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.

In-Circuit Flash Programming on ADSP-2106x SHARC® Processors

Contributed by Jeyanthi Jegadeesan

Rev 2 – March 19, 2007

Introduction

Modern embedded processor systems are equipped with non-volatile memory devices such as EPROM or flash memories. This provides an easy way to later reprogram memory with new content to correct problems in the current firmware or to enhance system capabilities.

With the advent of flash memories, it is now possible to save data permanently and update it when necessary without removing the component from the system. Flash memories are also an asset in systems that need to save data during a power outage or brownout. The processor can store its code/data contents from volatile internal memory to an external, non-volatile memory, and on revival of the system, rewrite the old information back to the processor. ADSP-2106x SHARC® processors can be booted from a single 8-bit-wide memory device like EPROM, EEPROM, or flash.

This EE-Note provides the details of interfacing the flash to the external port on ADSP-2106x processors. It also demonstrates programming the in-circuit flash with two approaches.

The first approach uses the flash programmer driver, which works with the VisualDSP++® Flash Programmer utility. The flash programmer driver code is provided to demonstrate how the flash on the EZ-KIT Lite® board can be programmed for PROM booting. The flash programmer driver code in the associated `.ZIP` file is provided for the ADSP-21065L EZ-KIT Lite board. The flash programmer driver code can be used to program the M29W040B 512-KB flash available from STMicroelectronics. The same code can be modified for all the other ADSP-2106x processors.

The second example code is a flash programmer application that programs the boot image into the flash directly. The example code in the associated `.ZIP` file is available for the ADSP-21061 processor, and the same code can be modified for the other processors.

In-Circuit Flash Programming on SHARC Processors (EE-223)[5] discusses programming the flash on ADSP-2116x, ADSP-2126x, ADSP-2136x, and ADSP-2137x SHARC processors.

Hardware Interface

The flash memory must be interfaced to the BMS space of the processor for booting from the flash. The address bus, data bus, `BMS#` and `RD#` signals must be connected as shown in Figure 1. Data lines `D23:16` of the ADSP-21060, ADSP-21061, and ADSP-21062 processors must be connected to data lines `D7:0` of the flash. For ADSP-21065L processors, data lines `D7:0` must be connected to data lines `D7:0` of the flash. The number of address lines depends on the size of the flash. To program the flash, the `WR#` signal of the processor must to be connected to the `WE#` signal of the flash as shown in Figure 2.

Flash Interface on the ADSP-21065L EZ-KIT Lite Board

The EPROM on the ADSP-21065L EZ-KIT Lite board is mapped to the BMS space of the ADSP-21065L processor. Refer to Figure 1.

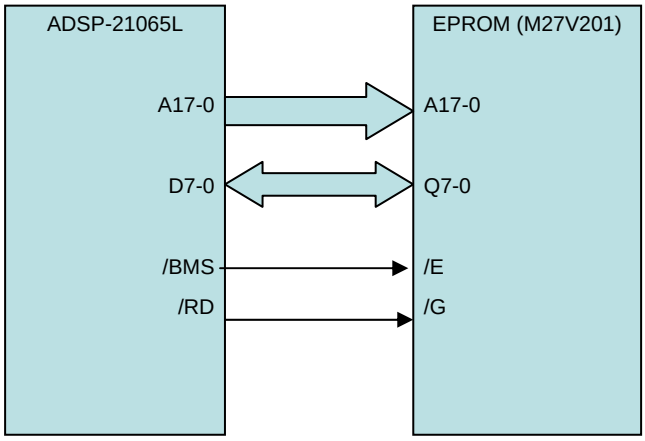

Figure 1. EEPROM interface on the ADSP-21065L EZ-KIT Lite board

The EPROM socket provided on the ADSP-21065L EZ-KIT Lite board cannot be used for the flash as is. Figure 2 shows the connections between the flash and the ADSP-21065L EZ-KIT Lite board.

