



Technical notes on using Analog Devices DSPs, processors and development tools
 Visit our Web resources <http://www.analog.com/ee-notes> and <http://www.analog.com/processors> or
 e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Implementing Second-Stage Loaders for ADSP-BF60x Blackfin® Processors

Contributed by Harrington, David

Rev 1 – November 18, 2014

Introduction

While ADSP-BF60x Blackfin® Processors (hereafter referred to as ADSP-BF609 processors) natively support numerous boot modes^[1], there may be applications that require additional boot time functionality. To support extensions to the boot process, a second-stage loader (SSL) can be introduced, where a small application is loaded into the processor using a natively supported boot mode. This SSL kernel can be used to customize the configuration of the processor and/or perform automated tasks as part of the boot process.

This EE-Note describes how to write a SSL for ADSP-BF609 processors to selectively boot one of multiple executable files (DXEs) from SPI flash memory using basic GPIO push-buttons during system start-up. The example code provided in the associated .ZIP file has been tested with CrossCore® Embedded Studio (CCES) version 1.1.0 and with the ADSP-BF609 EZ-KIT® evaluation board revision 1.0. When using the associated project files, make sure that the latest Board Support Package (BSP) for the ADSP-BF609 EZ-KIT board is installed (www.analog.com/BF609EZBoard).

Second-Stage Loader (SSL)

A second-stage loader is a stand-alone application that is executed at boot time before the actual application is dynamically loaded into memory. It has its own executable file (DXE) and hence, it will consume space in the boot source memory as part of the loader image (LDR).

First of all, the SSL is loaded into the first memory location in L1 instruction SRAM. Then, it performs whatever tasks it is intended to do, and it finally completes the boot process, overwriting itself with the actual end application.

The example project associated with this EE-Note allows for multiple DXE applications to be selected among at boot time, employing a simple GPIO interface to utilize two push-button switches available on the ADSP-BF609 EZ-KIT board to serve this purpose.

The method used in this document creates three separate LDR images – the SSL itself and two unique LED blink applications, LED_1 and LED_2 – each of which is programmed into different offsets within the SPI flash memory, as shown in Table 1.

Button	DXE	SPI Memory Offset
--	SSL	0x000000
PB1	LED_1	0x100000
PB2	LED_2	0x200000

Table 1. SPI Memory Map

The SSL boots from SPI memory and configures the processor to accept PB1 and PB2 as GPIO inputs, at which point it idles the processor until PB1 or PB2 are pressed. The push-buttons then determine which of the two DXEs to boot into the processor and execute.

Booting Applications

ADSP-BF609 processors feature instruction ROM (Read Only Memory) space that includes a function for selecting the memory address to boot from called `rom_Boot()`. Detailed information on the functions contained in L1 instruction ROM can be found in the *ADSP-BF60x Blackfin Processor Hardware Reference*^{[1][2]}. To use the ROM API, the file `bfrom.h` needs to be included in the project, as follows:

```
#include <bfrom.h>
```

This file is provided with CCES (default installation path):

```
C:\Analog Devices\CrossCore Embedded Studio 1.1.0\Blackfin\include
```

In the SSL, `rom_Boot()` is called in the interrupt service routine (ISR) for each of the two push-buttons:

```
rom_Boot(LED_1,0,0,0,0x20210002,0);
rom_Boot(LED_2,0,0,0,0x20210002,0);
```

LED_1 and LED_2 are byte offsets into memory, expressed in integer format. These are the memory locations that the corresponding LDR images need to be loaded to in the SPI flash memory. `0x2021002` is the `dBootCommand` parameter, which configures the boot mode to SPI master and sets the SPI clock, configures the SPI chip-select output channel and enables device detection (for more details about `dBootCommand`, see [1][2]).

New to ADSP-BF609 processors boot ROM is the ability to choose which chip-select output is to be controlled for use during SPI booting, as any of the `SPI_SELn` outputs can be used for SPI

device selection via `dBootCommand`. This allows for multiple SPI flash modules to be connected to SPI0 or SPI1, and booted from using the `rom_Boot()` API.



The `rom_Boot()` function will not correctly work while using an In-Circuit-Emulator (ICE) or Debug Agent. In order to test the `rom_Boot()` function, your application must be flashed to memory and booted without an attached emulator.

Creating LDR Files

To create multiple DXEs, each loader file (LDR) needs to be created as a binary file. To create the necessary LDR files, do the following for each of the three example projects provided in the associated .ZIP file:



If the application you are booting is a dual-core application, you must make sure your SSL is also a dual-core application to ensure proper core initialization, which is why this example has the LDR file being generated from Core1 application.

