Introduction

This chapter introduces the TMS320C6000 (C6000) DSP architecture and peripherals as well as the C6416 and C6713 DSP Starter Kit’s (DSK’s).

The chapter ends with a simple lab to setup the (DSK) and Code Composer Studio (CCS). We like to start small and easy and then build to much more complicated topics and exercises later.

Learning Objectives

Introduction to the:

- C6000 CPU Architecture
- C6000 Peripherals
- C6000 DSK’s
Chapter Topics

C6000 Introduction ................................................................................................................................... 1-1

    What Problem are we Trying to Solve .................................................................................................... 1-3
    Goals of ‘C6000 Architecture.................................................................................................................. 1-3

    C6000 Architecture ................................................................................................................................. 1-5
    CPU Architecture Overview .................................................................................................................... 1-5
    The C6000 (zooming out from the CPU) ............................................................................................... 1-8

    Connecting to a C6000 Device ................................................................................................................ 1-9

    C6000 DSK’s ........................................................................................................................................... 1-14
    Overview ................................................................................................................................................ 1-14
    DSK Diagnostic Utility ............................................................................................................................ 1-16
    Memory Map ........................................................................................................................................... 1-17
    In the DSK Package ................................................................................................................................ 1-18

Lab 1 - Prepare Lab Workstation ................................................................................................................ 1-19

    C64x or C67x Exercises? .......................................................................................................................... 1-20
    Computer Login ........................................................................................................................................ 1-21
    Connecting the DSK to your PC ............................................................................................................... 1-21
    Testing Your Connection ......................................................................................................................... 1-22
    CCS Setup ............................................................................................................................................ 1-22
    Set up CCS – Customize Options ........................................................................................................ 1-26

Appendix (For Reference Only) .................................................................................................................. 1-31

    Power On Self-Test stages ....................................................................................................................... 1-31
    DSK Help .............................................................................................................................................. 1-32
What Problem are we Trying to Solve

Goals of ‘C6000 Architecture

Conundrum: How to define Digital Signal Processing (DSP) in one slide.

In its simplest form, most DSP systems receive data from an ADC (analog to digital converter). The data is processed by the Digital Signal Processor (also called DSP) and the results are then transformed back to analog to be output. Digitizing the analog signal (by evaluating it to a number on a periodic basis) and the subsequent numerical (a.k.a. digital) analysis provides a more reliable and efficient means of manipulating the signal vs. performing the manipulation in the analog domain. With the growing interest in multimedia, the demand for DSPs to process the various media signals is growing exponentially.

While interest in DSP is constantly growing today, the DSProcessor grew out of TI over 20 years ago in its educational products group, namely the Speak and Spell. These products demanded speech synthesis and other traditional DSProcessing (like filters) but with quick time-to-market constraints.

The heart of DSP algorithms hasn’t changed from the early days of TI DSP; they still rely on the fundamental difference equation (shown above). Often this equation is referred to as a MAC (multiply-accumulate) or SOP (sum-of-products). TI has concentrated for years on providing solutions to MAC based algorithms. The wide variety of TI DSPs is a testament to this focus, even with the widely varying system tradeoffs discussed earlier.
For the ‘C6000 to achieve its goal, TI wanted to provide record setting performance while coding with the universal ANSI C language.

Fast MAC using only C

Multiply-Accumulate (MAC) in Natural C Code

```c
for (i = 1; i < count; i++)
    Y += coeff[i] * x[i];
```

- **Fastest Execution of MACs**
  - The ‘C6x roadmap ... from 200 to 4000 MMACs

- **Ease of C Programming**
  - Even using natural C, the ‘C6000 Architecture can perform 2 to 4 MACs per cycle
  - Compiler generates 80-100% efficient code

How does the ‘C6000 achieve such performance from C?

