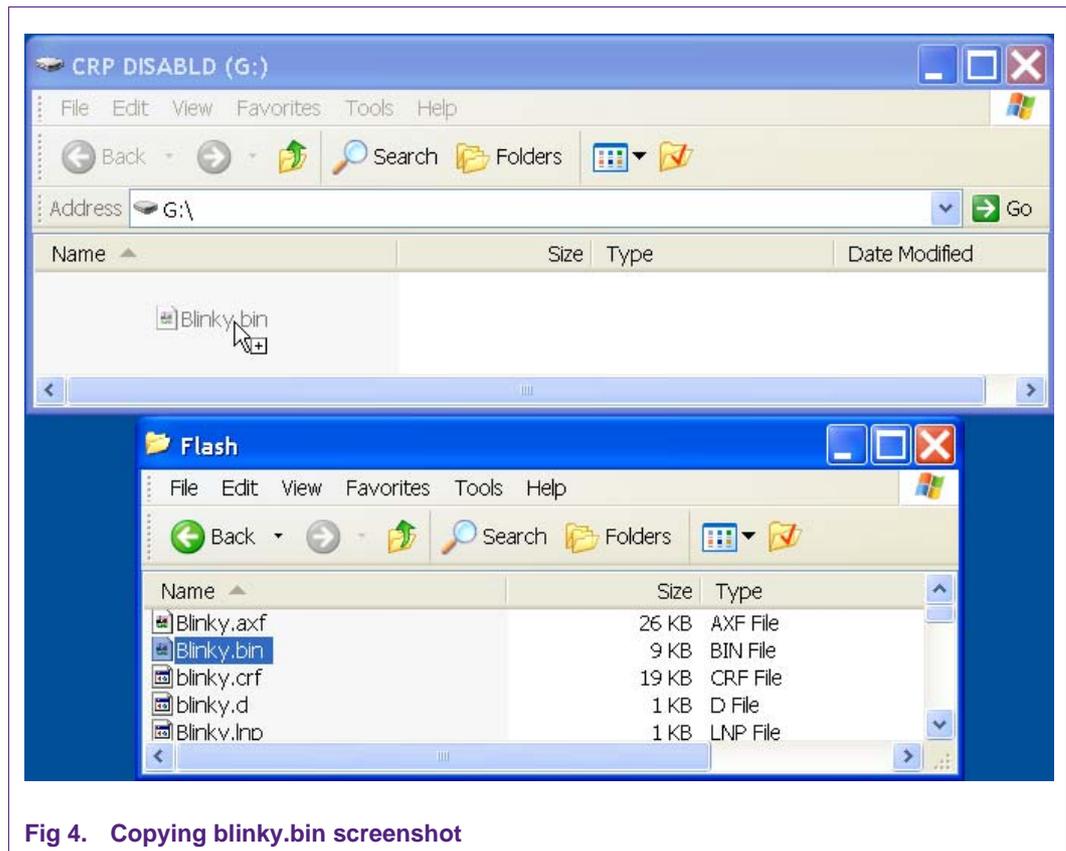
Fig 4.   Copying blinky.bin screenshot

Then copy the example binary file "Blinky.bin" to the mass storage device (you can drag-and-drop the file) as shown in Fig 4. The user flash is then updated with the new firmware image. When "Blinky.bin" is copied, it will appear as shown in Fig 6 (as long as the device/MCB2300 board is not reset or power cycled):
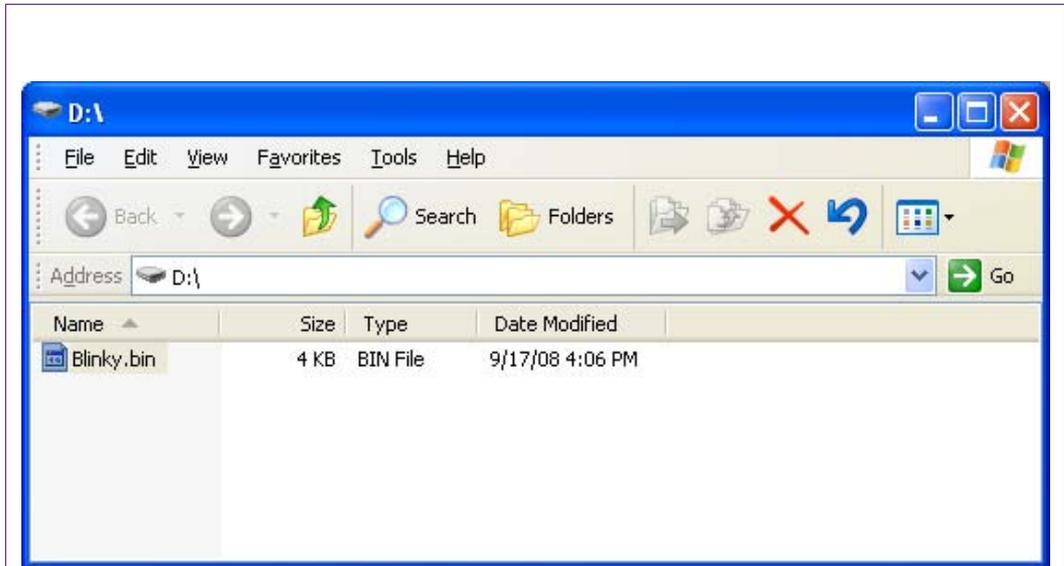
Fig 5. Blinky.bin screenshot

