EXPERIMENT 10

Introduction to Texas Instruments (TMS320C6713 DSP STARTER KIT (DSK)) Digital Signal Processing Board

Objective

In this experiment you will become familiar with a development system for programming DSP hardware in order to implement DSP operations.

You will study:

- Code Composer Studio
- TMS320C6713 DSP chip and supporting chip set (DSK) architecture
- The C programming language

1.1 Code Composer Studio

CCS consists of a set of software tools and libraries for developing DSP programs, compiling and linking them into machine code, and writing them into memory on the DSP chip and on-board external memory. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs.

It also contains diagnostic tools for analyzing and tracing algorithms as they are being implemented on-board. In this lab we will always use the CCS to develop, compile, and link the programs that are downloaded from the computer to DSP hardware.

1.2 TMS320 DSP Chips

In this lab we will be using C6713 which is a 225MHz processor having the ability to run 8 32-bit parallel instructions in one 4.4ns instruction cycle, for the instruction rate of 1800 million instructions per second.

The different families of TMS320 series are used for different applications. C2000 and C5000 series of chips are used in the portable devices because they consume very little power, and are primarily used for the digital control. C2000 and C5000 series chips are used in 3G mobile phones, GPS receivers, portable medical equipment and digital music players. The C6000 series of chips

provides both fixed and floating point processors that are used in systems that require high performance. These chips are not power efficient, so they are not used in the portable devices instead they are used in the high quality digital audio applications, broadband infrastructure and digital imaging.

1.3 DSP Starter Kit (DSK)

For the development of the programs in the TMS320 DSP chip we need a supporting architecture that store the programs and data and bring signals on and off the board. In order to use this DSP chip, the Spectrum Digital, Inc, has manufactured a circuit board containing appropriate components. The C6713 DSK allows you to download and step through code quickly and uses Real Time Data Exchange (RTDX™) for improved Host and Target communications. The full contents of the kit include:

- 512K of non-volatile FLASH memory with 16MB of on-board SDRAM
- 16-bit stereo codec
- Four 3.5-mm audio jacks
- Standard JTAG interface with 8-pin connector
- Plug-and-play JTAG support via USB
- Expansion port connectors for plug-in modules
- +5-volt universal power supply
- Three power cords (standard US, UK, & European)
- Code Composer Studio IDE, version 3.1
- Fast simulators enabling cache analysis and multi-event profiling
- Power Analyzer support
- MATLAB
- Support for third party daughter cards
- DSK quick start guide and technical reference

1.4 Programming Languages

Programming of the DSP chip is done in C using the Code Composer Studio (CCS) integrated development environment. Assembly language was once the most commonly used programming language for DSP chips and microprocessors because it enables the programmer to manage the CPU core registers and schedule events in the CPU core. Assembly language is most time consuming and is specific to the given architecture. Most of the programs created in this course will be coded in C. In CCS, the C compiler has four optimization levels. The highest level of optimization will not achieve the same level of optimization that programmer-optimized assembly programs will, but Texas Instrument has done a good job in making the optimized C compiler produce code that is comparable to programmer- optimized assembly code.

1.5 Codec

A codec is a device capable of encoding or decoding a digital data stream or signal. It is a chip located on-board DSK which interfaces with the analog world like signal generators and stereo headphones. The codec contains a coder, or analog-to-digital converter (ADC), and a decoder or digital-to-analog converter (DAC). Both coder and decoder run at sample rates which can be set from 8 KHz to 96 KHz and support data word lengths of 16b, 20b, 24b, and 32b at the digital interfaces.

1.6 C6713 DSP Chip

The C6713 DSP chip is a 225 MHz floating point processor that has a CPU, internal memory (L1/L2 Memory Architecture), 1 32-bit external memory interface (EMIF), 2 32-bit general purpose timers, 2 Multi-channel buffered serial ports McBSP, 2 McASP, Host port interface (HPI), interrupt selector, Flexible phase lock loop (PLL) based block generator module along with hardware for the Boot Configuration and Power Down Logic.

1.7 Timing

The DSP chip must be able to establish communication links between the CPU (DSP core), the codec, and memory. The two McBSPs, serial port 0 (SP0) and serial port 1 (SP1), are used to establish bidirectional asynchronous links between the CPU and the codec. SP0 is used to send control data between the codec and CPU, SP1 plays a similar role for digital audio data.