TI ‘C6000 devices deliver 200 to 4000 MMACs of performance, where MMAC is mega-MAC or millions of MACs. It's stellar performance, in any case. When this can be achieved using C code, it's even better. While providing efficiency ratings for a compiler is difficult, TI has benchmarked a large number of common DSP kernels to provide an example of the compiler’s efficiency - please visit the TI website for more information and benchmarking examples.
C6000 Architecture

CPU Architecture Overview

How does the ‘C6000 deliver its performance, the CPU is built to dispatch 8 instructions per cycle – and the cycle rates run as fast as about 1 ns.

- ‘C6000 Compiler excels at Natural C
- While dual-MAC speeds math intensive algorithms, flexibility of 8 independent functional units allows the compiler to quickly perform other types of processing
- All ‘C6000 instructions are conditional allowing efficient hardware pipelining
- ‘C6000 CPU can dispatch up to eight parallel instructions each cycle
The following example demonstrates the capability of the ‘C6000 architecture. Specifically, the ‘C67x floating-point DSP can execute these eight instructions in parallel, allowing two single-precision floating point MACs to be performed in just one processor cycle. Oh, and all that from ordinary C code.

Fastest MAC using Natural C

float mac(float *m, float *n, int count) {
    int i, float sum = 0;
    for (i=0; i < count; i++) {
        sum += m[i] * n[i];
    }
    return sum;
}

How does it look from a benchmark perspective?

Sample Compiler Benchmarks

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Used In</th>
<th>Asm Cycles</th>
<th>Assembly Time (μs)</th>
<th>C Cycles (Rel 4.0)</th>
<th>C Time (μs)</th>
<th>% Efficiency vs Hand Coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Mean Square Error</td>
<td>For motion compensation of image data</td>
<td>348</td>
<td>1.16</td>
<td>402</td>
<td>1.34</td>
<td>87%</td>
</tr>
<tr>
<td>Codebook Search</td>
<td>CELP based voice coders</td>
<td>977</td>
<td>3.26</td>
<td>961</td>
<td>3.20</td>
<td>100%</td>
</tr>
<tr>
<td>Vector Max 40 element input vector</td>
<td>Search Algorithms</td>
<td>61</td>
<td>0.20</td>
<td>59</td>
<td>0.20</td>
<td>100%</td>
</tr>
<tr>
<td>All-zero FIR Filter 40 samples, 10 coefficients</td>
<td>VSELP based voice coders</td>
<td>238</td>
<td>0.79</td>
<td>280</td>
<td>0.93</td>
<td>85%</td>
</tr>
<tr>
<td>Minimum Error Search Table Size = 2304</td>
<td>Search Algorithms</td>
<td>1185</td>
<td>3.95</td>
<td>1318</td>
<td>4.39</td>
<td>90%</td>
</tr>
<tr>
<td>IIR Filter 16 coefficients</td>
<td>Filter</td>
<td>43</td>
<td>0.14</td>
<td>38</td>
<td>0.13</td>
<td>100%</td>
</tr>
<tr>
<td>IIR – cascaded biquads 10 Cascaded biquads (Direct Form II)</td>
<td>Filter</td>
<td>70</td>
<td>0.23</td>
<td>75</td>
<td>0.25</td>
<td>93%</td>
</tr>
<tr>
<td>MAC Two 40 sample vectors</td>
<td>VSELP based voice coders</td>
<td>61</td>
<td>0.20</td>
<td>58</td>
<td>0.19</td>
<td>100%</td>
</tr>
<tr>
<td>Vector Sum Two 44</td>
<td></td>
<td>51</td>
<td>0.17</td>
<td>47</td>
<td>0.16</td>
<td>100%</td>
</tr>
</tbody>
</table>

- Great out-of-box experience
- Completely natural C code (non ‘C6000 specific)
- Code available at dspvillage.com

Ti C62x™ Compiler Performance Release 4.0: Execution Time in μs @ 300 MHz
Versus hand-coded assembly based on cycle count
The C64x devices provide tremendous Multiply-Accumulate performance. Not only are they running at frequencies 2-3 times faster than other C6000 processors, but each of the multiply units can now perform two 16x16 multiplies plus a 32-bit add in one cycle. This is accomplished by the DOTP2 assembly instruction.