After reset or power cycle, you will see the image as shown in Fig 6 (note the file size will always be the max device size minus 2 sectors):
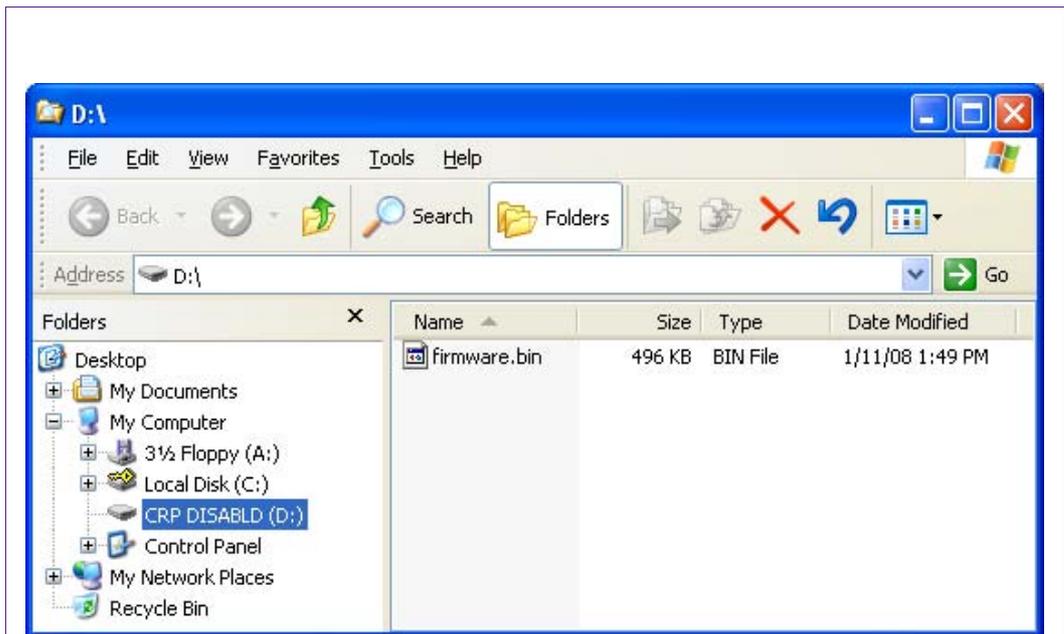
Fig 6. firmware.bin screenshot

With CRP set, any attempt to read the contents of the binary file will appear as "00".