1.8 Support Files

The following support files located in the folder support (except the library files) are used:

- **C6713dskinit.c:** contains functions to initialize the DSK, the codec, the serial ports, and for I/O. It is not included with CCS.
- **C6713dskinit.h:** header files with function prototypes. Features such as those used to select the mic input in lieu of line input (by default), input gain, and so on are obtained from this header file (modified from a similar file included with CCS).
- **C6713dsk.cmd:** sample linker command file. This generic file can be changed when using external memory in lieu of internal memory.
- **Vectors intr.asm:** a modified version of a vector file included with CCS to handle interrupts. Twelve interrupts, INT4 through INT15, are available, and INT11 is selected within this vector file. They are used for interrupt-driven programs.

- **Vectors_poll.asm:** vector file for programs using polling.
- **rts6700.lib, dsk6713bsl.lib, csl6713.lib:** run-time, board, and chip support library files, respectively. These files are included in the CCS are located in C6000\cgtools\lib, C6000\dsk6713\lib, and c6000\bios\lib, respectively

In this folder u:\DSP\Lab 07\ copy the file named led.c. Now you have support files and files required to our first project.

2. Creating the First Project on the DSK

2.1 Creating the Project File Lab7.pjt

In CSS, select 'Project' and then 'New'. A window named 'Project Creation' will appear. Fill all the fields according to the figure given below. Finally, click on 'Finish'. CCS has now created a project file Lab7.pjt, which will be used to build an executable program. This file is stored on your u:\ drive in the path

u:\DSP\lab07. See Figure 2.1 below.

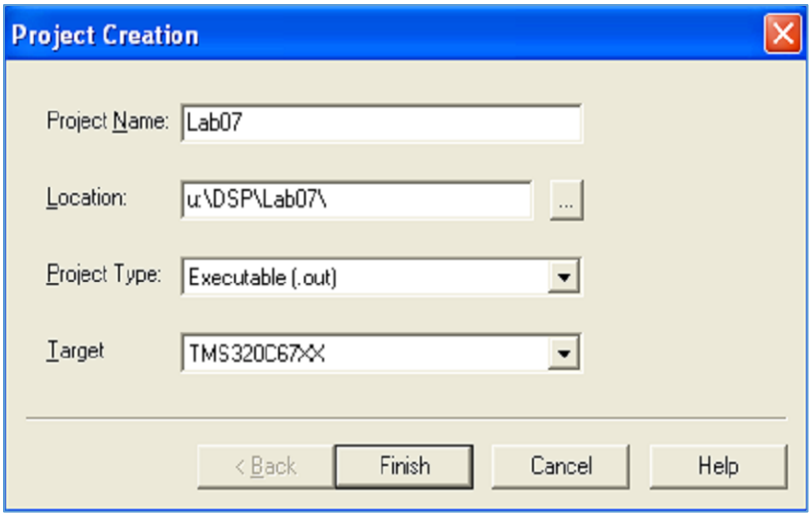


Figure 2.1: Project creation window

2.2 Adding Support Files to a Project

The next step in creating a project is to add the appropriate support files to the file lab7.pjt. In the CCS window, go to 'Project' and then 'Add Files to Project . . .'. In the window that appears, click on the folder next to where it says 'Look In:'. Make sure that you are in the path u:\DSP\Support. You should be able to see the file c6713dskinit.c. Notice that the 'Files of type' field is 'C source code'. Click on c6713dskinit.c and then click on 'Open'. Repeat this process two more times adding the files Vectors intr.asm and c6713dsk.cmd to the project file lab7.pjt. For field, 'Files of type', select 'Asm Source Files (*.a*)'. Click on Vectors intr.asm and then click on 'Open'. For

field, ‘Files of type’, select ‘Linker Command File (*.cmd)’. Click on c6713dsk.cmd and then click on ‘Open’. You have now charged your project file u:\DSP\Lab 07\Lab7{lab7.pjt}.

The C source code file contains functions for initializing the DSP and peripherals. The Vectors file contains information about what interrupts (if any) will be used and gives the linker 9 information about resetting the CPU. This file needs to appear in the first block of program memory. The linker command file (c6713dsk.cmd) tells the linker how the vectors file and the internal, external, and flash memory are to be organized in memory. In addition, it specifies what parts of the program are to be stored in internal memory and what parts are to be stored in the external memory. In general, the program instructions and local/global variables will be stored in internal random access memory or IRAM.