**C64x gets four MAC’s using DOTP2**

```
short mac(short *m, short *n, int count) {
  int i, short sum = 0;
  for (i=0; i < count; i++) {
    sum += m[i] * n[i];
  }
  return sum;
}
```

**PIPED LOOP KERNEL**

```
LOOP: ADD .L2 B8,B6,B6
  ADD .L1 A6,A7,A7
  DOTP2 .M2X B4,A4,B8
  DOTP2 .M1X B5,A5,A6
  [ B0] B .S1 LOOP
  [ B0] SUB .S2 B0,-1,B0
  LDDW .D2T2 *B7++,B5:B4
  LDDW .D1T1 *A3++,A5:A4
```

How many multiplies can the ‘C6x perform?

**MMAC’s**

✦ How many 16-bit MMACs (millions of MACs per second) can the ‘C6201 perform?

400 MMACs (two .M units x 200 MHz)

✦ How about 16x16 MMAC’s on the ‘C64x devices?

2 .M units
x 2 16-bit MACs (per .M unit / per cycle)
x 1 GHz

4000 MMACs

✦ How many 8-bit MMACs on the ‘C64x?

8000 MMACs (on 8-bit data)
The C6000 (zooming out from the CPU)

Zooming out from the CPU, we find a number of internal busses connected to it. The peripherals shown here will be discussed next.

As an example, here is an internal view of the C6415 device:

From this diagram notice two things:

- **Dual-level memory** (this will be discussed further in Chapter 4):
  - L1 (level 1) program and data caches
  - L2 (level 2) combined program/data memory

- **High-performance, internal buses**
  - Buses as large as 64- and 256-bits allow an enormous amounts of info to be moved
  - Multiple buses allow simultaneous movement of data in a C6000 system
  - Both the EDMA and CPU can orchestrate moving information

**Note:** While we have been looking into the C6415, you can extrapolate these same concepts to other C6000 device types. All device types have multiple, fast, internal buses. Most have a dual-level memory architecture, while a few have a single-level, flat memory.
Connecting to a C6000 Device

C6000 devices contain a variety of peripherals to allow easy communication with off-chip memory, co-processors, and other devices. The diagram below provides a quick overview:

![Example C6000 System Diagram]

Let’s quickly look at each of these connections beginning with VCP/TCP and working counter-clockwise around the diagram.

**Viterbi Coprocessor (VCP)**
- Used for 3G Wireless applications
- Supports >500 voice channels at 8 kbps
- Programmable decoder parameters include constraint length, code rate, and frame length
- Available on the ‘C6416

**Turbo Coprocessor (TCP)**
- Used for 3G Wireless applications
- Supports 35 data channels at 384 kbps
- 3GPP / IS2000 Turbo coder
- Programmable parameters include mode, rate and frame length
- Available on the ‘C6416

Note: Not all C6000 devices have all the various peripherals shown above. Please refer to the C6000 Product Update for a device-by-device listing.
Timer / Counters
- Two (or three) 32-bit timer/counters
- Use as a Counter (counting pulses from input pin)
or as a Timer (counting internal clock pulses)
- Can generate:
  - Interrupts to CPU
  - Events to DMA/EDMA
  - Pulse or toggle-value on output pin
- Each timer/counter as both input and output pin

General Purpose Input/Output (GPIO)
- Observe or control the signal of a single-pin
- Dedicated GPIO pins on ‘C6713 and all ‘C64x devices
- All ‘C6000 devices have shared GPIO with unused peripheral pins

Hardware Interrupts (HWI)
- Allows synchronization with outside world:
  - Four configurable external interrupt pins
  - One Non-Maskable Interrupt (NMI) pin
  - Reset pin
- C6000 CPU has 12 configurable interrupts.
  Some of the properties that can be configured are:
  - Interrupt source (for example: Ext Int pin, McBSP receive, HPI, etc.)
  - Address of Interrupt Service Routine (i.e. interrupt vector)
  - Whether to use the HWI dispatcher
  - Interrupt nesting
- The DSP/BIOS HWI Dispatcher makes interrupts easy to use