Here are screen shots of CRP1, CRP 2 and CRP 3 (Fig 7, Fig 8, and Fig 9).
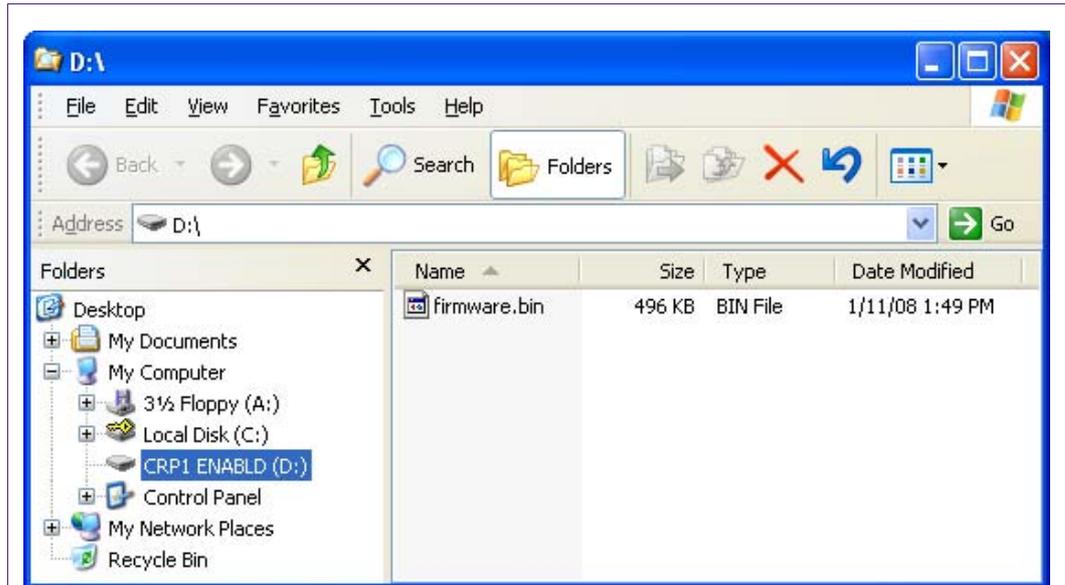
Fig 7.   CRP1 screenshot

Fig 8.   CRP2 screenshot

AN10759_1

Application note **Rev. 01 — 16 October 2008** **11 of 19**
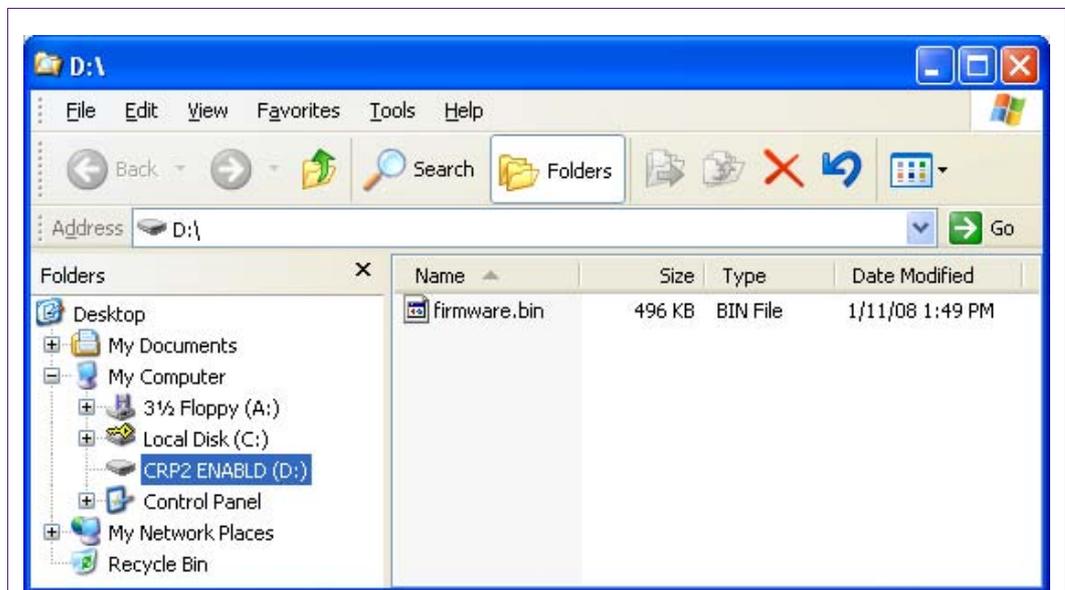
Fig 9. CRP3 screenshot
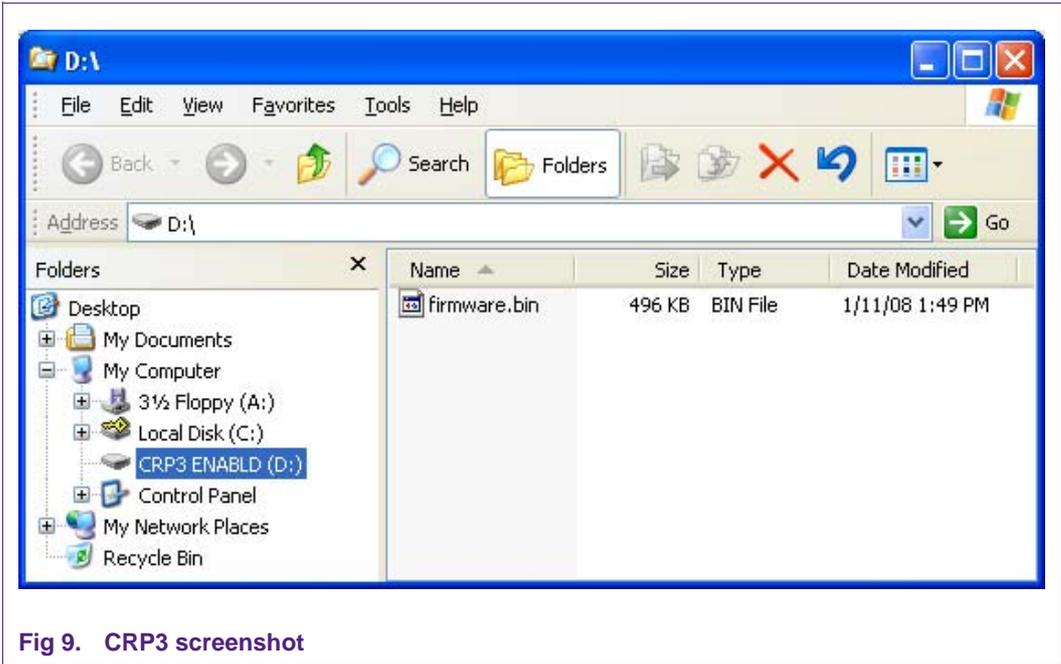
4. Modifying the code

