



System Development and Programming with the ADI Blackfin Processor Family

| | |
|-----------------------|---|
| Course Name: | System Development and Programming with the ADI Blackfin Processor Family |
| Course Code: | WS_BFDEV |
| Course Description: | <p>This is a practical and interactive course that is designed to systematically teach how to use the Blackfin processor to its fullest potential. Emphasis is placed on understanding the steps required to create an efficient Blackfin CPU based system in the way that ADI had intended the processor to be used. Several hands on exercises provide an opportunity for the instructor to work one on one with the attendee. Throughout the workshop, attendees are encouraged to ask questions.</p> <p>The VisualDSP++ IDDE is covered in detail, including topics on projects and project configuration, the build process, and debug features. Tools based optimizations including compiler and linker optimization are covered. The salient features of System Services and Device Drivers are first introduced. Later sections on specific peripherals or system features will connect back to the related Services or Driver API. An understanding of the Blackfin architecture will enable getting the best performance out of the processor. Architecture topics covered include loop/branch optimization and interrupt handling, L1 Memory configuration (ie L1 SRAM and Cache), specialized instructions including the quad 8-bit Video ALU operations, and DMA operation between peripherals and memory, as well as from memory to memory. The usage and capabilities of the various I/O and timer peripherals are discussed in detail. A section on booting covers what happens during the boot process, creating boot image files, and discussing how to get them into the target system. Hardware considerations such as operation of the power management and reset are also discussed. Hardware development tools, such as evaluation boards and ICE's are also covered, including setting up hardware debug sessions and other hardware debug topics. An introduction into VDK (Visual DSP Kernel) is also covered in the workshop.</p> <p>Throughout the course, a number of hands on exercises will take the attendee through the various aspects of the software development process. Topics covered through exercises include setting up and building projects, C language programming (eg using intrinsics, exploring different data types), various optimizations (eg compiler, linker, mixed C/assembly), code debugging (eg profiling, plotting, pipeline analysis), simulation (eg DMA, interrupts, stream I/O), and 'C' programming support (eg System Services and Device Drivers). One set of exercises will focus on bringing the various elements together (eg peripheral setup, DMA, interrupts, etc) in the setup of a typical application framework, initially in straight C and later using the System Services and Device Drives programming model.</p> |
| Goals and Objectives: | The main course objective is to instill to the attendee a high level of confidence to create an efficient Blackfin® Processor based system. By being exposed to the various capabilities and features of the Blackfin processor and VDSP tool chain, the attendee will be better armed to tackle any issues that arise in their specific system implementations. |
| Pre-requisites: | Previous embedded microprocessor experience would be an asset (hardware and/or software). It is also highly recommended to take in the Visual Learning & Development (VLD) Video Tutorial entitled "Blackfin® Core Architecture" |
| Target Audience: | System Designers needing to make informed decisions on design tradeoffs, Hardware Designers needing to develop external interfaces and low level code, and Code Developers needing to know how to get the highest performance from their algorithms |
| Duration: | 3.5 days |



- 1 Introduction**
 - 1.1 Goal of Workshop**
 - 1.2 Course Overview**
 - 1.3 Course Handouts**
 - 1.4 Kaztek Systems Overview**
 - 1.5 Blackfin Architecture Overview**
 - 1.6 VisualDSP++ Development Tools Overview**

- 2 Introduction to Software Tools**
 - 2.1 VisualDSP++ Overview**
 - 2.1.1 Software Development Process**
 - 2.2 IDDE and Projects**
 - 2.2.1 Creating projects and the Project Wizard**
 - 2.2.2 Project Options and Build Configurations**
 - 2.3 Invoking the Tools**
 - 2.4 Debug Sessions**
 - 2.5 Exercise – “Hello World”**
 - 2.6 Debugger Features**
 - 2.6.1 Debug control**
 - 2.6.2 Debug windows**
 - 2.6.3 Profiling**
 - 2.7 Help**
 - 2.8 Exercise – “Sorts”**



-
- 3 Data Formats**
 - 3.1 Supported Data Types**
 - 3.2 Fixed Point Numbering Formats**
 - 3.2.1 Built-in Support for Fractional Data Types**
 - 3.3 Floating Point**
 - 3.4 Exercise – Floating Point Operations**