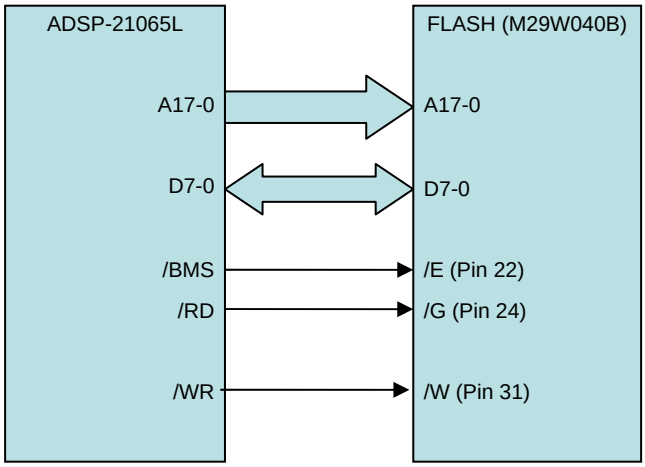

Figure 2. ADSP-21065L flash interface

The `/WR` signal of the ADSP-21065L EZ-KIT Lite must be connected to the `/W` signal of the flash on the EZ-KIT Lite board.

Flash Programmer Driver

The flash programmer driver files included in the associated `.ZIP` file programs the `input.ldr` file into the flash. The flash programmer driver is used with the VisualDSP++ Flash Programmer utility. To program the flash, the flash programmer driver is loaded into memory initially. After loading it into memory, the flash programmer driver:

- Allocates memory to hold the `input.ldr` file contents
- Initializes the sector's start and end addresses for the GUI
- Resets the flash and verifies the device ID

The `input.ldr` file is also loaded into the memory using the Flash Programmer utility. After loading it, the flash programmer driver:

- Erases the flash
- Writes the contents of `input.ldr` into flash
- Verifies the data

The flash programmer driver uses the following functions to write to and read from the flash:

- WriteFlash
- ReadFlash

The `/BMS` signal is used as the chip select signal for the flash. This signal is asserted low by setting the `BSO` (boot select override) bit of the `SYSCON` register before accessing the flash. When the `BSO` bit is set, external port DMA accesses can enable the `/BMS` signal only. The flash programmer driver uses DMA channel 9 to write to and read from the flash. The `WriteFlash` and `ReadFlash` functions set the `BSO` bit of the `SYSCON` register and initiate an external port DMA using channel 9. After initiating the DMA transfer, the processor waits in idle mode for the transfer to end. Once the transfer ends, the processor comes out of idle mode and reset the `BSO` bit of the `SYSCON` register for normal accesses.

Programming the Flash

The flowchart of Figure 3 shows the steps to program code into flash using the Flash Programmer utility.

