DSP Basic Training

Setup Requirements:

S/W: Code Composer Studio 2.0 (Install on your PC) for C6000
H/W: C6201/C6205 DSP EVM board / DSK 6416 Kit

Note: All paths mentioned in this training are relative to the folder where you have installed the CCS 2.0

Session 1: (1 day)

1. Code Composer Studio Setup

Task:
Read the chapter one of the spru509.pdf.

Objective:
Learn about Code Composer Studio Setup.

Assignment:
None.

2. Code Composer Studio IDE

Task:
Go through the "Developing a simple program" topic of the "Getting Started With the Code Composer Studio Tutorial". To access the "Developing a simple program" topic follow the following steps.

- Open the CCS
- Select Help menu and click on Tutorial submenu
- Click on Code Composer Studio IDE
- Click on Developing a simple program

Objective:

- Learn how to create a simple program.
- Learn basic debugging technique.
- Understand CCS components.

Assignment:

- Write a simple program, which print "**Welcome To DSP World**".

3. DSP/BIOS

Task:
Go through the "Using DSP/BIOS" topic of the "Getting Started With the Code Composer Studio Tutorial". To access the "Developing a simple program" topic follow the following steps.

- Open the CCS
- Select Help menu and click on Tutorial submenu
- Click on Using DSP/BIOS

Objective:

- Learn how to configure DSP/BIOS.
- Understand DSP/BIOS components.

Assignment:

- Write a simple program, which print "Welcome To DSP World", using DSP/BIOS.

Session 2: (1 day)

1. **DSP/BIOS Overview**

   Task:
   Read the chapter one of the spru423.pdf.

   Objective:
   Get familiar with the DSP/BIOS.

   Assignment:
   None.

2. **Task Object**

   Task:
   Go through the topic 4.4 and 4.5 of the spru423.pdf.

   Objective:
   Learn about the task and task scheduling.

   Assignment:
   Execute and understand the slice project.
   Path for the slice project is
   "\Ti \examples\sim62xx\bios\slice", for C2xx processor training
   "\Ti \examples\sim64xx\bios\slice", for C4xx processor training

3. **Semaphore Object**

   ### Task:
   Go through the topic 4.6 of the spru423.pdf.

   Objective:
   Learn about the semaphore.

   ### Assignment:
   Execute and understand the semtest project.
   Path for the semtest project is
   "\Ti\tutorial\sim62xx\semtest", for C2xx processor training
   "\Ti\tutorial\sim62xx\semtest", for C4xx processor training

4. **Mailbox Object**

Task:
Go through the topic 4.7 of the spru423.pdf.

Objective:
Learn about the mailbox.

Assignment:
Execute and understand the mbxtest project.
Path for the mbxtest project is
"\Ti\tutorial\sim62xx\mbxtest", for C2xx processor training
"\Ti\tutorial\sim64xx\mbxtest", for C4xx processor training

Session 3: (2 days)

1. **Technical overview of C6000.**

Task:
Read the chapter one and two of the spru395b.pdf.

Objective:
Learn the architecture of the C6000 DSP platform.

Assignment:
None.

2. **C6000 DSP peripherals**

Task:
Read chapter one of the spru190d.pdf – Internal Memory.
Read chapter four of the spru190d.pdf – DMA Controller.
Read chapter six of the spru190d.pdf – EDMA Controller.
Read chapter thirteen of the spru190d.pdf – Timers.

Objective:
- Learn about Internal program memory, Internal data memory, DMA controller, EDMA controller and Timers.
- Learn how to use control register of peripherals for various operations.

Assignment:
- None.

Note: This session is theoretical so you can do session 4 in parallel, which has assignments for all peripherals mentioned above.

Session 4: (2 days)

1. **Assignment One.**

   Task:

   For 62xx Platform:
   Execute the dma1 project on 62xx simulator.
   Path for the dma1 project is "\ti\examples\evm6201\csl\manual_config\".
   Understand the source code of this project.

   For 64xx Platform:
   Execute the edma project on 64xx simulator.
   Path for the edma project is "\ti\examples\dsk6416\csl\manual_config\".
   Understand the source code of this project.

   Objective:
   Learn about various APIs of CSL library for DMA/EDMA transfer.
   Learn polling base and interrupt base DMA transfer.

   Note: This dma1/edma project is for EVM620x/DSK6416, but you have to run it on 62xx/64xx simulator, so create one project for 62xx/64xx simulator platform and add source file of dma1/edma project respectively.