Parallel Peripheral Interface
- C6000 provides three different parallel peripheral interfaces; the one you have depends upon which C6000 device you are using (see C6000 Product Update for which device has which interface)
  - HPI: Allows another processor access to C6000’s memory using a dedicated, async 16/32-bit bus; where C6000 is slave-only to host.
  - XBUS: Similar to HPI but provides but adds: 32-bit width, Master or slave modes, sync modes, and glueless I/O interface to FIFOs or memory (memory I/O can transfer up to full processor rates, i.e. single-cycle transfer rate).
  - PCI: Standard master/slave 32-bit PCI interface
    (latest devices – e.g. DM642 – now allow 66MHz PCI communication)
Connecting to a C6000 Device

Direct Memory Access (DMA / EDMA)
- EDMA stands for the Enhanced DMA (each C6000 has either a DMA or EDMA)
- Transfers any set of memory locations to any another (internal or external)
- Allows synchronized transfers; that is, they can be triggered by any event (i.e. interrupt)
- Operates independent of CPU
- 4 / 16 / 64 channels (set’s of transfer parameters) (various by C6000 device type)
- “If you are not using the DMA/EDMA, you’re probably not getting the full performance from your ‘C6000 device.”

DMA: Offers four fully configurable channels (additional channel for the HPI), Event synchronization, Split mode for use with McBSP, and Address/count reload
EDMA: Enhanced DMA (EDMA) offers 16 fully configurable channels (64 channels on ‘C64x devices), Event synchronization, Channel linking, and Channel auto-initialization.

Boot Loader
- After reset but before the CPU begins running code, the “Boot Loader” can be configured to either:
  - Automatically copy code and data into on-chip memory
  - Allow a host system (via HPI, XBUS, or PCI) to read/write code and data into the C6000’s internal and external memory
  - Do nothing and let the CPU immediately begin execution from address zero
- Boot mode pins allow configuration
- Please refer to the C6000 Peripherals Guide and each device’s data sheet for the modes allowed for each specific device.

External Memory Interface (EMIF)
EMIF is the interface between the CPU (or DMA/EDMA) and the external memory and provides all of the required pins and timing to access various types of memory.
- Glueless access to async or sync memory
- Works with PC100 SDRAM — cheap, fast, and easy! (more recent designs now allow use of PC133 SDRAM)
- Byte-wide data access
- C64x devices have two EMIFs (16-bit and 64-bit width)
- 16, 32, or 64-bit bus widths (please check the specifics for your device)
Connecting to a C6000 Device

Ethernet
- 10/100 Ethernet interface
- To conserve cost, size and power – Ethernet pins are muxed with PCI (you can use one or the other)
- Optimized TCP/IP stack available from TI (under license)

Multi-Channel Buffered Serial Port (McBSP)
- Commonly used to connect to serial codecs (codec: combined A/D and D/A devices), but can be used for any type of synchronous serial communication
- Two (or three) synchronous serial-ports
- Full Duplex: Independent transmit and receive sections (each can be individually sync’d)
- High speed, up to 100 Mb/sec performance
- Supports:
  - SPI mode
  - AC97 codec interface standard
  - Supports multi-channel operation (T1, E1, MVIP, …)
  - And many other modes
- Software UART available for most C6000 devices
  (Check the DSP/BIOS Drivers Developer Kit (DDK))

McASP
- All McBSP features plus more …
- Targeted for multi-channel audio applications such as surround sound systems
  - Up to 8 stereo lines (16 channels) - supported by 16 serial data pins configurable as transmit or receive
  - Throughput: 192 kHz (all pins carrying stereo data simultaneously)
- Transmit formats:
  - Multi-pin IIS for audio interface
  - Multi-pin DIT for digital interfaces
- Receive format:
  - Multi-pin IIS for audio interface
- Available on C6713 and DM642 devices.