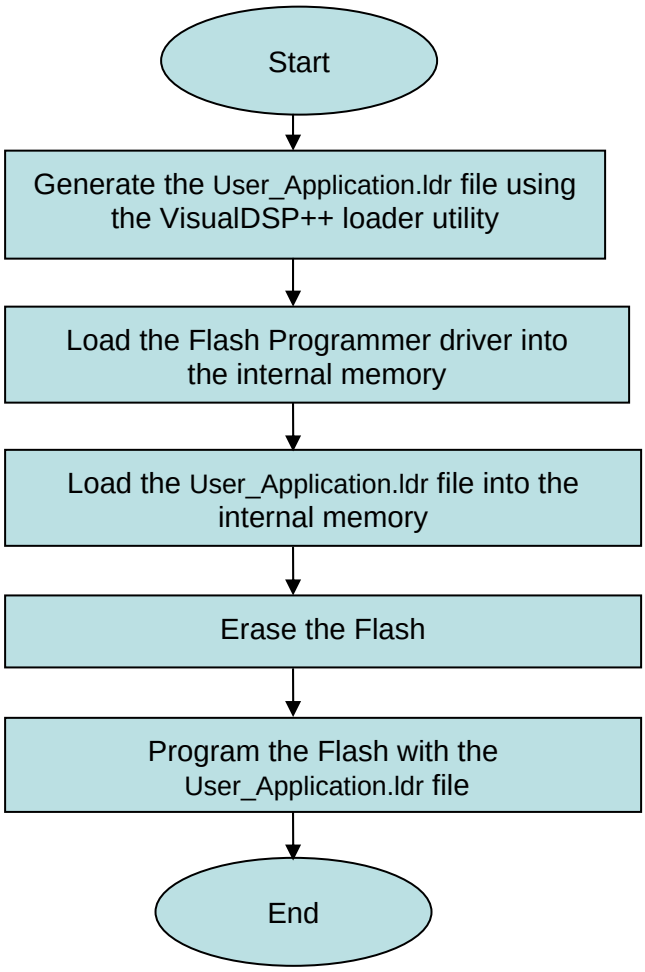

Figure 3. Using the flash programmer driver

The `21065LEZFlash.c` file available with the flash programmer driver contains a generic flash programming algorithm. The blink project files, which toggle the ADSP-21065L processor's flags and blinks the LEDs on the EZ-KIT Lite board, are used as an example. Build and test `user_application.dpj` to program the user application into the flash. Then generate `user_application.ldr` with the VisualDSP++ loader. The flash programmer driver and the data from the `user_application.ldr` file are loaded into internal memory using the Flash Programmer. The flash programmer driver uses the data loaded into internal memory from the `user_application.ldr` file. It erases the flash and then programs the flash with the new data. Once the user application is validated, it can then be programmed into the flash. The example code contains the `Blink.dxe` and `Blink.ldr` files, which can be used to program the flash. The `Blink.exe` file can be loaded and tested on the EZ-KIT Lite board. Then the `Blink.ldr` file can be programmed into the flash. The following two steps describe the generation of the loader file using the VisualDSP++ tools and how to program the flash for the user application using the flash programmer driver.

The flash can also be programmed using the Flash Programmer application provided with this EE-Note. For the Flash Programmer application, the `user_application.ldr` file is used as part of the internal memory. The application erases the flash initially, and then reads the `user_application.ldr` contents from internal memory and programs the flash.

Step 1. Create the PROM Boot Image

Use the VisualDSP++ loader utility to generate the boot-image for the user application code.

To generate the loader file, select `Loader file` as the output file type in the `Project Options` dialog box for `user_application.dpj` (Figure 4).

The Flash Programmer utility accepts the data in hexadecimal format. For the flash programmer project the loader file must be generated with ASCII format.

The `user_application.ldr` file is generated by selecting `Hex` as the boot format as shown in Figure 5. When the project is built using the `Rebuild Project` command (via `Build` menu or toolbar button), the `user_application.ldr` file is generated.