2.3 Adding Appropriate Libraries to a Project

In addition to the support files that you have been given, there are pre-compiled files from TI that need to be included with your project. For this project, we need run-time support libraries. For the C6713 DSK, there are three support libraries needed: csl6713.lib, dsk6713bsl.lib, and rts6700.lib. The first is a chip support library, the second a board support library, and the third is a real-time support library. Besides the above support libraries, there is a GEL (general extension language) file (dsk6211 6713.gel) used to initialize the DSK. The GEL file was automatically added when the project file lab7.pjt was created, but the other libraries must be explicitly included in the same manner as the previous files. Go to ‘Project’ and then ‘Add Files to Project’. F or ‘Files of type’, select ‘Object and Library Files (*.o*,*.l*)’. Navigate to the path u:\DSP\Support and select the files csl6713.lib,. .... In the left sub-window of the CCS main window, double-click on the folder ’Libraries’ to make sure the file was added correctly.

These files, along with our other support files, form the black box that will be required for every project created in this class. The only files that change are the source code files that code a DSP algorithm and possibly a vectors file.

2.4 Adding Source Code Files to a Project

The last file that you need to add to led.pjt is your C source code file. This file will contain the code that is needed to control the LEDs through the DIP switch. Go back to ‘Project’ and then ‘Add Files to Project . . .’, but this time browse to the path u:\DSP\Lab 07\. Click on the file led.c and add it to your project by clicking on ‘Open’.

You may have noticed that the .h files cannot be added – there is no ’Files of type’ entry for .h files. Instead, they are added in the following manner: go to ‘Project’ and select ‘Scan All Dependencies’. In CCS, double-click on ‘led.pjt’ and then double-click on ‘Include’. You should see any header files on an #include line in your source files (including c6713dskinit.c) plus approximately 14 other header files found by the scan step. The latter files are supplied with the Code Composer Studio software and are used to configure the DSP chip and board. CCS

automatically found and included all needed header files starting from the header files included in the source files. Open c6713dskinit.c and observe that it includes c6713dskinit.h; this header and dsk6713 aic23.h (included in led.c) include other header files which in turn include others, leading to the list observed. Note that some include files are prefixed csl; these are chip-support header files supplied by TI. The file dsk6713.h is a top level header file supplied by the DSK board manufacturer, Spectrum Digital, Inc. The project file lab7.pjt has now been charged with all of the files required to build the first executable .out file.

2.5 Build Options

The next objective is to customize the compiler and linker options so that the executable file gets built correctly. Also, the compiler will first convert the C coded programs into DSP assembly programs before it compiles them into machine code. By selecting the appropriate options as shown in Figure 2.2 and 2.3, we can keep these intermediate assembly files.