 - 4 Blackfin Core Review**
 - 4.1 Registers**
 - 4.1.1 Core Registers**
 - 4.1.2 Memory Mapped Registers**
 - 4.1.3 Register File**
 - 4.2 Arithmetic Units**
 - 4.2.1 Features**
 - 4.2.2 Arithmetic Logic Unit (ALU)**
 - 4.2.3 Multiplier/ Accumulator (MAC)**
 - 4.2.4 Shifter**
 - 4.2.5 Example Instructions**

 - 4.3 Fetching Data**
 - 4.3.1 Data Address Generators (DAG) Features**
 - 4.3.2 Data Move Examples**
 - 4.3.3 Address Pointer Manipulation**
 - 4.3.4 Stack Operations**
 - 4.3.5 DSP Addressing**
 - 4.3.5.1 Circular Buffering**
 - 4.3.5.2 Bit Reversal**

 - 4.4 8-bit ALU Instructions**
 - 4.4.1.1 Compiler builtin support for Video ALU**
 - 4.5 Vector Operations**
 - 4.6 Issuing Parallel Instructions**
 - 4.7 Other Code Optimization Topics**
 - 4.8 Instructor Led Assembly Code Exercise**



5 Memory

5.1 Features

- 5.1.1 Memory Levels**
- 5.1.2 Configurable L1 Memory**
- 5.1.3 Memory Maps**

5.2 L1 Memory

- 5.2.1 Internal Memory Architecture**
 - 5.2.1.1 Configurable Memory Blocks (Cache/SRAM)**
- 5.2.2 Instruction Memory and Control Registers**
- 5.2.3 Data Memory and Control Registers**

5.3 Cache Usage

5.4 Memory Protection

- 5.4.1 CPLBs (Cachability Protection Lookaside Buffers)**

6 Program Sequencer

6.1 Features

6.2 Conditional Sequencing

6.3 Instruction Pipe Line

- 6.3.1 Synchronizing pipeline to Core/System events**
- 6.3.2 Stalls and avoiding them**
- 6.3.3 Pipeline viewer**

6.4 Branching

- 6.4.1 Conditional vs Unconditional**
- 6.4.2 Zero Overhead Looping**
- 6.4.3 Compiler branch/loop optimizations**

6.5 Event Controller

- 6.5.1 Interrupt vs Exception Handling**
- 6.5.2 CEC (Core Event Controller) and SIC (System Interrupt Controller)**
- 6.5.3 Interrupt processing flow**
- 6.5.4 Interrupt handling in C**
- 6.5.5 Exercise – Interrupt Handling in C**



7 Linker Operations

7.1 Converting C and Assembly files to Object files

- 7.1.1 Features and Overview**
- 7.1.2 Assembler Expressions and Directives**
- 7.1.3 Object Sections**
- 7.1.4 DefBF5xx.h Files**

7.2 Linker / Linker Description File (LDF)

- 7.2.1 Features and Overview**
- 7.2.2 LDF Commands**
- 7.2.3 Example LDF**
- 7.2.4 Linker Optimizations**
- 7.2.5 Expert Linker**
- 7.2.6 The Linker and the C Run Time Environment**

7.3 Exercise – Fract Arithmetic project

8 System Services and Device Drivers

8.1 What are they?

8.2 Services Overview

- 8.2.1 Interrupt Manager**
- 8.2.2 EBIU Service**
- 8.2.3 Dynamic Power Management Service**
- 8.2.4 Port Control Service**
- 8.2.5 Deferred Callback Manager**
- 8.2.6 DMA Manager**
- 8.2.7 Programmable Flag Service**
- 8.2.8 Timer Service**
- 8.2.9 RTC (Real Time Clock) Service**
- 8.2.10 File System Service**
- 8.2.11 Semaphore Service**



8.3 Drivers Overview

- 8.3.1 Device Driver API**
- 8.3.2 Relationship to Services**
- 8.3.3 Data Buffers**
- 8.3.4 Data Flow Types**
- 8.3.5 Off Chip Drivers**

8.4 Finding Additional Information

9 Using System Services and Device Drivers

9.1 Project considerations when using Device Drivers

- 9.1.1 Using adi_ssl_Init.c/h**
- 9.1.2 Memory requirements**
- 9.1.3 Interpreting Result Codes**