Utopia
- For connection to ATM (async transfer mode)
- Utopia 2 slave interface
- 50 MHz wide area network connectivity
- Byte wide interface
- Available on ‘C64x devices
PLL
- On-chip PLL provides clock multiplication. The ‘C6000 family can run at one or more times the provided input clock. This reduces cost and electrical interference (EMI).
- Clock modes are pin configurable.
- On most devices, along with the Clock Mode (configuration) pins, there are three other clock pins:
  - CLKin: clock input pin
  - CLKOUT: clock output from the PLL (multiplied rate)
  - CLKOUT2: a reduced rate clockout. Usually ½ or less of CLKOUT
Please check the datasheet for the pins, pin names, and CKKOUT2 rates available for your device.
- Here are the PLL rates for a sample of C6000 device types:

<table>
<thead>
<tr>
<th>Device</th>
<th>Clock Mode Pins</th>
<th>PLL Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6201</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6204</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6205</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6701</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6202</td>
<td>CLKMODE0</td>
<td>x1, x4, x6, x7, x8, x9, x10, x11</td>
</tr>
<tr>
<td>C6203</td>
<td>CLKMODE1</td>
<td>x1, x4, x6, x7, x8, x9, x10, x11</td>
</tr>
<tr>
<td>C6211</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6711</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6712</td>
<td>CLKMODE</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6414</td>
<td>CLKMODE0</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6415</td>
<td>CLKMODE1</td>
<td>x1, x4</td>
</tr>
<tr>
<td>C6416</td>
<td>CLKMODE1</td>
<td>x1, x4</td>
</tr>
</tbody>
</table>

Power Down
- While not shown in the previous diagram, the ‘C6000 supports power down modes to significantly reduce overall system power.

For more detailed information on these peripherals, refer to the ‘C6000 Peripherals Guide.'
C6000 DSK’s

Overview

Here’s a detailed look at the DSK board and its primary features:

C6416T DSK

C6416 / C6713 DSK Features

- **TMS320C6416 DSP:** 1GHz, fixed-point, 1M Byte internal RAM
  - or
  - **TMS320C6713 DSP:** 225MHz, floating-point, 256K Byte internal RAM
- **External SDRAM:**
  - C6416 – 64-bit interface
  - C6713 – 32-bit interface
- **External Flash:** 512K Bytes, 8-bit interface
- **AIC23 Codec:** Stereo, 8KHz –96KHz sample rate, 16 to 24-bit samples; mic, line-in, line-out and speaker jacks
- **CPLD:** Programmable "glue" logic
- **4 User LEDs:** Writable through CPLD
- **4 User DIP Switches:** Readable through CPLD
- **3 Configuration Switches:** Selects power-on configuration and boot modes
- **Daughtercard Expansion I/F:** Allows user to enhance functionality with add-on daughtercards
- **HPI Expansion Interface:** Allows high speed communication with another DSP
- **Embedded JTAG Emulator:** Provides high speed JTAG debug through widely accepted USB host interface
Daughter-Card I/F

The daughter card sockets included on the DSK are similar to those found on other the C5000/C6000 DSKs and EVMs available from Texas Instruments. Thus, any work (by you or any 3rd Party) applied to daughter card development can be reused with the DSK. If you’re interested in designing a daughter card for the DSK/EVM, check the TI website for an application note which describes it in detail.

Block Diagram

Here’s a block diagram view of the C6416 DSK.

The C6713 would be almost exactly the same. (We pulled this diagram from the C6416 help file. Look in the C6713 help file <CCS Help menu> to find a similar diagram for that platform.)
DSK Diagnostic Utility

DSK’s Diagnostic Utility

- Test/Diagnose DSK hardware
- Verify USB emulation link
- Use Advanced tests to facilitate debugging
- Reset DSK hardware
Memory Map

The following memory-map describes the memory resources designed into the ‘C6416 DSK.