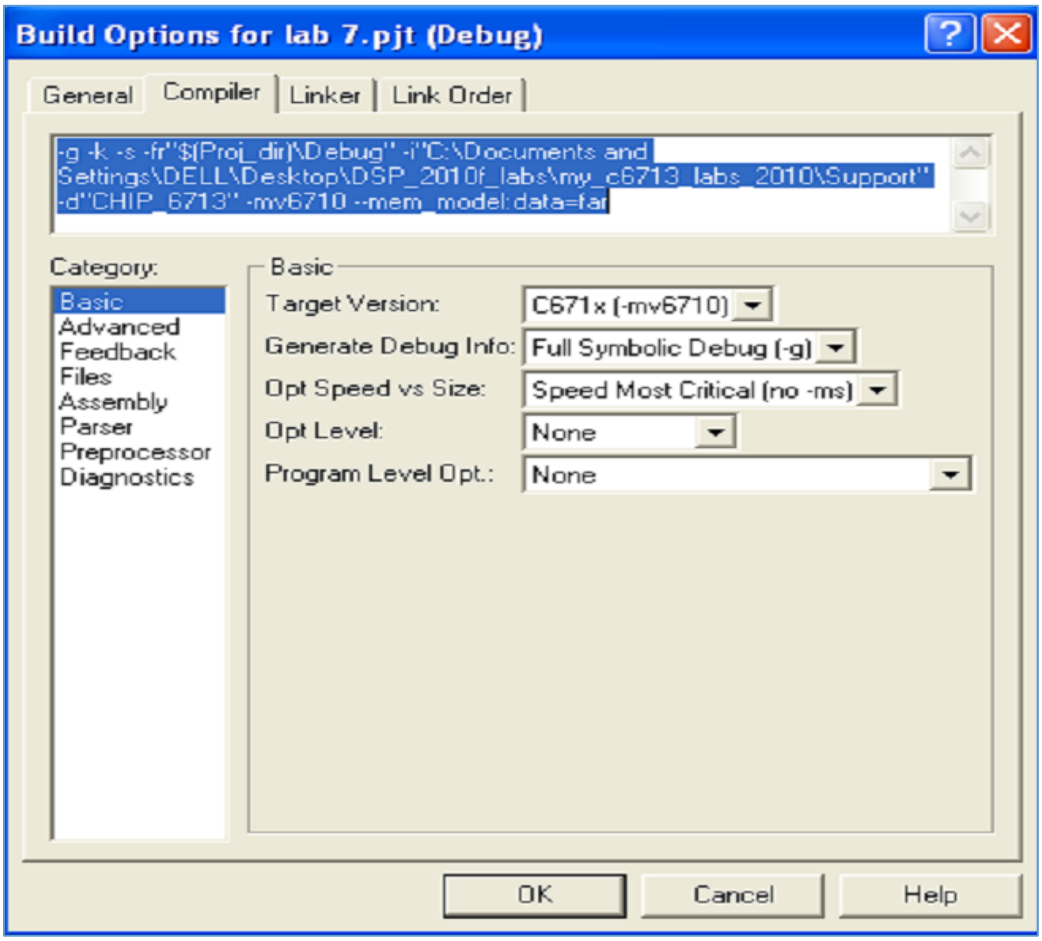


Figure 2.2

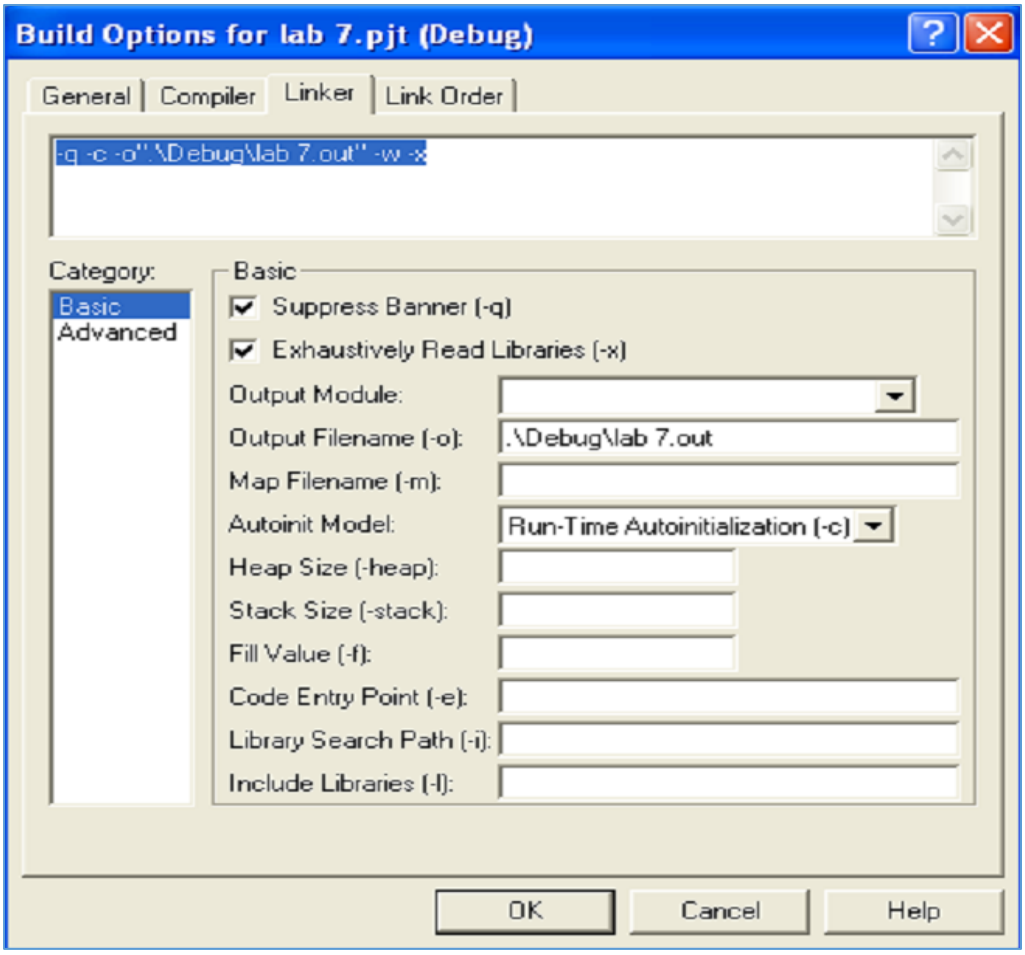


Figure 2.3

3. Building and Running the Project

Now you must build and run the project. To build the first project, go to 'Project' pull down menu in the CCS window, then select 'Build' (or press the button with three red down arrows on the top toolbar in the CCS window). A new sub-window will appear on the bottom of the CCS window. When building is complete, you should see the following message in the new sub-window:

- Build Complete,
- 0 Errors, 2 Warnings, 0 Remarks.