1. In CCES, under the Project Explorer window, select Core1 of the desired project by clicking on the project name, and then go to File -> Properties from the menu bar.
2. On the left-hand pane, select, C/C++ Build ->Settings.
3. Click on the Build Artifact tab.
4. Under Artifact Type, select Loader File.

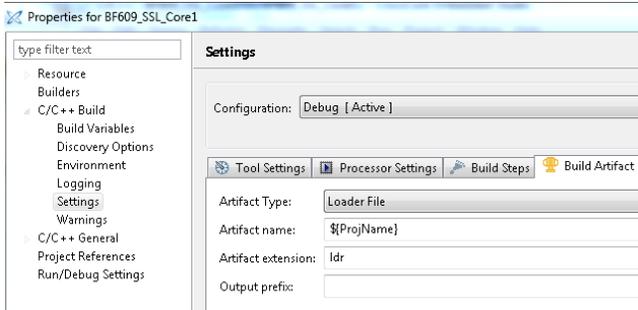


Figure 1. CCES Loader Settings

5. Return to the Tool Settings tab and select the General window under CrossCore Blackfin Loader. Select the following options:

- Boot Mode: SPI0 master
- Boot Format: Binary
- Output Width: 16 bits
- Use default start address checked
- Initialization file (-init). Use the default initialization file provided with CCES. Browse to (default installation path):

C:\Analog Devices\CrossCore Embedded Studio 1.1.0\Blackfin\ldr

Select “BF609_init_v01.dxe”.

The Initialization (init) file is responsible for configuring the registers for external memory accesses during the boot process of your application. The init file is loaded before your application, ensuring that external memory is properly configured before your application makes any attempts to access it.

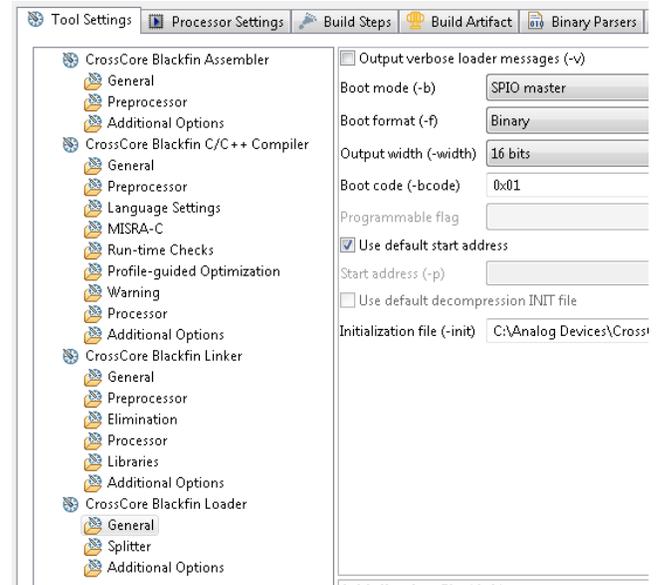


Figure 2. CCES Loader File Options

6. Click on Additional Options of the CrossCore Blackfin Loader.
7. Click on the Add button and add the following onto a single line to generate an LDR that contains both Core0 and Core1:

```
../BF609_SSL_Core0/Debug/BF609_SSL_Core0.dxe
NoFinalTag=../BF609_SSL_Core0/Debug/BF609_SSL_Core0.dxe
```



If your pathnames contain spaces, they must be cared for by employing double-quotes and escape characters, e.g.,
 "C:/Directory\ With\ Spaces/Debug/YourProject_Core0.dxe".

8. Click OK and build the project in the default Debug mode by going to Build->Build Project or by hitting F7 on your keyboard. The generated LDR file will be located in the \Debug folder in your project directory.

Programming Multiple LDR Files to SPI Flash Memory

After building the LDR images for the SSL and each of the applications, they need to be programmed into the flash memory. CCES provides the Command Line Device Programmer (CLDP) to load applications into flash memory. More information on the CLDP can be found in the Help Contents in CCES.

To utilize the CLDP to write the three LDRs of interest to the SPI flash memory, open a command prompt and navigate to the root directory for CCES (default):

```
C:\Analog Devices\CrossCore Embedded Studio x.x.x
```

where x.x.x refers to the tools release version used.