The code can be used without modification; however, the user will probably need to change the Vendor ID and Product ID. This can be done in the usbcfg.h file. When this file is selected, the user can modify the parameters directly using the Text Editor or using the Configuration Wizard shown in Fig 10.
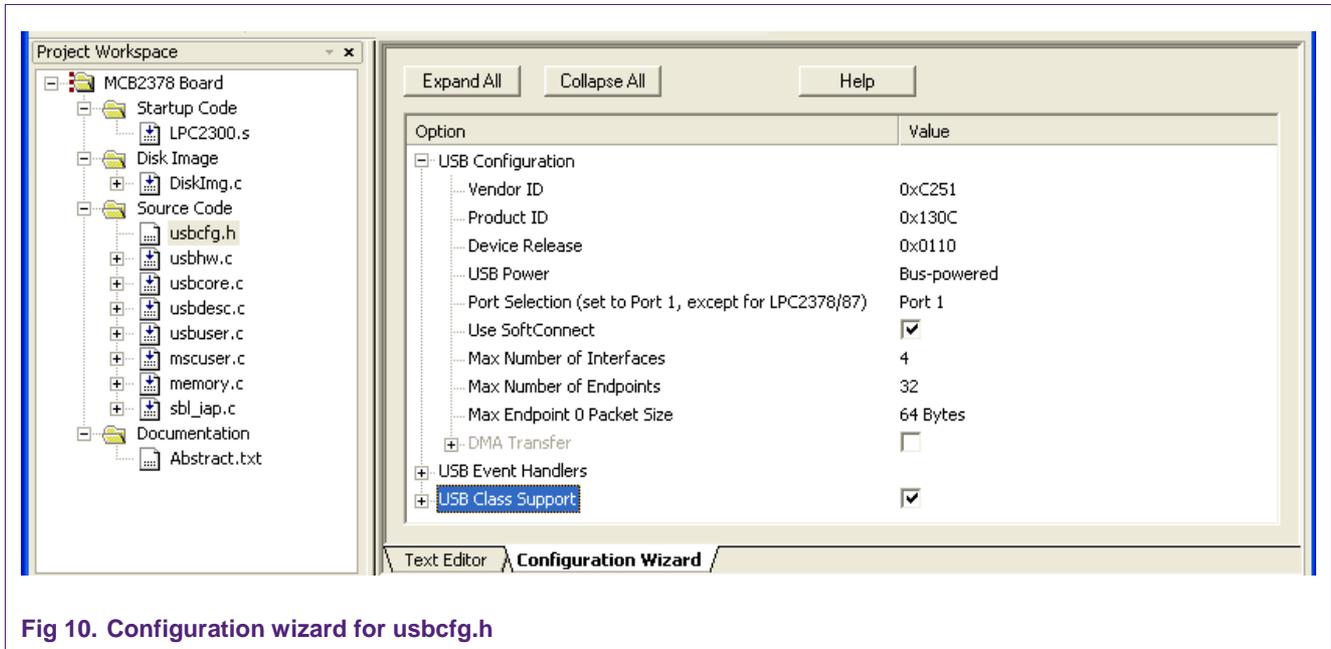
Fig 10. Configuration wizard for usbcfg.h

Vendor ID – the user may purchase a vendor ID directly from http://www.nxp.com/redirect/usb.org. There are also companies that sell a Vendor ID with a small block of Product IDs. You can find these companies by searching VID/PID on the internet.