When CCS "built" your project, it compiled the C coded source files and header files into assembly code, using a built-in compiler. Then it assembled the assembly code into a COFF (common object file format) file that contains the program instructions, organized into modules. Finally, the linker organized these modules and the run-time support library (rts6701.lib) into memory locations to create the executable .out file led.out. The executable file, led.out, may be downloaded onto the

DSK. When led.out is loaded onto the DSK, the assembled program instructions, global variables, and runtime support libraries are loaded to their linker-specified memory locations.

The following steps builds and runs the LED project on our DSP board.

In order to test the program led.out on the DSK (DSP starter kit), it must first be loaded onto the DSP board. Before a new program is loaded onto the board, it is good practice to reset the CPU. To reset the CPU, click on the 'Debug' pull-down menu and select 'Reset CPU'.

STEPS:

1. First of all, run the 6713DSK Diagnostics to check the diagnostic status of DSP Board.
2. Then start CCS_v3.1 code composer studio software. Click on the 'Project' pull down menu and select Open Project from where you select **led.pjt.** This **led.pjt** project now appears under the projects heading on the left-hand side.
3. After that you create your **led.c** source file which has the relevant C code for running your LEDs project. (File > New > Source)
4. Then you build your **led.c** Program to check for any errors and warning and for creating the **led.out** file which will be stored in **Debug** folder.
5. Then you load your program **led.out** file onto the DSP starter kit. (File > Load Program) and then build your Program to recheck for any errors or warnings. This will download the executable file led.out onto the DSK.
6. In the end, you run your **led.pjt** project by pressing the F5 key or 'running man' icon on the left-hand side toolbarto show the result of your program on DSP board.
7. Verify by pressing down the **DIP switch # 3** that which led turns **ON**. After you have completed this task, go to the **'Debug'**pull down menu to select **'Halt'** to stop the execution of the program.

Exercise 1:

Task on LEDs operation on DSP Board:

In this Lab, we will study the blinking sequence of LEDs on DSP Board by running the project led.pjt which shows the sequence of LEDs operation.

This task blinks **LED #0** at a rate of about 2 times per second using the **LED module** of the DSK6713 Board Support Library (bsl). The example also reads the state of **DIP switch #3** and lights **LED #3** if the switch is depressed or turns it off if the switch is not depressed.

The Board Support Library is divided into several modules, each of which has its own include file. The file dsk6713.h must be included in every program that uses the BSL.

This task also includes dsk6713_led.h and dsk6713_dip.h because it uses the LED and DIP switches modules.

```
#include "dsk6713.h"
#include "dsk6713_led.h"
#include "dsk6713_dip.h"
void main()
{
   /* Initialize the board support library, must be first BSL call */
   DSK6713_init();
   /* Initialize the LED and DIP switch modules of the BSL */
   DSK6713_LED_init();
   DSK6713_DIP_init();
while(1)
   {
      /* Toggle LED #0 */
      DSK6713_LED_toggle(0);
      /* Check DIP switch #3 and light LED #3 accordingly, 0 = switch pressed */
if (DSK6713_DIP_get(3) == 0)
         /* Switch pressed, turn LED #3 on */
         DSK6713_LED_on(3);
else
         /* Switch not pressed, turn LED #3 off */
         DSK6713_LED_off(3);
      DSK6713_waitusec(200000);/* Spin in a software delay loop for about 200ms */
   }
}
```

The steps to perform the LED blinking sequence on DSP Board are shown pictorially below:

