

<b>Course Name:</b>	System Development and Programming with the ADSP-218x
<b>Course Number:</b>	ADST-180
<b>Course Description:</b>	<p>This is a practical course with ‘hands on’ training using the latest VisualDSP++ code development tools. First the core elements of the processor, 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, along with external port interfacing. Advanced instructions are presented with a follow on lab on code optimization. The peripherals, which include the Timer, SPORTS, and external port, are discussed in detail along with IDMA and BDMA operation between these peripherals and internal memory. System related information such as booting and the powerdown feature are also discussed. Hardware development tools, such as evaluation boards and the In Circuit Emulator (ADDS-218X-ICE) are then 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, and ‘C’ programming support.</p>
<b>Goals/Objectives:</b>	<p>The main course objective is to understand the architecture of the ADSP-218x DSP’s 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>
<b>Pre-requisites:</b>	Previous embedded microprocessor background would be an asset (hardware and/or software)
<b>Target Audience:</b>	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
<b>Target Duration:</b>	3 days

- 1 Introduction**
  - 1.1 Introduction/ Course Overview**
    - 1.1.1 DSP at Analog**
    - 1.1.2 Goal of Workshop**
    - 1.1.3 Course Overview**
    - 1.1.4 Course Handouts**
    - 1.1.5 ADSP-218x DSP Core Features**
    - 1.1.6 Modified Harvard Architecture**
    - 1.1.7 Binary Number Formats**
- 2 Introduction to VisualDSP++**
  - 2.1 What is VisualDSP++**
  - 2.2 Software Development Flow**
  - 2.3 Integrated Development and Debugging Environment (IDDE)**
  - 2.4 IDDE Features**
  - 2.5 Project Development**
    - 2.5.1 Project Development Steps**
    - 2.5.2 Project Property Page**
    - 2.5.3 Assembler**
    - 2.5.4 Compiler**
    - 2.5.5 Linker**
    - 2.5.6 Loader**
  - 2.6 Introduction to the Debugger Front End**
    - 2.6.1 Selecting Sessions**
    - 2.6.2 User Interface**

### **3 ADSP 21xx Family Core Architecture – Part I**

#### **3.1 Core Overview**

#### **3.2 Arithmetic Logic Unit (ALU)**

##### **3.2.1 Features**

##### **3.2.2 Instructions**

##### **3.2.3 Flags**

##### **3.2.4 Simulator Exercise**

#### **3.3 Multiplier/ Accumulator (MAC)**

##### **3.3.1 Features**

##### **3.3.2 Instructions**

##### **3.3.3 Flags**

##### **3.3.4 Fractional and Integer Math**

##### **3.3.5 Simulator Exercise**

#### **3.4 Shifter**

##### **3.4.1 Features**

##### **3.4.2 Instructions**

##### **3.4.3 Flags**

##### **3.4.4 Simulator Exercise**

### **4 ADSP 21xx Family Core Architecture – Part II**

#### **4.1 Data Address Generators (DAG)**

##### **4.1.1 Features**

##### **4.1.2 Assembly Instructions**

##### **4.1.3 Data Moves**

##### **4.1.4 Circular Buffering**

##### **4.1.5 Bit Reversing**

##### **4.1.6 Simulator Exercise**

#### **4.2 Program Sequencer**

##### **4.2.1 Features**

##### **4.2.2 Assembly Instructions**

###### **4.2.2.1 Conditional Sequencing**

###### **4.2.2.2 Branching**

##### **4.2.3 Zero Overhead Looping**

##### **4.2.4 Interrupt Handling**

#### **4.3 Control and Status Registers**

- 5 ADSP-218x Memory Model**
  - 5.1.1 Memory Configuration**
  - 5.1.2 Memory Maps**
    - 5.1.2.1 Data**
    - 5.1.2.2 Program**
    - 5.1.2.3 Overlay**
  - 5.1.3 External Memory Interfaces**
- 6 Program Development**
  - 6.1 IDDE and Projects**
  - 6.2 Assembler**
    - 6.2.1 Features and Overview**
    - 6.2.2 Assembler Expressions**
    - 6.2.3 Assembler Directives**
    - 6.2.4 Definition Files**
  - 6.3 Linker / Linker Description File (LDF)**
    - 6.3.1 Features and Overview**
    - 6.3.2 Example LDF**
    - 6.3.3 LDF Commands**
  - 6.4 Using the Simulator for Code Debug**
    - 6.4.1 Features and Overview**
  - 6.5 Assembly Code Programming Exercise**
- 7 Advanced Instructions**
  - 7.1 Multifunction Instructions**
  - 7.2 Divide Support**
  - 7.3 Multi-precision Math**
  - 7.4 Floating Point and Block Floating Point Support**
  - 7.5 Instruction Set Summary**
- 8 ADSP-218x Peripherals**
  - 8.1 Features/Overview**
  - 8.2 Timer**
  - 8.3 Serial PORT (SPORT)**
  - 8.4 Direct Memory Access (DMA)**
    - 8.4.1 Internal DMA port (IDMA)**
    - 8.4.2 Byte DMA port (BDMA)**
  - 8.5 Powerdown**

**9 Advanced Linker Features****9.1 Overlays****10 C Compiler****10.1 Features****10.2 Configuration****10.3 LDF Related issues****10.4 Register Usage****10.5 Run Time Environment****10.5.1 Stack and HEAP****10.5.2 Interrupt Handling****10.6 Assembly Code Interface****11 Software Exercises**

This section includes a number of exercises of varying complexity, which are designed to guide the student through the various aspects of code development using VisualDSP++, as well as, to become more familiar with the programming model for the ADSP-218x.

**12 Hardware Tools****12.1 ADSP-218x EZ-Kits****12.2 Emulators**