Product ID – a product ID is a number the user would give to identify the end-product. The ID for this mass storage flash updater should be different than the Product ID used for the user's actual application.

Device Release – this is a binary code decimal value that defines the product revision of the user application. In this example, the 0x0110 means that it is revision 1.1.

USB Power – there are two possibilities: bus-powered and self-powered. The MCB2300 is bus-powered since it derives its power from the bus. If the user's application has its own power source, then the self-powered option needs to be selected.

Port Selection – the LPC2378 and LPC2387 can route the USB device signals to two different pin locations. The user can select from Port 1 or Port 2. It is important to leave this selection as Port 1 for other members of LPC23xx family. The MCB2300 demo board only supports Port 1.

SoftConnect – is a feature allowing the USB device controller in the LPC23xx family to connect/disconnect the pull-up resistor to the D+ USB signal. For bus-powered applications, either selection can be used. The SoftConnect feature should be enabled for self-powered USB applications. The MCB2300 board has the available hardware to support the SoftConnect feature so you can enable this feature by checking this option.

Jumper UMODE on the MCB2300 demo board should be in the upper position when using SoftConnect, and it should be in the lower position if SoftConnect is not used.

The other parameters will probably not need to be changed.

The user can use the sbl_config.h Configuration Wizard to modify the code so that no knowledge of C coding is required to make changes. The Configuration Wizard can be entered by selecting the sbl_config.h file and then selecting the Configuration Wizard tab on the right side of the Keil IDE. Press the Expand All button to see the screen shown below (Fig 11).
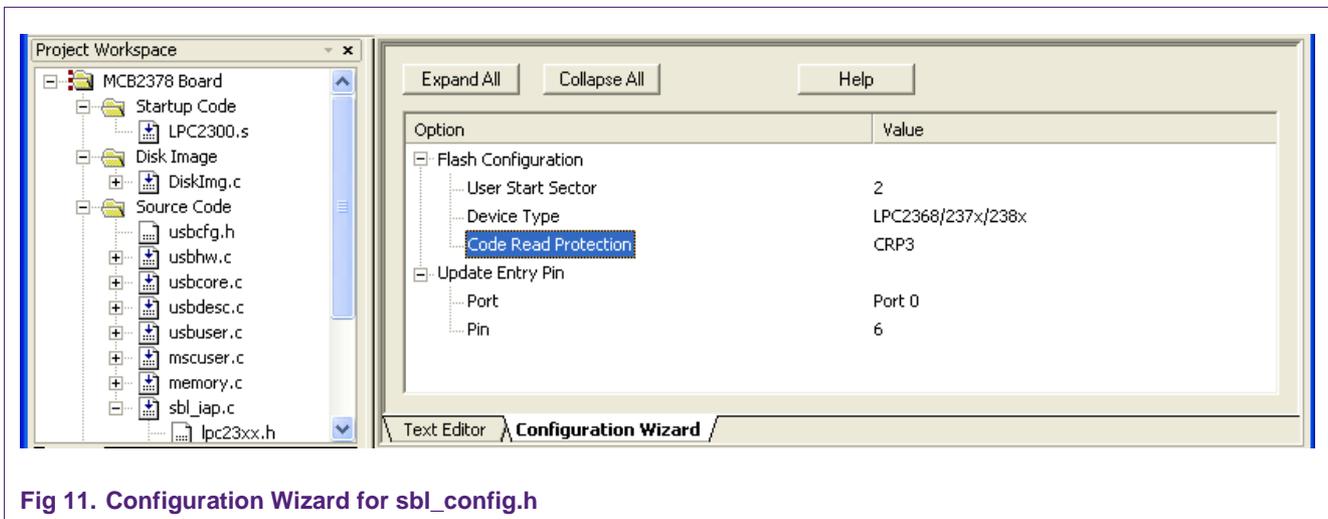
Fig 11. Configuration Wizard for sbl_config.h

4.1 Flash configuration

User Start Sector – normally, this value should be sector 2 since we are using the first two sectors of the flash for the USB update code.