1. Run the **6713DSK** diagnostics as shown in Figure 2.4 to check the diagnostic status of DSP Board.

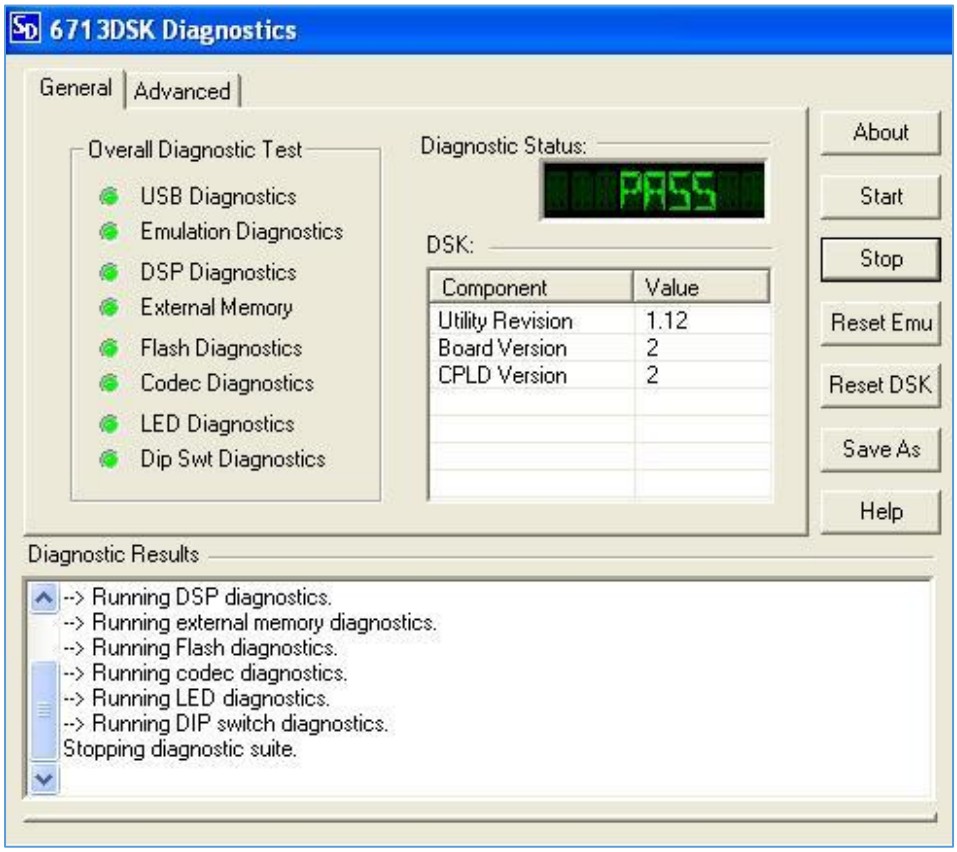


Figure 2.4

2. Load the project **led.pjt** as shown in Figure 2.5by going on Project > Open.