2. **Assignment two.**

   Task:
   Assignment one is block interrupt based. Modify it for frame interrupt. Transfer the same amount of data but as two frames.

   Objective:
   Learn about block, frame and element in DMA/EDMA transfer.

3. **Assignment three.**

   Task:
   Write a program which print message at every 2 seconds for EVM620x / DSK6416.

   Objective:
   Learn about various APIs of CSL library for Timer operations.

   Note: Take timer1 project as reference. The path of timer1 project is "\ti\examples\evm6201\csl\manual_config\". You need EVM620x / DSK6416 for this assignment. The code is only available for evm6201 examples, so for DSK6416, need to take the .c file and create a new project and add this file.

4. **Assignment four.**

   Task:
   Transfer 2K words from SDRAM bank 0 to SDRAM bank 1 in EVM620x / DSK6416.

   Measure the time for above DMA / EDMA transfer using Timer1.

Objective:
Learn about DMA and Timer peripherals of the DSP.

Note: You need EVM620x / DSK6416 for this assignment.

Session 5:

Presentation:

1. DSP/BIOS.
2. DSP C6000 architecture.

Session 6: (3 days)

DSP Assignment:

Task:
Convert 24-bit bitmap image into the 8-bit bitmap image.

Objective:
- Learn file operation in C.
- Learn about DMA/EDMA operations in DSP.
- Get familiar with Hardware Interrupt, Software Interrupt, Task, and Semaphore in DSP/BIOS.
- Learn about code optimization in DSP platform.

Assignment:

For 620x EVM:
- Write host software to convert 24 bit Image.bmp bitmap file into simple binary file as raw data to DSP software. Store raw data into the SDRAM bank 0 of the EVM board. Use DSP EVM host library. Draw Software flow diagram.
- Write DSP software to convert 24 bit raw data of Image.bmp bitmap file into 8 bit raw data. Store 8 bit raw data into the SDRAM bank 1 of the EVM board. Measure the timings of this conversion and make it as optimum as you can. Use DSP/BIOS, DMA transfer, task, semaphore and interrupts. Draw Software flow diagram.
- Write host software to convert 8 bit raw data from SDRAM bank 1 of EVM board into the 24-bit bitmap image (pad remaining 16 bits with 0). Use DSP EVM host library. Draw Software flow diagram.

For DSK 6416:
- Write DSP software to convert 24 bit Image.bmp bitmap file into simple binary file as raw data. Store raw data into the SDRAM bank 0 of the EVM board. The image.bmp file can be read from Host machine, using JTAG. Draw Software flow diagram.

- Write DSP software to convert 24 bit raw data of Image.bmp bitmap file into 8 bit raw data. Store 8 bit raw data into the SDRAM bank 1 of the EVM board. Measure the timings of this conversion and make it as optimum as you can. Use DSP/BIOS, DMA transfer, task, semaphore and interrupts. Draw Software flow diagram.
- Write DSP software to convert 8 bit raw data from SDRAM bank 1 of EVM board into the 24-bit bitmap image (pad remaining 16 bits with 0). Put this converted data as a bitmap file into host through JTAG. Draw Software flow diagram.

Note: Image.bmp file and bitmap file format are available.
8-bit data format: R7 R6 R5 G7 G6 G5 B7 B6
Follow the standard programming practice.
You need EVM6201 for this assignment.

BITMAP FILE FORMAT

Field	Bytes from the first location	Meaning of the field
Header (54 bytes)	2 bytes	0x4D42
	4 bytes	Number of bytes per line * number of lines + 54
	2 bytes	0x0000
	4 bytes	0x00000036
	4 bytes	0x00000028
	4 bytes	Total number of pixels per line
	4 bytes	Total number of lines
	2 bytes	0x0001
	4 bytes	0x00000018 (24 bit resolution)
	2 bytes	0x0000
	4 bytes	Number of bytes per line * number of lines
	4 bytes	0x00000EC4
	4 bytes	0x00000EC4
	4 bytes	0x00000000
	4 bytes	0x00000000
24 bit RGB Data With 0x00 padding If required	Number of bytes per line	1 byte for B 1 byte for G 1 byte for R (Up to number of pixels per line) + Padding bytes (maximum 3 bytes) (Padding value is 0x00

Note: If number of pixels per line * 3 is not multiple of 4 then there is padding of 0x00 in each line to make it multiple of 4.