Device Type – by choosing the appropriate device from the drop-down menu, the MAX_USER_SECTOR will be modified

Code Read Protection – choose the appropriate level of Code Read Protection. See Table 1 for an explanation of CRP.

4.2 Update entry pin

The flash update mode is entered when the Update Entry Pin is low at reset. Note that it is up to the user to define a pin that is available and appropriate. For example, it would not be a good idea to use P0.12 to P0.14 since these pins are not implemented on the LPC236x devices. The Configuration Wizard does not check to ensure the user has requested a valid pin.

Port – defines which port is used for the Update Entry Pin. Choose between Port 0 or Port 1. The Wizard places the correct address in the ISP_ENTRY_GPIO_REG definition.

Pin – a low on this pin at reset will force the flash update mode to be entered. The value will be placed in the ISP_ENTRY_PIN in the sbl_config.h file.

4.3 Default name of flash file

Default name of the flash file on the image is currently set to FIRMWARE.BIN. This can be modified by changing the contents of the DiskImg.c file. In the RootDirEntry array (see Fig 12), the name of the file appears as 11 characters. The first 8 characters are the filename while the last three are the suffix. The number of characters should not be modified.
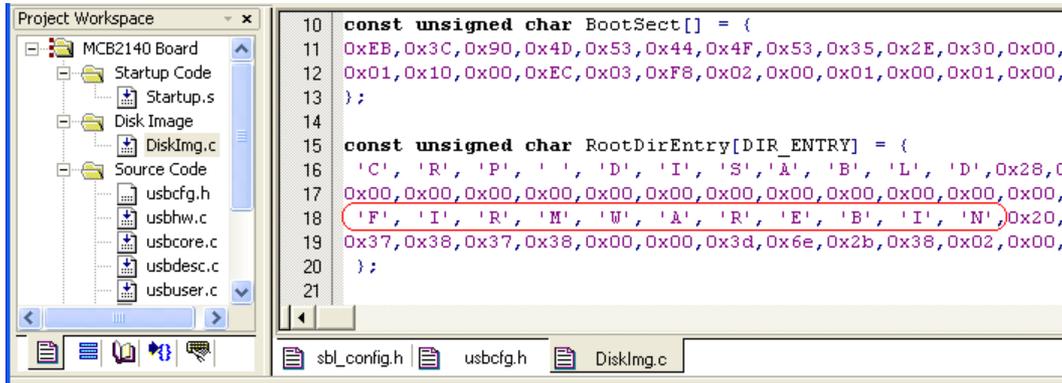

Fig 12. DiskImg.c file

4.4 Disk Properties

When the LPC23xx demo board is plugged into the USB port, and the secondary bootloader has been programmed into it, Windows will display the disk properties shown in Fig 13.
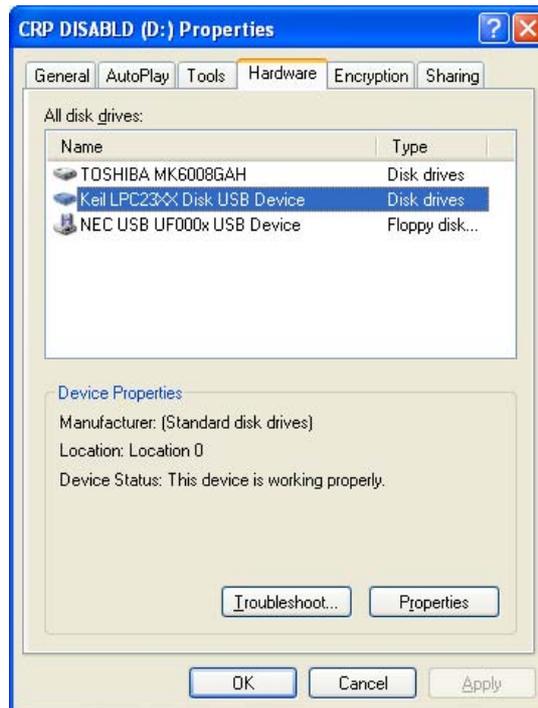
Fig 13. Mass storage device properties in a Windows XP environment

AN10759_1

Application note **Rev. 01 — 16 October 2008** **15 of 19**

The name of the disk shown in the Properties dialog box can be modified in the MSC_Inquiry() function, in the mscuser.c file.