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000_0000</td>
<td>Internal RAM: 1MB</td>
</tr>
<tr>
<td>0010_0000</td>
<td>Internal Peripherals or reserved</td>
</tr>
<tr>
<td>6000_0000</td>
<td>EMIFB CE0: 64MB</td>
</tr>
<tr>
<td>6400_0000</td>
<td>EMIFB CE1: 64MB</td>
</tr>
<tr>
<td>6800_0000</td>
<td>EMIFB CE2: 64MB</td>
</tr>
<tr>
<td>6C00_0000</td>
<td>EMIFB CE3: 64MB</td>
</tr>
<tr>
<td>8000_0000</td>
<td>EMIFA CE0: 256MB</td>
</tr>
<tr>
<td>9000_0000</td>
<td>EMIFA CE1: 256MB</td>
</tr>
<tr>
<td>A000_0000</td>
<td>EMIFA CE2: 256MB</td>
</tr>
<tr>
<td>B000_0000</td>
<td>EMIFA CE3: 256MB</td>
</tr>
</tbody>
</table>

The left map describes the resources available on the ‘C6416 DSP, the right map details how the external memory resources were used on the DSK.
# In the DSK Package

## DSK Contents (i.e. what you get…)

<table>
<thead>
<tr>
<th>Documentation</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>◆ DSK Technical Reference</td>
<td>◆ eXpressDSP for Dummies</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>◆ Code Composer Studio</td>
<td>◆ SD Diagnostic Utility</td>
<td></td>
</tr>
<tr>
<td></td>
<td>◆ Example Programs</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td>MISC Hardware</td>
<td></td>
</tr>
<tr>
<td>◆ 1GHz C6416T DSP</td>
<td>◆ LEDs and DIPs</td>
<td></td>
</tr>
<tr>
<td>or 225 MHz C6713 DSP</td>
<td>◆ Daughter card expansion</td>
<td></td>
</tr>
<tr>
<td>◆ TI 24-bit A/D Converter (AIC23)</td>
<td>◆ 1 or 2 additional expansions</td>
<td></td>
</tr>
<tr>
<td>◆ External Memory</td>
<td>◆ Power Supply &amp; USB Cable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>◆ 8 or 16MB SDRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>◆ Flash ROM - C6416 (512KB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- C6713 (256KB)</td>
<td></td>
</tr>
</tbody>
</table>
Lab 1 - Prepare Lab Workstation

The computers used in TI’s classrooms and dedicated workshops may be configured for one of ten different courses. The last class taught may have been DSP/BIOS, TMS320 Algorithm Standard, or a C5000 workshop. To provide a consistent starting point for all users, we need to have you complete a few steps to reset the CCS environment to a known starting point.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Hook up the DSK</td>
<td>1. Run Diagnostic Utility</td>
</tr>
<tr>
<td>2. Supply power and observe POST</td>
<td>2. Run CCS Setup</td>
</tr>
<tr>
<td></td>
<td>3. Start CCS</td>
</tr>
<tr>
<td></td>
<td>4. Configure CCS Options</td>
</tr>
<tr>
<td></td>
<td>5. Close CCS</td>
</tr>
</tbody>
</table>

In Lab 1, we're going to prepare your lab workstations. This involves:
- Hooking up your DSK
- Running the DSK Diagnostic Utility to verify the USB connection and DSK are working
- Running CCS Setup to select the proper emulation driver (DSK vs. Simulator)
- Starting CCS and setting a few environment properties

Time: 20 minutes
C64x or C67x Exercises?
We support two processor types in these workshop lab exercises. Please see the specific callouts for each processor as you work. Overall, there are very little differences between the procedures.

Lab Exercises – C67x vs. C64x
- Which DSK are you using?
- We provide instructions and solutions for both C67x and C64x.
- We have tried to call out the few differences in lab steps as explicitly as possible:
Computer Login

1. If the computer is not already logged-on, check to see if the log-on information is posted on the workstation. If not, please ask your instructor.