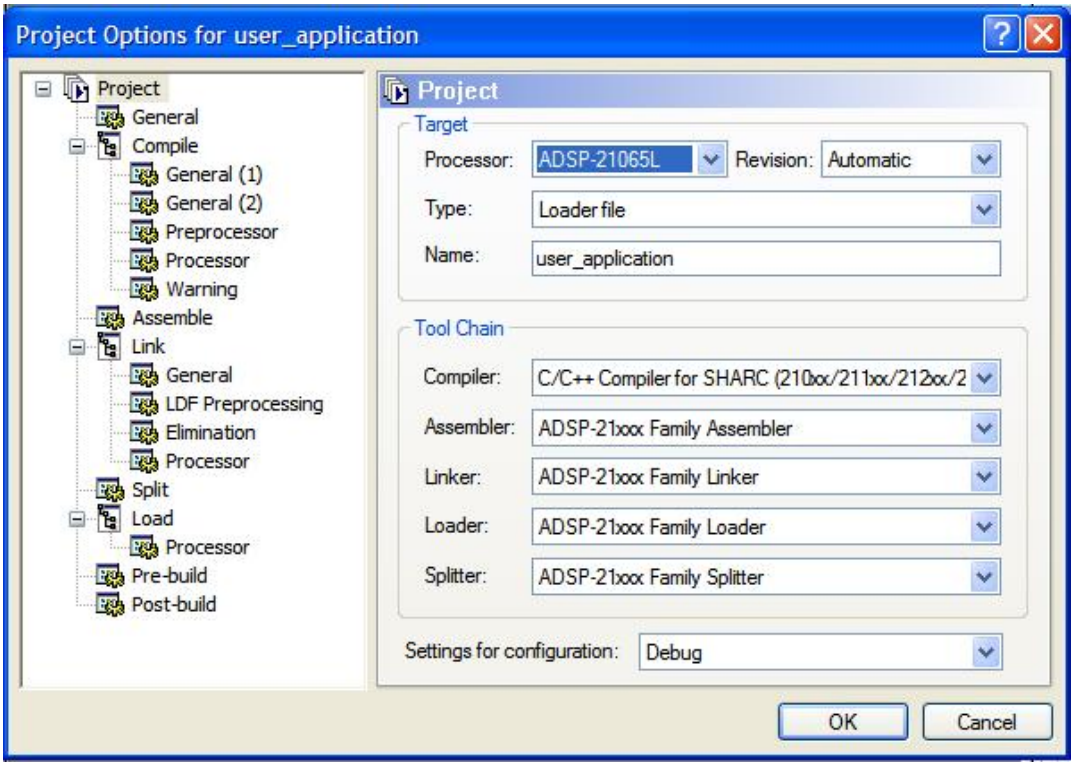

Figure 4. Configuring a loader file via the Project Options dialog box

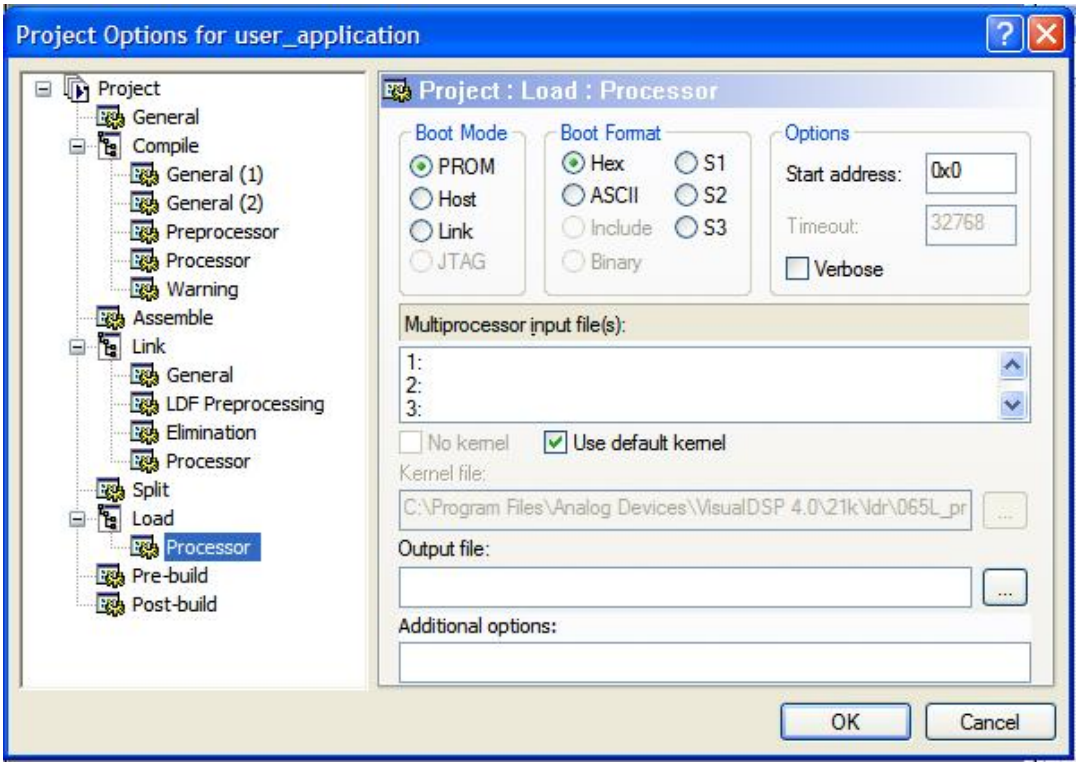

Figure 5. Selecting hex format for the PROM boot

Step 2a. Using the Flash Programmer Utility

Invoke the Flash Programmer utility from the VisualDSP++ `Tools` menu.

Specify the following options, as follows:

1. Click the `Driver` tab. Click the `Browse (…)` button next to the `Load Driver` button (Figure 6) and select the path for the flash programmer driver file.
2. Click the `Load Driver` button to load the flash programmer driver into memory.

   After loading the driver into internal memory, `Success: Driver loaded` is displayed in the `Message center`.

   The manufacturer's code, device code, and part description of the flash are displayed.
3. Click the `Programming` tab. Click the `browse (…)` button under `Data File` (Figure 7) and select the path for the generated `user_application.ldr` file.
4. Click the `Program` button to program the flash with the `user_application.ldr` file.

   When the flash programming is completed, `Success: Erased sector(s)` and `Success: Program complete` are displayed in the `Message center`.

