

Outline: *System Development and Programming with the ADSP-BF535*

Course Name:	System Development and Programming with the ADSP-BF535
Course Number:	ADST-150
Course Description:	<p>This is a practical course with ‘hands on’ training using the latest software development tools. First the core elements of the Blackfin DSP, which includes the Computational Units, the Data Address Generators, and the Program Sequencer, are examined in detail along with the relevant assembly code instructions. A number of simulator labs help in understanding operation of the individual elements. Memory configuration (both internal and external) is discussed next. Advanced instructions are presented with a follow on lab on code optimization. The I/O peripherals, which include the SPORTS, SPI, UART, Host Port, and External Port, are discussed in detail along with DMA operation between these peripherals and internal memory. System related information such as booting, operation of the PC133 SDRAM controller, PCI/USB interfaces, RTC, power management, and other hardware design issues are also discussed. Hardware development tools, such as evaluation boards and ICE’s are introduced. Throughout the course, the various aspects of the software development process using the latest tools are discussed including setting up and building projects, assembly language programming, code debugging, simulation, tool support for code overlays and shared memory, and ‘C’ programming support.</p>
Goals/Objectives:	<p>The main course objective is to understand the architecture of the ADSP-BF535 DSP sufficiently to enable DSP system designers to resolve hardware/software issues with their applications. Additional goals include gaining a thorough understanding of both assembly language programming and code development (including ‘C’ programming issues) with the latest software tools</p>
Pre-requisites:	Previous embedded microprocessor background would be an asset (hardware and/or software)
Target Audience:	System Designers needing to make informed decisions on design tradeoffs, Hardware Designers needing to develop external interfaces, and Code Developers needing to know how to get the highest performance from their algorithms
Target Duration:	3.5 days

1 Introduction

1.1 Introduction/ Course Overview

- 1.1.1 Goal of Workshop**
- 1.1.2 Course Overview**
- 1.1.3 Course Handouts**
- 1.1.4 Characteristics of a Good DSP**
- 1.1.5 ADSP-BF535 Architecture Overview**
- 1.1.6 Clock Domains**

2 Introduction to Software Tools

- 2.1.1 Software Development Process**
- 2.1.2 VisualDSP++**
- 2.1.3 Assembler**
- 2.1.4 Linker**
- 2.1.5 Loader**
- 2.1.6 Integrated Development and Debugging Environment (IDDE)**
- 2.1.7 VisualDSP Software Debugger**

3 Registers and Number Formats

- 3.1.1 Fixed Point Numbering Formats**
- 3.1.2 Register Types and Naming Conventions**
- 3.1.3 Register File**

4 Arithmetic Units

4.1 Arithmetic Logic Unit (ALU)

4.1.1 Features

4.1.2 Instructions

4.1.3 Flags

4.1.4 Simulator Exercise

4.2 Multiplier/ Accumulator (MAC)

4.2.1 Features

4.2.2 Instructions

4.2.3 Simulator Exercise

4.3 Shifter

4.3.1 Features

4.3.2 Instructions

4.3.3 Simulator Exercise

5 Addressing Modes

5.1 Data Address Generators (DAG)Features

5.2 Assembly Instructions

5.3 Data Moves

5.4 Circular Buffering

5.5 Bit Reversal

5.6 Simulator Exercise

6 Memory

6.1 Features

6.2 Memory Map

6.3 Internal Memory Architecture

6.3.1 L1 Memory

6.3.2 Cache Operation

6.3.3 Memory Protection

6.3.4 L2 Memory

6.4 External Memory

7 Program Sequencer

7.1 Features

7.2 Assembly Instructions

7.3 Conditional Sequencing

7.4 Instruction Pipe Line

7.5 Branching

7.6 Zero Overhead Looping

7.7 Event Controller

7.7.1 Interrupt Handling

7.7.2 Exception Handling

8 Program Development

8.1 IDDE and Projects

8.2 Assembler

8.2.1 Features and Overview

8.2.2 Assembler Expressions

8.2.3 Assembler Directives

8.2.4 Definition Files

8.3 Linker / Linker Description File (LDF)

8.3.1 Features and Overview

8.3.2 Example LDF

8.3.3 LDF Commands

8.4 Using the Simulator for Code Debug

8.4.1 Features and Overview

8.4.2 Invoking the Simulator

8.5 Assembly Code Exercise

9 Advanced Instructions

9.1 8-bit ALU Instructions

9.2 Vector Operations

9.3 Issuing Parallel Instructions

9.4 Code Optimization

9.5 Optimization Exercise

10 Timers and Flags**10.1 Timers****10.2 Real Time Clock****10.3 Watch Dog Timer****10.4 Programmable Flags****11 Direct Memory Access (DMA)****11.1 Overview****11.2 Descriptor based DMA****11.3 Autobuffer based DMA****11.4 Memory DMA****12 Parallel Interface****12.1 External Bus Interface Unit (EBIU)****12.1.1 Features and Overview****12.1.2 External Memory Interface****12.2 Asynch Memory Controller****12.3 SDRAM Controller**

13 Serial Interfaces

13.1 Serial Port (SPORT)

13.1.1 Features

13.1.2 Pin Descriptions

13.1.3 Modes of Operation

13.1.4 Configuration

13.2 Serial Peripheral Interface (SPI)

13.2.1 Features and Setup

13.3 UART

13.3.1 Features and Setup

14 USB

14.1 What is USB?

14.2 Features

14.3 Using USB on the ADSP-BF535

15 PCI

15.1 What is PCI

15.2 Features

15.3 Using PCI on the ADSP-BF535

16 System Booting

- 16.1 Booting Modes**
- 16.2 Boot Kernel (2nd Stage Loader)**
- 16.3 VisualDSP Loader Utility**
- 16.4 Loader file formats**

17 System Design

- 17.1 Operating modes**
- 17.2 Dynamic Power Management**
 - 17.2.1 Clock generation / PLL**
 - 17.2.2 Power-down modes**
- 17.3 Resetting the DSP**
 - 17.3.1 Reset Types**
 - 17.3.2 Power-up sequence**
- 17.4 JTAG Overview**
 - 17.4.1 ICE Header**
- 17.5 Board Design and Layout**
 - 17.5.1 Navigating the Datasheet**
 - 17.5.2 Anomaly Sheet**
 - 17.5.3 Design Guidelines**

18 C/C++ Compiler

- 18.1 Features**
- 18.2 Configuration**
- 18.3 LDF Related issues**
- 18.4 Register Usage**
- 18.5 Run Time Environment**
 - 18.5.1 Stack and HEAP**
 - 18.5.2 Interrupt Handling**
 - 18.5.3 Assembly Code Interface**
- 18.6 Run-time Library Overview**
- 18.7 C++ capabilities**
- 18.8 C Code Optimization**

19 ADSP-BF535 Hardware Tools

- 19.1 Overview with Part Numbers**
- 19.2 ADSP-BF535 EZ-Kit**
- 19.3 In Circuit Emulators (ICE)**