



www.kaztek.com

WS_CCESSH

System Development with CCES and the ADI SHARC Processor

Course Name:	System Development with CrossCore Embedded Studio (CCES) and the ADI SHARC Processor
Course Code:	WS_CCESSH
Course Description:	<p>This is a practical and interactive course that is designed to systematically teach how to use the SHARC processor to its fullest potential. Emphasis is placed on understanding the steps required to create an efficient SHARC 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 CrossCore Embedded Studio (CCES) IDE is covered in detail, including topics on navigating through the IDE, projects and project configuration, the build process, and debug features. Tools based optimizations including compiler and linker optimization are covered.</p> <p>An understanding of the SHARC architecture will enable getting the best performance out of the processor. Architecture topics covered include the core elements of the SHARC, which includes the Computational Units, the Data Address Generators, and the Program Sequencer. The core section is common to all members of the SHARC family. Memory configuration (both internal and external) is discussed next. Advanced instructions and SIMD operation are presented with a follow on lab on code optimization. The I/O peripherals, which include the DAI and DPI, are discussed in detail along with DMA operation between these peripherals and internal memory. 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 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 Micrium's uC/OS 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 and Assembly language programming, various optimizations, code debugging (simulation and hardware), and software module support (eg System Services and Device Drivers).</p>
Goals and Objectives:	The main course objective is to understand the SHARC architecture (with a focus on the ADSP-2136x/37x/4xx Sharc Processor Family members) sufficiently to enable DSP system designers to resolve hardware/software issues with their applications. Additional goals include gaining a thorough understanding of SHARC code development (using both assembly and 'C') using the CCES tool chain.
Pre-requisites:	Previous embedded microprocessor experience 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 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 Workshop outline**
 - 1.2 CCES Highlights**
 - 1.3 Sharc Architecture Overview**
 - 1.3.1 ADSP-2136x**
 - 1.3.2 ADSP-214xx**

- 2 Introduction to CCES**
 - 2.1 CCES IDE Overview**
 - 2.1.1 IDE Concepts**
 - 2.1.2 Navigating the IDE**
 - 2.1.3 Help**
 - 2.2 IDE and Projects**
 - 2.2.1 Creating projects and the Project Wizard**
 - 2.2.2 Project Options and Build Configurations**
 - 2.2.3 Project Structure**
 - 2.2.4 Invoking the Build Tools**
 - 2.3 Exercise – “Hello World”**
 - 2.4 Debug Sessions**
 - 2.4.1 Types of Debug Sessions**
 - 2.4.2 Creating/Launching Debug Sessions**
 - 2.4.3 Configuring a Debug Session**
 - 2.5 Debugger Features**
 - 2.5.1 Debug control**
 - 2.5.2 Debug windows**
 - 2.6 Exercise – “FIR Filter”**
 - 2.6.1 Compiler Optimization topics**
 - 2.6.2 Performance assessment topics**



3 Sharc Core Review

3.1 Overview

- 3.1.1 Registers**
- 3.1.2 Arithmetic Units**
- 3.1.3 Fetching Data**
- 3.1.4 Assembly Code Syntax**

3.2 Assembly Code Example

- 3.2.1 Source code structure**
- 3.2.2 Issuing Parallel Instructions**

3.3 Optional Simulator Exercise: registers, basic simulator, MAC, ALU, Shifter operation

4 Memory Architecture

4.1 SHARC Memory

- 4.1.1 SHARC Memory Basics**
- 4.1.2 SHARC Memory Map**
- 4.1.3 SHARC Internal Architecture**

4.2 SHARC Internal SRAM

- 4.2.1 Internal SRAM Architecture**
- 4.2.2 Memory Overhead Considerations**
- 4.2.3 Internal Memory Maps**
- 4.2.4 Configuring Internal Memory**
- 4.2.5 Example LDF Memory Section**



5 Program Sequencer and Core Timer

5.1 Program Sequencer

5.1.1 Features

- 5.1.1.1 Instructions**
- 5.1.1.2 Instruction pipeline**
- 5.1.1.3 Branching, Delayed branching**
- 5.1.1.4 Zero overhead looping**
- 5.1.1.5 Hardware Stacks**

5.1.2 Interrupts

- 5.1.2.1 Programmable Interrupts**
- 5.1.2.2 Interrupt Vector Table**
- 5.1.2.3 Handling Interrupts in C**

5.1.3 Instruction cache, PM data access

5.2 Core Timer

5.2.1 Features

5.3 System and Memory Mapped registers

- 5.3.1 Status and Mode registers / USTAT registers**
- 5.3.2 System register bit operations**
- 5.3.3 FLAG Bits/FLAGS register**
- 5.3.4 SYSCTL register**
- 5.3.5 Accessing System and Memory Mapped registers in C**

5.4 Simulator Exercise – Core Timer and Interrupts

6 Linker Operations

6.1 Converting C and Assembly files to Object files

- 6.1.1 Features and Overview**
- 6.1.2 Assembler Expressions and Directives**
- 6.1.3 Object Sections**
- 6.1.4 Def21xxx.h Files**

6.2 The Link Process Linker Description File (LDF)

- 6.2.1 Overview**
- 6.2.2 Linker Description File (LDF)**
 - 6.2.2.1 Example LDF**
- 6.2.3 Linker Optimizations**
- 6.2.4 Customizing and Auto Generation**