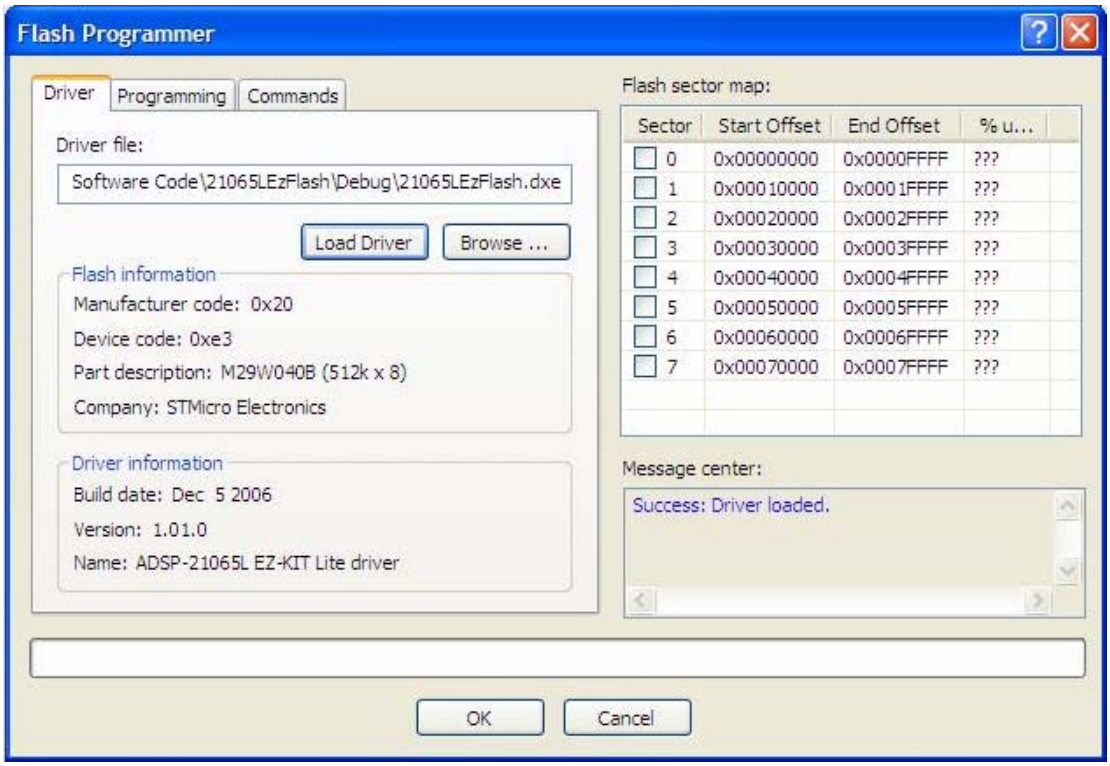

Figure 6. Flash Programmer – loading the flash programmer driver file

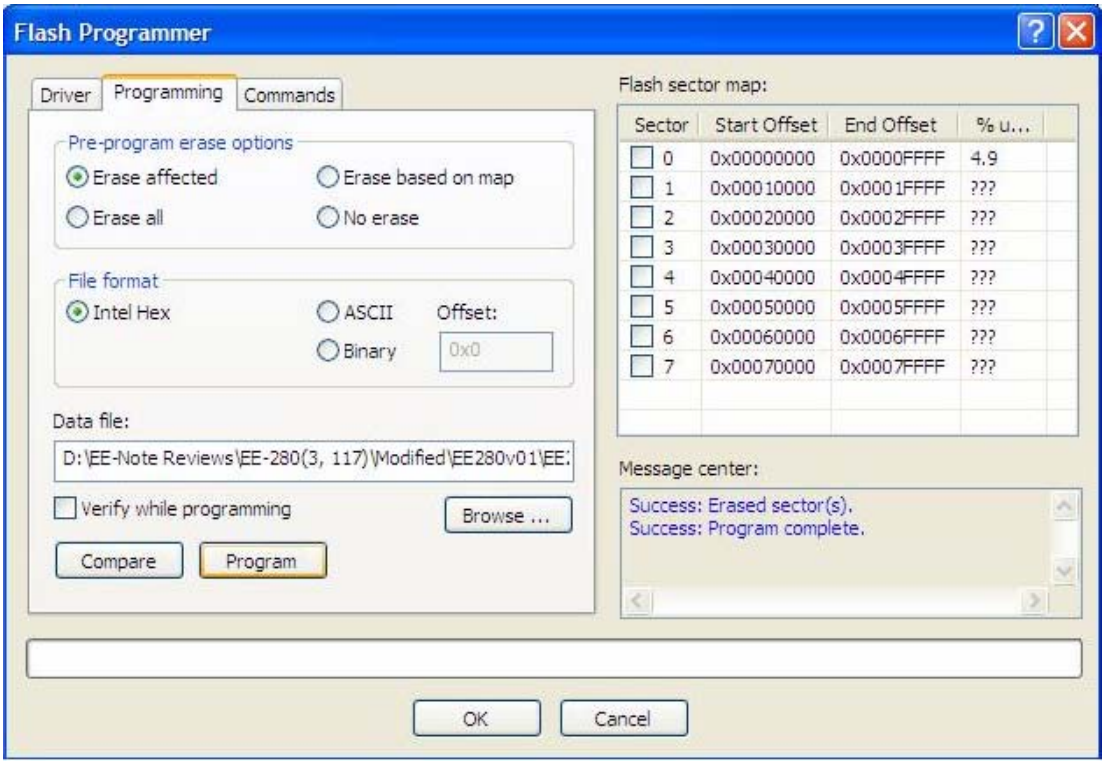

Figure 7. Flash Programmer – loading the loader file for user application

Step 2b. Using the Flash Programmer Application

1. Generate the loader file for the user application for `PROM` boot with `ASCII` format.
2. Copy the flash programmer example code project files provided with the associated `.ZIP` file.
3. Copy the generated loader (`user_application.ldr`) file to the folder where the flash programmer project is placed.
4. Rebuild the flash programmer project files and load the application.
5. Run the application to program the flash. This application erases the flash and programs it with the contents of `user_application.ldr` file.