Connecting the DSK to your PC

The software should have already been installed on your lab workstation. All you should have to do physically connect the DSK

2. Connect the supplied USB cable to your PC or laptop.

   If you connect the USB cable to a USB Hub, be sure the hub is connected to the PC or laptop and power is applied to the hub.

   **Note:** After plugging in the USB cable, if a message appears indicating that the USB driver needs to be installed, put the CCS CD from the DSK into the CD-ROM drive and allow the driver to be installed. In most classroom installations, this has already been completed for you.

3. Plug-in the appropriate audio connections.
   - Connect your headphone or speaker to the audio output.
   - An audio patch cable is provided to connect your computer’s soundcard (or your music source) to the line-in connector on the DSK board.

   **Note:** Make sure you insert the audio source and headphone plugs all the way into their respective sockets. Failing to do this may allow audio to short from the input to the output. While this may not hurt the board, it will prevent you from effectively evaluating your DSP code.

4. Plug the AC power cord into the power supply and AC source.

   **Note:** Power cable must be plugged into AC source prior to plugging the 5 Volt DC output connector into the DSK.

5. Plug the power cable into the board. (note: when the POST runs in the next step and you have the earpiece in your ear, it will HURT!)

6. When power is applied to the board, the Power On Self Test (POST) will run. LEDs 0-3 will flash. When the POST is complete all LEDs blink on and off then stay on.

   **Hint:** At this point, if you were installing the DSK for the first time on your own machine you would now finish the USB driver installation. We have already done this for you on our classroom PC’s.
Lab 1 - Prepare Lab Workstation

Testing Your Connection

7. Test your USB connection to the DSK by launching the DSK Diagnostic Utility from the icon on the PC desktop.

From the diagnostic utility, press the start button to run the diagnostics. In approximately 20 seconds all the on-screen test indicators should turn green.

Note: If using the C6713 DSK, the title on this icon will differ accordingly.

If the utility fails while testing the DSK:
- Check to make sure the DSK is receiving power.
- Also, verify the USB cable is plugged into both the DSK and the PC.
- After ruling out cabling, a failure is most often caused by an incomplete USB driver installation. Deleting and reinstalling the driver often solves this problem. (Again, you should rarely see this problem.)

CCS Setup

While Code Composer Studio (CCS) has been installed, you will need to assure it is setup properly. CCS can be used with various TI processors – such as the C6000 and C5000 families – and each of these has various target-boards (simulators, EVMs, DSKs, and XDS emulators). Code Composer Studio must be properly configured using the CCS_Setup application.

In this workshop, you should initially configure CCS to use either the C6713 DSK or the C6416 V1.1 DSK. Between you and your lab partner, choose one of the DSK’s and the appropriate driver. In any case, the learning objectives will be the same whichever target you choose.

8. Start the CCS Setup utility using its desktop icon:

Be aware there are two CCS icons, one for setup, and the other to start the CCS application. You want the Setup CCS C6000 icon.

Sidebar: CCS Setup

The version of CCS that ships with the DSK will not place the Setup CCS 2 icon on the desktop, nor will the shortcut appear under the Windows start menu:

Start → Programs → Texas Instruments → Code Composer Studio 2 (C6000) → Setup Code Composer Studio

The setup program <cc_setup.exe> is installed to the hard drive for both the full and DSK versions of CCS, although the desktop icon and Start menu shortcut are only added when installing the full version of CCS.

For your convenience, during installation of the workshop labs and solutions an icon for CCS Setup was placed on the desktop. If, for some unexpected reason, this icon has been deleted, you can find and run the program from:

c:\ti\cc\bin\cc_setup.exe

(where “ti” is the directory you installed CCS)
9. When you open CC_Setup you should see a screen similar to this:

![Import Configuration dialog box](image)

**Note:** If you don’t see the *Import Configuration* dialog box, you should open it from the menu using **File → Import…**

Once the *Import Configuration* dialog box is open, you can change the CC_Setup default to force this dialog to open every time you start CC_Setup. Just check the box in the bottom of the import dialog.
10. Clear the previous configuration.

