DRAFT

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.

ADSP-BF50xF FLASH Tutorial

Contributed by mweiner

Rev 3 – March 15, 2010

Introduction

The ADSP-BF50x processor series differs from previous Blackfin processors for development and debugging of code since there is no access to external SDRAM. The ADSP-BF504F and ADSP-BF506F contain a stacked parallel FLASH. Internal memory and FLASH can be used simultaneously for effective debugging. This tutorial demonstrates the creation of a VisualDSP++ project for effective debugging using internal memory and FLASH.

Problem

This tutorial provides solutions for the following:

- Setting up a project to properly build for L1 Internal Memory and FLASH.
- Successfully loading program code into L1 Internal Memory and FLASH.

Creating a Project for Internal Memory and FLASH

This section explains how to create a simple project that allocates code into the appropriate L1 and FLASH sections.

1. Ensure that VisualDSP++ 5.0 ADSP-BF506F EZ-KIT is installed.
2. Create a new session
    a. Click *Session-New Session…*
    b. Select *ADSP-BF504F* or *ADSP-BF506F* and click *Next*
    c. Select *EZ-KIT Lite* or *Emulator* as appropriate and click *Next*
    d. Select the appropriate platform and click *Finish*
3. Create a new project
    a. Click *File-New Project…*
    b. Select *Standard application*, type in the name *FLASH Tutorial* and click *Next*
    c. Click *Yes* to create a new directory
    d. Click *Next* to select the appropriate processor

Copyright 2010, Analog Devices, Inc. All rights reserved. Analog Devices assumes no responsibility for customer product design or the use or application of customers' products or for any infringements of patents or rights of others which may result from Analog Devices assistance. All trademarks and logos are property of their respective holders. Information furnished by Analog Devices applications and development tools engineers is believed to be accurate and reliable, however no responsibility is assumed by Analog Devices regarding technical accuracy and topicality of the content provided in Analog Devices Engineer-to-Engineer Notes.

e. Click *Next* on the *Applications Settings* window

f. Select *Add an LDF and startup code* and click *Finish*

4. Add code to L1 and FLASH

   a. In the project window, double-click *FLASH Tutorial.c*

   b. Replace the program with the following text:

Please note that using puts() instead of printf() can reduce code size.

```
/*****************************************************************************
 * FLASH Tutorial.c
 *****************************************************************************/
#include <stdio.h>

#pragma section("L1_code")
void routine_in_L1 (void )
{
    /* Place L1 code here */
    printf ("This code is in L1.\n");
    return;
}

#pragma section("FLASH_code")
void routine_in_FLASH ( void )
{
    /* Place FLASH code here */
    printf ("This code is in FLASH.\n");
    return;
}

int main( void )
{
    routine_in_L1();
    routine_in_FLASH();
    routine_in_L1();
    routine_in_FLASH();
    return 0;
}
```

Loading into Internal Memory and FLASH

VisualDSP++ now automatically uses the Flash Programmer to program the FLASH prior to loading the L1 SRAM code and data. No special steps are required to load a program to both FLASH and L1 SRAM. Loading takes place automatically as part of the build process. The following can be done to verify that code is properly loaded into both FLASH and L1 SRAM and executes appropriately.

1. Set up breakpoints

a. On line 10 (*printf()*), click *F9* to set a software breakpoint in internal memory.

b. On line 18 (printf()), click *Shift-F9* to set a hardware breakpoint in FLASH. Please note that software breakpoints cannot be set in FLASH.

2. Click *F7* or select *Project – Build Project* to build the project
3. Click *F5* a few times to run, watching the disassembly to see the code in internal memory and FLASH. The program is working if you see the correct print out in the console window.

Debugging Internal Memory and FLASH

Debugging in both internal memory and FLASH is fairly straight forward as long as you keep a few things in mind. First, the ADSP-BF50x processor series has 6 Watchpoints allowing for only 6 Hardware Breakpoints. There is an unlimited number of Software Breakpoints. Second, only Hardware Breakpoints will halt the processor in FLASH. Because of this, try to place the program that you are debugging in L1 and stable code in FLASH. This will also reduce the frequency of programming the FLASH. Similarly, single stepping into code in FLASH is possible but running to the cursor is not.

main()

Code with automatic breakpoints or software breakpoints must reside in L1 SRAM. In addition, all functions containing a definition of main() will be placed in L1(). To all greater flexibility in placing functions in FLASH, it is recommended to place main() in a separate source file and keep the size to a minimum.

Document History

Revision	Description
Rev 4 – March 4, 2010 by M. Weiner	Revised for new feature allowing for easier loading to FLASH and L1 SRAM.
Rev 3 – January 19, 2010 by M. Weiner	Revised for the VisualDSP++ 5.0 ADSP-BF506F EZ-KIT release.
Rev 2 – December 8, 2009 by M. Weiner	Increased Mask from 16 to 21 to allow the entire FLASH to be written.
Rev 1 – December 7, 2009 by M. Weiner	Initial Draft Release.