Booting from Flash Memory

Pins 2 and 3 of jumper `JP6` must be shorted in order to boot from the flash on the ADSP-21065L EZ-KIT Lite board. If the jumper is not connected, the code boots from the flash.

After programming the code on the flash, applying a power reset causes a boot from the flash.

The Blink example code can be booted from the flash and verified. Program the flash with the `Blink.ldr` file using the Flash Programmer utility. When the power reset is applied to the board, the Blink example code boots from the flash and toggles the LEDs on the EZ-KIT Lite board.

Modifying the Flash Programmer Driver for Different Flash Parts

The flash programmer driver available with this EE-Note is implemented for a STMicroelectronics M29W040B flash device. The code can be modified for other flash devices.

The flash programmer driver calls the following functions in the given sequence:

- GetSectorStartEnd()
- SetupForFlash()
- GetCodes()
- EraseFlash() or EraseBlock()
- WriteData()
- UnlockFlash()

The following device-dependant functions must be modified:

- ResetFlash()
- EraseFlash()
- EraseBlock()
- GetCodes()
- UnlockFlash()
- GetSectorStartEnd()
- GetSectorNumber()

The `GetSectorStartEnd()` and the `GetSectorNumber()` functions depend on the number of sectors and the sector size on the device. The `NUM_SECTORS` can be modified with the available sectors. The `AFP_SectorSize` can be modified with the sector size. If the flash has variable sector size, the `GetSectorNumber()` and `GetSectorStartEnd()` function implementation must be modified.

Listing 1 shows the `UnlockFlash()` function. The commands used in this function may vary from device to device and must be modified.

```
ERROR_CODE UnlockFlash()
{
  int flash_address;
  /* Write commands to unlock the
     flash. */
  flash_address = 0x5555;
  flash_data = 0xaa;
  WriteFlash( flash_address,
              &flash_data);
  flash_address = 0x2aaa;
  flash_data = 0x55;
  WriteFlash( flash_address,
              &flash_data);
  flash_address = 0x5555;
  flash_data = 0xa0;
  WriteFlash( flash_address,
              &flash_data);
  return NO_ERR;
}
```

Listing 1. UnlockFlash() function

Summary

With minimal hardware modification to the EZ-KIT Lite board, the board's flash can be programmed easily. The flash programmer driver files and the flash programmer application can be modified for different flash devices.

Software Code

The source codes for the flash programmer driver and for the flash programmer application are supplied in the `.ZIP` file associated with this EE-Note.

References

[1] *ADSP-21065L SHARC DSP User's Manual*. Rev 2.0, July 2003. Analog Devices, Inc.

[2] *ADSP-2106x SHARC Processor User's Manual*. Rev 2.1, March 2004. Analog Devices, Inc.

[3] *ADSP-21065L EZ-KIT Lite Evaluation System Manual*. Rev 2.0, January 2003. Analog Devices, Inc.

[4] *ADSP-21061 EZ-KIT Lite Evaluation System Manual*. Rev 3.0, January 2003. Analog Devices, Inc.

[5] *In-Circuit Flash Programming on SHARC Processors (EE-223).* Rev 2, February 2007, Analog Devices Inc.

Document History

Revision	Description
Rev 2 – March 19, 2007 *by Jeyanthi Jegadeesan*	Generalized the EE-Note for all ADSP-2106x processors and changed the title from *In-circuit Flash Programming on ADSP-21065L SHARC EZ-KIT Lite Boards* to *In-Circuit Flash Programming on ADSP-2106x SHARC Processors*. Merged with the contents from *Writing to Flash Memory on the ADSP-2106x (EE-3)*, *Interfacing Byte Programmed FLASH Memory to the ADSP-2106x SHARC series (EE-55)*, and *In-System-Programming (ISP) of ADSP-2106x boot-images into FLASH Memories (EE-117)*.
Rev 1 – December 15, 2005 *by Jeyanthi Jegadeesan*	Initial release for the ADSP-21065L processor.