4.5 User Code Modifications

User code loaded onto the mass storage drive must start at address 0x2000. The starting location of the code can be found in the Target tab of the Keil IDE, as shown in Fig 14.
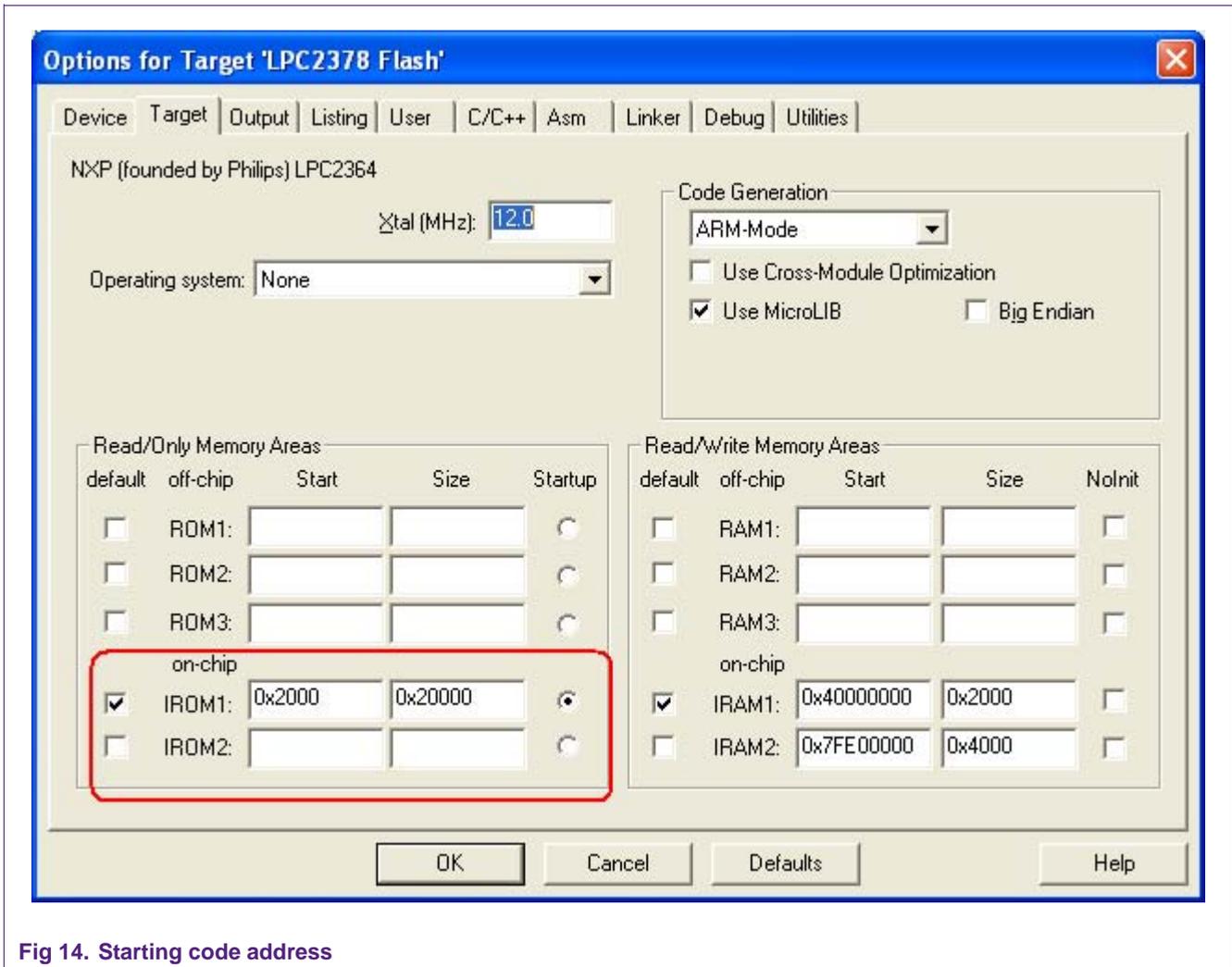
Fig 14. Starting code address