Before you select a new configuration you should delete the previous configuration. Click the Clear System Configuration button. CC_Setup will ask if you really want to do this, choose “Yes” to clear the configuration.

11. Select a new configuration from the list and click the “Import” button.

If you are using the C6416 DSK in this workshop, please choose the C6416 V1.1 DSK:
If you are using the C6713 DSK in this workshop, please choose the **C6713 DSK**:

12. Save and Quit the *Import Configuration* dialog box.

13. **Go ahead and** start CCS upon exiting *CCS Setup*. 
Set up CCS – Customize Options

There are a few option settings that need to be verified before we begin. Otherwise, the lab procedure may be difficult to follow.

- Disable open Disassembly Window upon load
- Go to main() after load
- Program load after build
- Clear breakpoints when loading a new program
- Set CCS Titlebar information

14. Use the Customize Dialog box to set specific options.

Select:

Option → Customize…

Uncheck the box for *Open the Diasassembly Window automatically*. Check the *Perform Go Main automatically* box. Check the following check box: *Connect to the target when a control window is open*.

Here are a couple options that can help make debugging easier.

- Unless you want the Disassembly Window popping up every time you load a program (which annoys many folks), deselect this option.

- Many find it convenient to choose the “Perform Go Main automatically”. Whenever a program is loaded the debugger will automatically run thru the compilers initialization code to your main() function.
15. Set Program Load Options

On the “Program/Project Load” tab, make sure the options shown below are checked:

- Load Program After Build
- Clear All Breakpoints When Loading New Programs

By default, these options are not enabled, though a previous user of your computer may have already enabled them.

Conceptually, the CCS Integrated Development Environment (IDE) is made up of two parts:

- **Edit** (and Build) programs (uses editor and code gen tools to create code).
- **Debug** (and Load) programs (communicates with DSP/simulator to download/run code).

The **Load Program After Build** option automatically loads the program (.out file) created when you build a project. If you disabled this automatic feature, you would have to manually load the program via the File→Load Program menu.

**Note:** You might even think of IDE as standing for Integrated Debugger Editor, since those are the two basic modes of the tool.
16. CCS Title Bar Properties

CCS allows you to choose what information you want displayed on its title bar.

Note: To reach this tab of the “Customize” dialog box, you may have to scroll to the right using the arrows in the upper right corner of the dialog.

- Make sure that the options shown above are checked.
Choose Text-Based Linker

CCS includes two different linkers. The Visual Linker is now obsolete – therefore we want to make sure it is not selected. Do NOT use or experiment with the Visual Linker.

17. Open the CCS linker selection dialog.

   Tools → Linker Configuration

18. Select *Use the text linker* and click OK (as shown below).


You're Done
*** can you explain why you’re reading a blank page? ***
Appendix (For Reference Only)

Power On Self-Test stages

The following table details the various states of the POST routine and how you can visually track its progress.

<table>
<thead>
<tr>
<th>Test</th>
<th>LED4</th>
<th>LED 3</th>
<th>LED 2</th>
<th>LED 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DSP's Internal Memory test</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>External SDRAM test</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Check manufacture ID of Flash chip</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>McBSP 0 loopback test</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>McBSP 1 loopback test</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>McBSP 2 loopback test</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Transfer small array with EDMA</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Codec test (output 1KHz tone)</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Timer test (cfg and wait for 100 ints)</td>
</tr>
</tbody>
</table>

B L I N K   A L L  All tests completed successfully

- Stored in FLASH memory and runs every time DSK is powered on
- Source code on DSK CD-ROM
- When test is performed, index number is shown on LED’s. If test fails, the index of that test will blink continuously.
- When complete, all LEDs will blink three times, then turn off
- See C6713 DSK help file for its index of tests.

Note: Don’t worry if it takes a few seconds to perform Test 2 (External SDRAM test). It can take a while to test all the SDRAM memory included on the DSK. (Of course, if it takes more than 15-30 seconds, then there might be a problem.)
DSK Help

This file describes the board design, its schematics, and how the DSK utilities work.