Once there, execute the CLDP three times, as follows:

 The default directory for the device programmer driver is: C:\Analog Devices\ADSP-BF609_Evaluation_Board-Relx.x.x\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial

1. Load BF609_SSL_Core1.ldr into flash memory:

```
cldp -emu kit -proc ADSP-BF609 -cmd prog -driver "C:\Analog Devices\ADSP-BF609_Evaluation_Board-Rel1.1.0\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial\bff609_w25q32bv_dpia.dxe" -format bin -erase all -offset 0 -file "<path name>\BF609_SSL_Core1.ldr"
```

2. Load LED_1.ldr into flash memory:

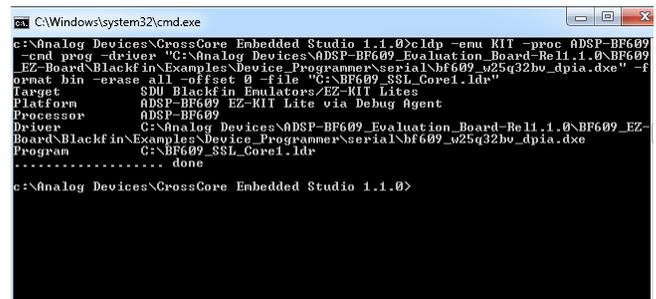
```
cldp -emu kit -proc ADSP-BF609 -cmd prog -driver C:\Analog Devices\ADSP-BF609_Evaluation_Board-Rel1.1.0\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial\bff609_w25q32bv_dpia.dxe" -format bin -erase affected -offset 1048576 -file "<path name>\LED_1.ldr"
```

3. Load LED_2.ldr into flash memory:

```
cldp -emu kit -proc ADSP-BF609 -cmd prog -driver "C:\Analog Devices\ADSP-BF609_Evaluation_Board-Rel1.1.0\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial\bff609_w25q32bv_dpia.dxe" -format bin -erase affected -offset 2097152 -file "<path name>\LED_2.ldr"
```



In the above command lines, <path name> is the full path to the LDR files. Also, -offset must be in decimal form, it cannot be in HEX. 0x100000 is 1048576 in decimal, and 0x200000 is 2097152 in decimal.



```
C:\Windows\system32\cmd.exe
C:\Analog Devices\CrossCore Embedded Studio 1.1.0>cldp -emu kit -proc ADSP-BF609 -cmd prog -driver "C:\Analog Devices\ADSP-BF609_Evaluation_Board-Rel1.1.0\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial\bff609_w25q32bv_dpia.dxe" -format bin -erase all -offset 0 -file "C:\BF609_SSL_Core1.ldr"
Target      C:\Blackfin_Emulators\EZ-KIT_Lite
Platform    ADSP-BF609 EZ-KIT Lite via Debug Agent
Processor   ADSP-BF609
Driver      C:\Analog Devices\ADSP-BF609_Evaluation_Board-Rel1.1.0\BF609_EZ-Board\Blackfin\Examples\Device_Programmer\serial\bff609_w25q32bv_dpia.dxe
Program     C:\BF609_SSL_Core1.ldr
..... done
C:\Analog Devices\CrossCore Embedded Studio 1.1.0>
```

Figure 3. Command Line Device Programmer

Once all three LDR images are programmed into the SPI memory, the example application can be run as intended.

Using the SSL Example Program

After all three LDR files are programmed into flash memory, reset or power-cycle the ADSP-BF609 EZ-KIT board to load and run the SSL. Once the SSL is loaded, the processor idles awaiting user input before continuing with the boot process.

At this point, pushing the push-buttons PB1 or PB2 will select between the two executable files LED_1 and LED_2. Specifically, pressing PB1 will boot the LED_1 application from the SPI flash memory offset 0x100000 (1048576 bytes), whereas pressing PB2 will boot the LED_2

application from SPI flash memory offset 0x200000 (2097152 bytes).

As discussed, the SSL kernel is a temporary application that overwrites itself with the actual end application as part of the two-stage boot process. As such, once the SSL is booted, it will

be destroyed when the processor boots the blink application of choice. At this point, the blink application is resident on the processor until the next reset or power-cycle, at which point the SSL will boot again and await user input.

References

- [1] *ADSP-BF606/ADSP-BF607/ADSP-BF608/ADSP-BF609 Blackfin Dual Core Embedded Processors Data Sheet*. Rev. A, February 2014. Analog Devices, Inc.
- [2] *ADSP-BF60x Blackfin Processor Hardware Reference*. Rev. 0.5, February 2013. Analog Devices, Inc.
- [3] *CrossCore Embedded Studio Loader and Utilities Manual*. Rev. 1.3, May 2014. Analog Devices, Inc.

Document History

Revision	Description
<i>Rev 1 – November 18, 2014 by D. Harrington</i>	Initial Release