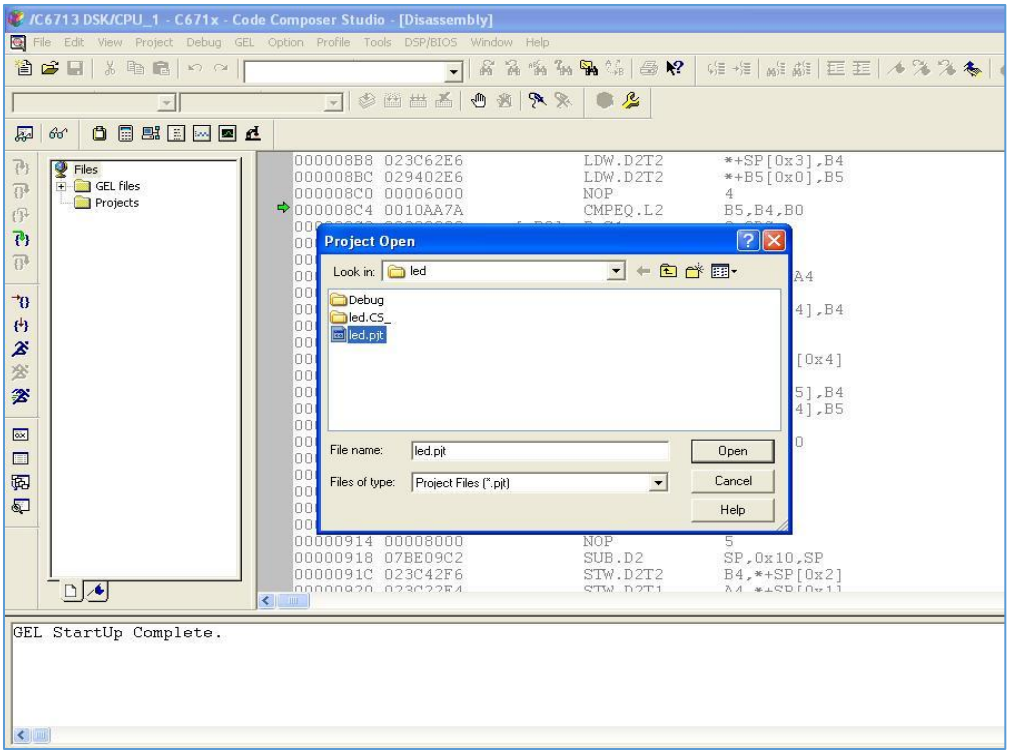


Figure 2.5: Loading the project led.pjt

3. Load the source file **led.c** by going on File> Source. See Figure 2.6.

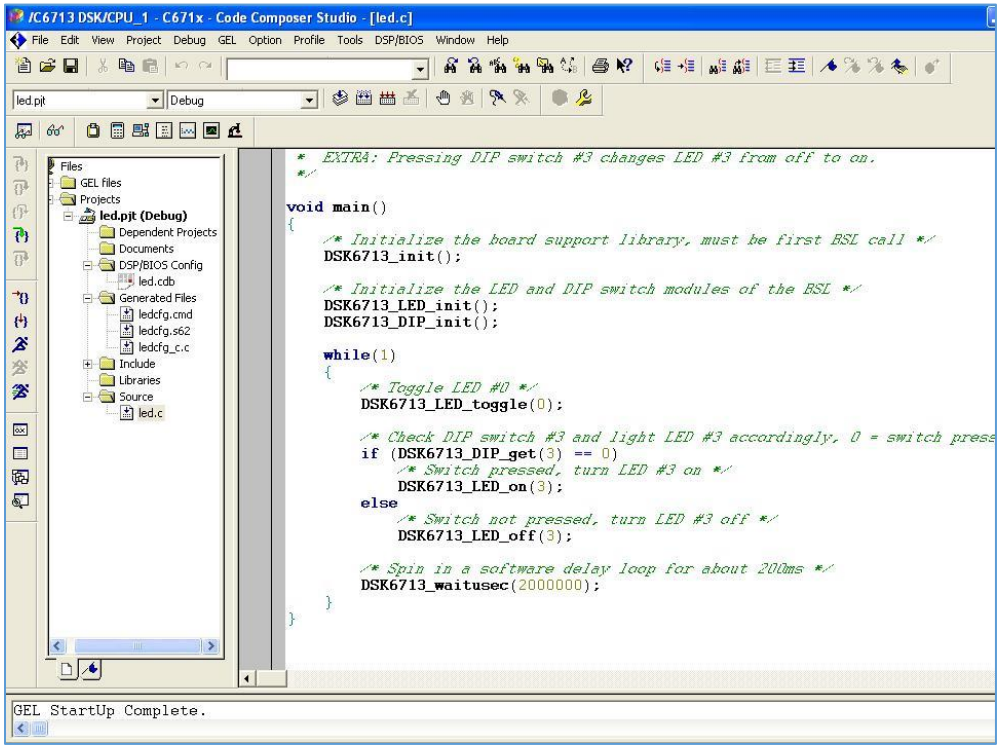
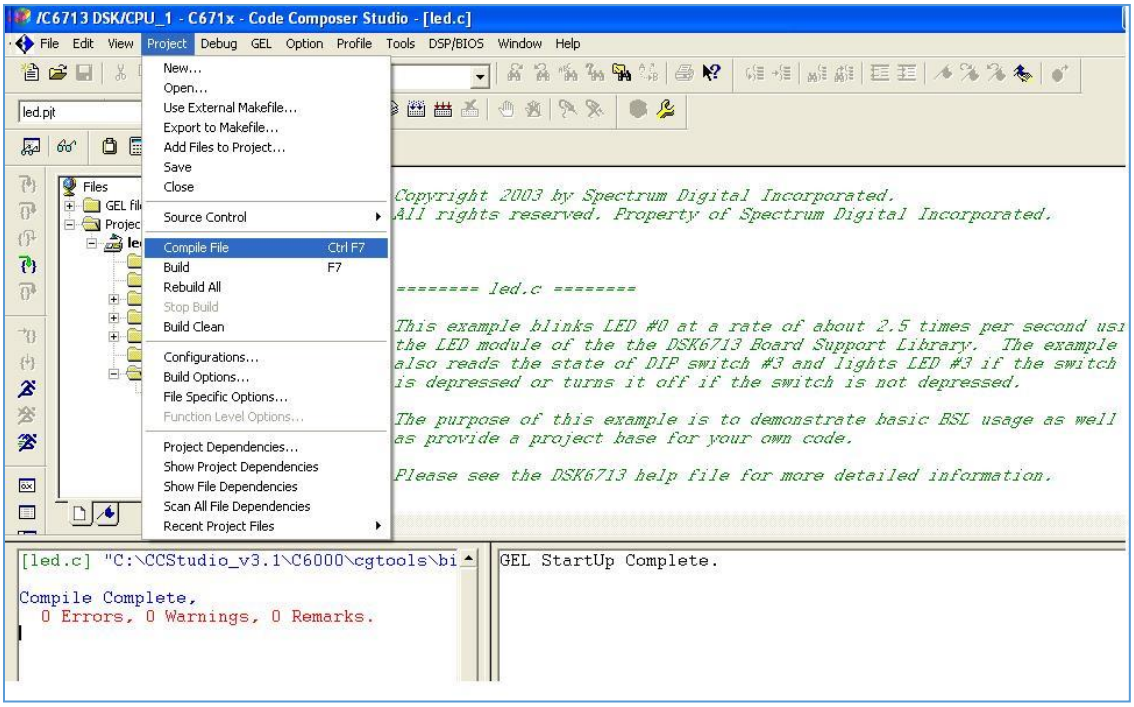


