www.kaztek.com

WS_CCESSH5

System Development with CCES and the 5th Generation ADI SHARC Processor

Course Name:	System Development with CrossCore Embedded Studio (CCES) and the ADI ADSP-SC5xx/215xx SHARC Processor Family
Course Code:	WS_CCESSH5
Course Description:	This is a practical and interactive course that is designed to systematically teach how to use the 5th generation multicore 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 utilized. 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. 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. An understanding of the SHARC architecture will enable getting the best performance out of the processor. Architecture topics will cover the enhanced SHARC+ core, memory configuration (internal and external), internal bus architecture (System Fabric), common peripherals, and DMA. Additional architectural topics will include fault management, security features, system event handling, and the boot process. 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. 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 (i.e. System Services and Device Drivers).
Goals and Objectives:	The main course objective is to understand the SHARC architecture (with a focus on the ADSP-SC5xx/215xx 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 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 ADSP-SC58x Architecture Highlights
1.3.1 Sharc+ Core
1.3.2 ARM Core
1.3.3 Peripherals
1.3.4 System Fabric
1.3.5 Fault Management
1.3.6 System Security
1.3.7 Clock Domains

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
2.6.1 Compiler Optimization topics
2.6.2 Performance assessment topics

3 SHARC+ Core Review
3.1 Differences from previous core
3.2 Overview
3.2.1 Registers
3.2.2 Arithmetic Units
3.2.3 Fetching Data
3.2.4 Sequencer
3.2.4.1 Program Flow Control
3.2.4.2 Core Interrupt Controller
3.2.5 Core Timer
3.3 Assembly Code
3.3.1 Syntax
3.3.2 Source code structure
3.4 C Programming Considerations
3.4.1 Accessing Registers
3.4.2 Interrupt Handling
3.5 Exercise – Core Timer and Interrupts

4 Memory Architecture
4.1 SHARC L1 Memory
4.1.1 Local Memory Map
4.1.2 Memory Architecture
4.2 System Memory
4.2.1 System Memory Map
4.2.2 L2 Memory
4.2.3 External Memory Spaces
4.2.4 Accessing Memory Spaces
4.3 External Memory Interfaces
4.3.1 Features and Overview
4.3.2 Static Memory Controller
4.3.3 Dynamic Memory Controller
4.4 Memory Protection Overview
4.4.1 Parity/ECC for L1/L2 memory
4.4.2 CRC engines
4.4.3 SMPU (System memory Protection Unit)
4.4.4 Example LDF Memory Section

5 SIMD and Advanced Instructions
5.1 Issuing Parallel Instructions
5.2 Advanced Instructions
5.3 SIMD operation

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
6.1.3.1 Byte vs Word Addressing
6.2 The Link Process Linker Description File (LDF)
6.2.1 LDF structure
6.2.2 Linker Optimizations
6.3 Customizing and Auto Generation

7 Hardware Tools
7.1 EZKITs
7.1.1 Overview
7.1.2 Board Support Package
7.1.2.1 Application examples
7.2 In Circuit Emulators (ICE)
7.2.1 Configuration of a Debug Target
7.3 Emulator Debug Sessions
7.3.1 CCES Features for HW Debug
7.3.2 Multicore Debug Considerations
7.4 CPU Hardware Debug Features
7.4.1 System Debug and Trace Unit
7.4.2 System Watchpoint Unit

8 System Booting
8.1 Boot process
8.2 Boot Loader Stream
8.2.1 Format
8.2.2 Creating
8.2.3 Multi Core Considerations
8.3 Booting Methods
8.3.1 Boot Modes
8.3.2 Hardware Configuration

8.4 Command Line Device Programmer (CLDP) Utility
8.4.1 Demonstration – Boot stream create/flash

9 System Services and Device Drivers
9.1 Overview
9.2 System Services API
9.3 Device drivers
9.3.1 API
9.3.2 Buffers
9.3.3 Callback considerations
9.4 Creating CCES projects that use SS/DD
9.4.1 Finding the files
9.5 Walkthrough UART echo example
9.5.1 Demonstrate API and SSL considerations

10 System Event Handling
10.1 GIC (Generic Interrupt Controller) and SEC (System Event Controller)
10.1.1 Fault Management
10.1.2 Interrupt processing flow
10.1.3 Programming model
10.2 Trigger Routing Unit (TRU)
10.2.1 Description
10.2.2 Comparison to traditional Event Handling
10.2.3 Programming model

11 PORTs and GPIO
11.1 PORTs Overview
11.1.1 Multiplexing GPIO with peripheral functions
11.1.2 Pin Mux Control
11.2 General Purpose I/O (GPIO)
11.2.1 Features
11.2.2 Configuration
11.2.3 Use in Event management

12 Basic Peripherals
12.1 SPI
12.2 General Purpose Timers (TIMER)
12.2.1 Operating modes
12.3 Watchdog timer (WDT)
12.4 Two Wire Interface (TWI)
12.5 UART

13 Direct Memory Access (DMA)
13.1 Overview
13.1.1 Types of DMA
13.2 Configuring DMAs
13.2.1 Register based DMA
13.2.2 Descriptor based DMA
13.2.3 System Services MemDMA Support
13.3 DMA control
13.3.1 Interrupts
13.3.2 Trigger Support
13.3.3 Priority (via System Crossbars)
13.3.4 CRC (Cyclic Redundancy Check)
13.3.5 Bandwidth Limiting/Monitoring
13.4 Exercise - MemDMA

14 Digital Audio Interface (DAI)
14.1 DAI Overview
14.2 DAI Peripherals
14.2.1 SPORT
14.2.2 Precision Clock Generator (PGC)
14.2.3 Asynchronous Sample Rate Converter (ASRC)
14.3 Signal Routing Unit (SRU)
14.3.1 Terminology and Naming Convention
14.3.2 Configuring the SRU
14.4 DAI Interrupts

15 Hardware Accelerators
15.1 FIR/IIR/FFT Accelerators
15.2 Software Support

16 Security Features
16.1 Overview
16.2 System Protection Unit (SPU)
16.3 System Memory Protection Unit (SMPU)
16.4 Code example

17 C Run Time Environment
17.1 Compiler Overview
17.2 Function Call and Return
17.3 C/Assembly Language Interface
17.4 Code example

18 Platform Design Topics
18.1 Intercore Communications
18.2 Clock Management
18.2.1 System Services Support

19 Micrium uC/OS-III
19.1 Overview
19.2 Adding uC/OS Support to a Project
19.3 RTOS Features
19.3.1 Threads and Scheduling
19.3.2 Signaling and Synchronization
19.3.3 Error Handling
19.3.4 Debug Assistance
19.4 Demonstrate API Functions via Example