6.3 Overlays

- 6.3.1 LDF support**
- 6.3.2 Example**

6.4 Exercise – Fract Arithmetic project



- 7 Advanced Instruction Types and SIMD Operation**
 - 7.1 Parallel Instruction Types and Multifunction Computations**
 - 7.1.1 Data registers usage for multifunction computes**
 - 7.1.2 Reciprocal and Divide**
 - 7.1.3 Reciprocal Square Root and Square Root**
 - 7.2 SIMD Single Instruction Multiple Data**
 - 7.2.1 Terminology and Features**
 - 7.2.2 SISD vs SIMD**
 - 7.2.3 Data Access**
 - 7.2.4 SIMD Programming Models**
 - 7.2.5 Status Flags**
 - 7.2.6 Optional Simulator Exercise: code optimization**

- 8 Hardware Tools**
 - 8.1 EZKITs**
 - 8.1.1 Overview with Part Numbers**
 - 8.1.2 EZKIT Extender Boards**
 - 8.1.3 EZKIT Debug Sessions**
 - 8.1.4 Application examples**
 - 8.2 In Circuit Emulators (ICE)**
 - 8.2.1 Configuration of a Debug Target**
 - 8.3 Emulator Debug Sessions**
 - 8.3.1 CCES Features for HW Debug**

- 9 System Booting**
 - 9.1 Boot process**
 - 9.2 Boot Loader Stream**
 - 9.2.1 Format**
 - 9.2.2 Creating**
 - 9.3 Booting Methods**
 - 9.3.1 Boot Modes**
 - 9.3.2 Hardware Configuration**
 - 9.4 Command Line Device Programmer (CLDP) Utility**
 - 9.4.1 Demonstration – Boot stream create/flash**



10 IOP Introduction

- 10.1 Overview of IOP features including DMA, External Port, DAI/DPI, HW accelerators**

11 Direct Memory Access (DMA)

- 11.1 DMA Architecture**
- 11.2 DMA Features**
- 11.3 DMA Channel Prioritization**
- 11.4 DMA Transfer Types**
 - 11.4.1 External Port DMA**
 - 11.4.2 Peripheral DMA**
 - 11.4.3 Memory to Memory DMA**
- 11.5 Transfer Control Blocks (TCB)**
- 11.6 DMA Control and Status registers**
- 11.7 DMA Chaining and Interrupts**
- 11.8 DMA programming examples**
- 11.9 Hardware Exercise - DMA**

12 External Port Overview

- 12.1 EP Configuration**
- 12.2 Asynchronous Memory Interface**
- 12.3 SDRAM Controller**
 - 12.3.1 DDR Controller for ADSP-21469**
- 12.4 Shared Memory Interface (ADSP-31368 only)**

13 Digital Audio Interface (DAI)

- 13.1 DAI Overview**
- 13.2 DAI Peripherals**
 - 13.2.1 SPORT**
 - 13.2.2 SPDIF**
 - 13.2.3 Precision Clock Generator (PGC)**
 - 13.2.4 Sample Rate Converter (SRC)**
 - 13.2.5 Input Data Port (IDP)**
- 13.3 Signal Routing Unit (SRU)**
 - 13.3.1 Terminology and Naming Convention**
 - 13.3.2 Configuring the SRU**
 - 13.3.3 Configuration examples**
- 13.4 DAI Interrupts**



-
- 14 Digital Peripheral Interface (DPI)**
 - 14.1 DPI Overview**
 - 14.2 DPI Peripherals**
 - 14.2.1 SPI**
 - 14.2.2 Two Wire Interface (TWI)**
 - 14.2.3 UART**
 - 14.2.4 General Purpose Timers**
 - 14.3 DPI Interrupts**
 - 14.4 Hardware Exercise – SRU Configuration**

 - 15 Hardware Accelerators (214xx)**
 - 15.1 Overview**
 - 15.2 FIR, FFT, IIR HW Accelerators**
 - 15.2.1 Features**
 - 15.2.2 Programming**

 - 16 Optimization Topics**
 - 16.1 Overview**
 - 16.2 General Approach to Optimization**
 - 16.3 Algorithmic considerations**
 - 16.3.1 DSP Run Time Library**
 - 16.4 Getting to know the Compiler**
 - 16.4.1 Optimization Switches and pragma's**
 - 16.4.2 Built-in functions**
 - 16.4.3 Example compiler output**
 - 16.4.4 Summary**
 - 16.5 Where to start optimizing?**
 - 16.6 C/Assembly Language Interfacing**
 - 16.6.1 Register Usage**
 - 16.6.2 Parameter Passing**
 - 16.6.3 Stack Usage**
 - 16.6.4 Example**



www.kaztek.com

WS_CCESSH

System Development with CCES and the ADI SHARC Processor

17 uC/OS-III

- 17.1 Overview**
 - 17.1.1 VDK comparison**
- 17.2 Adding uC/OS support to a project**
- 17.3 RTOS Features**
 - 17.3.1 Threads and scheduling**
 - 17.3.2 Signaling and synchronization**
 - 17.3.3 Error handling**
 - 17.3.4 Debug assistance**
- 17.4 Demonstrate main API functions via example**