9.2 Detailed example walkthrough

- 9.2.1 Initialization of the various services**
- 9.2.2 Customization considerations for custom hardware**
- 9.2.3 Setting up deferred callbacks**
- 9.2.4 Opening a device driver and peripheral setup**
- 9.2.5 Structure of the callback function**

10 Timers and Flags

10.1 Timer Overview

- 10.1.1 Timer API under System services**

10.2 Core Clock

10.3 SSL Exercise

10.4 Real Time Clock

10.5 Watch Dog Timer

10.6 GP Timers

- 10.6.1 PWM Mode**

- 10.6.2 Pulse Capture Mode**

- 10.6.3 External Clock Mode**

10.7 Programmable Flags Overview

- 10.7.1 Multiplexing with peripheral functions**

10.8 ADSP-BF537 IO Ports

- 10.8.1 Configuration**

- 10.8.2 Port support under System Services**



11 Direct Memory Access (DMA)

- 11.1 Overview**
- 11.2 DMA channel priorities**
- 11.3 DMA Mapping with SSL**
- 11.4 Setting up DMAs**
 - 11.4.1 Register based DMA**
 - 11.4.2 Descriptor based DMA**
 - 11.4.3 SSL Data Structures**
- 11.5 Peripheral ⇔ Memory transfers**
- 11.6 Memory ⇔ Memory transfers**
 - 11.6.1 Opening a Memory DMA Channel with SSL**
- 11.7 DMA Traffic control**
- 11.8 2D DMA**
- 11.9 SSL MemDMA Exercise**

12 External Bus Interface Unit (EBIU)

- 12.1 Features and Overview**
- 12.2 External Memory Interface**
 - 12.2.1 Asynch Memory Controller**
 - 12.2.2 SDRAM Controller**
- 12.3 Performance**

13 Serial Communications

- 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**
- 13.4 Two Wire Interface (TWI)**
 - 13.4.1 Overview**
- 13.5 Controller Area Network (CAN)**
 - 13.5.1 Overview**



14 PPI

- 14.1 What is PPI?**
- 14.2 Operating Modes**
- 14.3 Video Basics**

15 Ethernet

- 15.1 Overview**
- 15.2 Blackfin® Ethernet MAC Features and Overview**
- 15.3 System interface considerations**
- 15.4 Developing applications with lwIP protocol stack**
- 15.5 Configuring lwIP**
- 15.6 Example application**

16 System Booting

- 16.1 Booting Methods**
- 16.2 Loader file formats**
- 16.3 Boot time hardware configuration**
- 16.4 Multi Application Booting**
- 16.5 VisualDSP Loader Utility**
- 16.6 Flash Programmer Utility**

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**
- 17.4 JTAG Overview**
- 17.5 Board Design and Layout**
- 17.6 Debug Registers**



18 Optimization Topics

- 18.1 Overview**
- 18.2 General Approach to Optimization**
- 18.3 Algorithmic considerations**
- 18.4 Getting to know the Compiler**
 - 18.4.1 Optimization Switches and pragma's**
 - 18.4.2 Built-in functions**
 - 18.4.3 Example compiler output**
 - 18.4.4 PGO (Profile Guided Optimization)**
 - 18.4.5 C++ considerations**
 - 18.4.6 Summary**
- 18.5 Where to start optimizing?**
 - 18.5.1 Using the profiler effectively**
- 18.6 Assembly Language Interfacing**
- 18.7 Video Frameworks**
 - 18.7.1 Challenges**
 - 18.7.2 Common Frameworks**

19 Hardware Tools

- 19.1 EZKITs**
 - 19.1.1 Overview with Part Numbers**
 - 19.1.2 EZKIT Extender Boards**
 - 19.1.3 EZKIT Debug Sessions**
 - 19.1.4 Application examples**
- 19.2 In Circuit Emulators (ICE)**
 - 19.2.1 Configuration of a Debug Target**
 - 19.2.2 Emulator Debug Sessions**



www.kaztek.com

WS_BFDEV

System Development and Programming with the ADI Blackfin Processor Family

20 VDK Introduction

- 20.1 Threads and scheduling**
- 20.2 VDK projects**
- 20.3 Signaling and synchronization**
- 20.4 SSL Device Drivers and VDK**
- 20.5 Memory pools and messaging**
- 20.6 Multiple heaps**
- 20.7 VDK Conventions**
- 20.8 Error handling**
- 20.9 Debug assistance**