4.6 Creating a bin file

A binary (bin) file can be created by the Keil IDE by adding the highlighted statement shown in Fig 15.
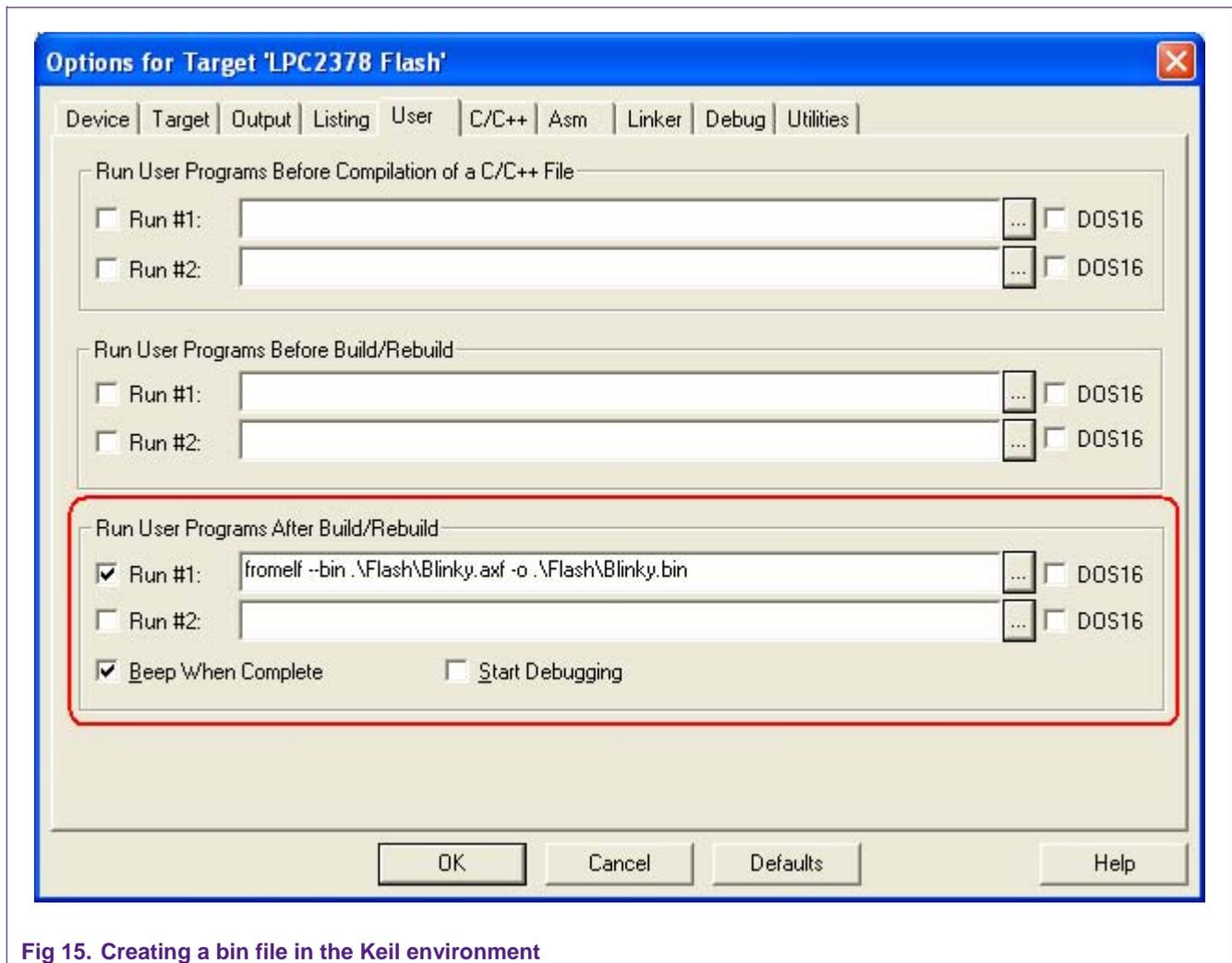
Fig 15. Creating a bin file in the Keil environment

5. Conclusion

A secondary ISP bootloader is shown using USB Mass Storage class. The user can develop a different Secondary ISP bootloader using other serial communication based on this concept; the important thing is to develop the Host/Slave interface for the serial communication.

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

General — 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.

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 medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a 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 for 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.

6.3 Trademarks

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

AN10759_1

Application note **Rev. 01 — 16 October 2008** **18 of 19**

7. Contents

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

founded by
PHILIPS