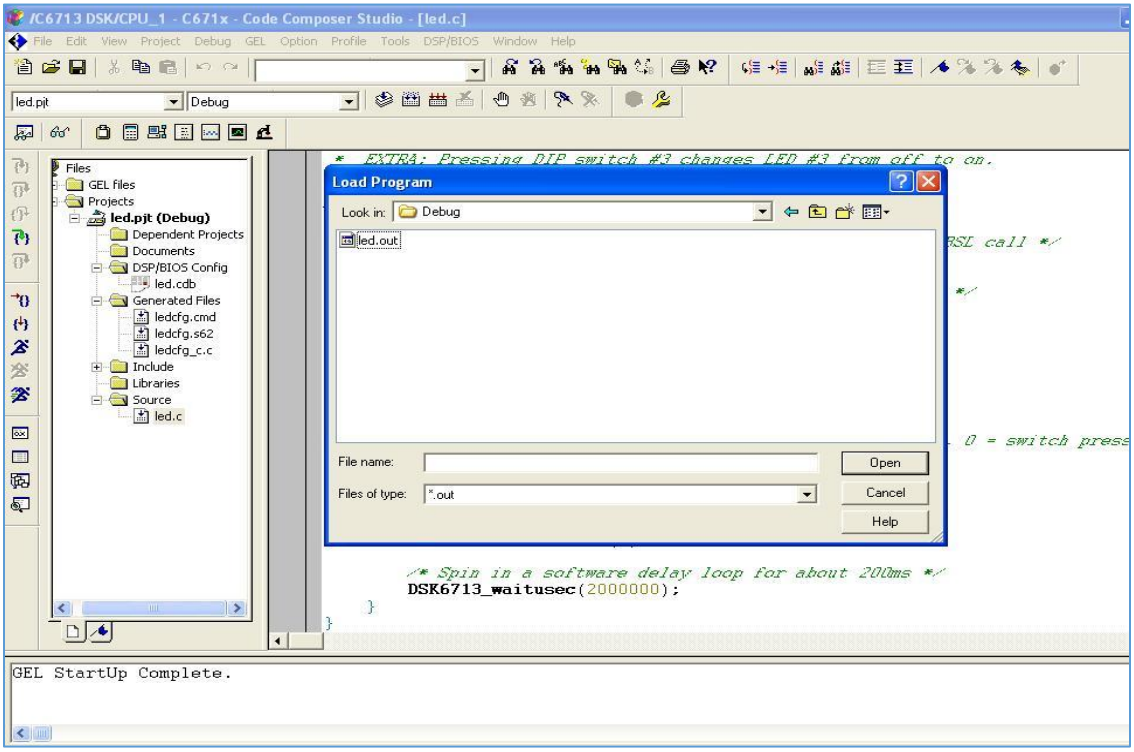
Figure 2.6: Loading the source file led.c

4. Compile your File(led.c project) as shown in Figure 2.7to check for any errors or warnings.



Compiling led.c file:Figure 2.7

5. Load Program **led.out** on the DSP Board as shown in Figure 2.8.



Loading program led.out:Figure 2.8

6. Build your project as shown in Figure 2.9.

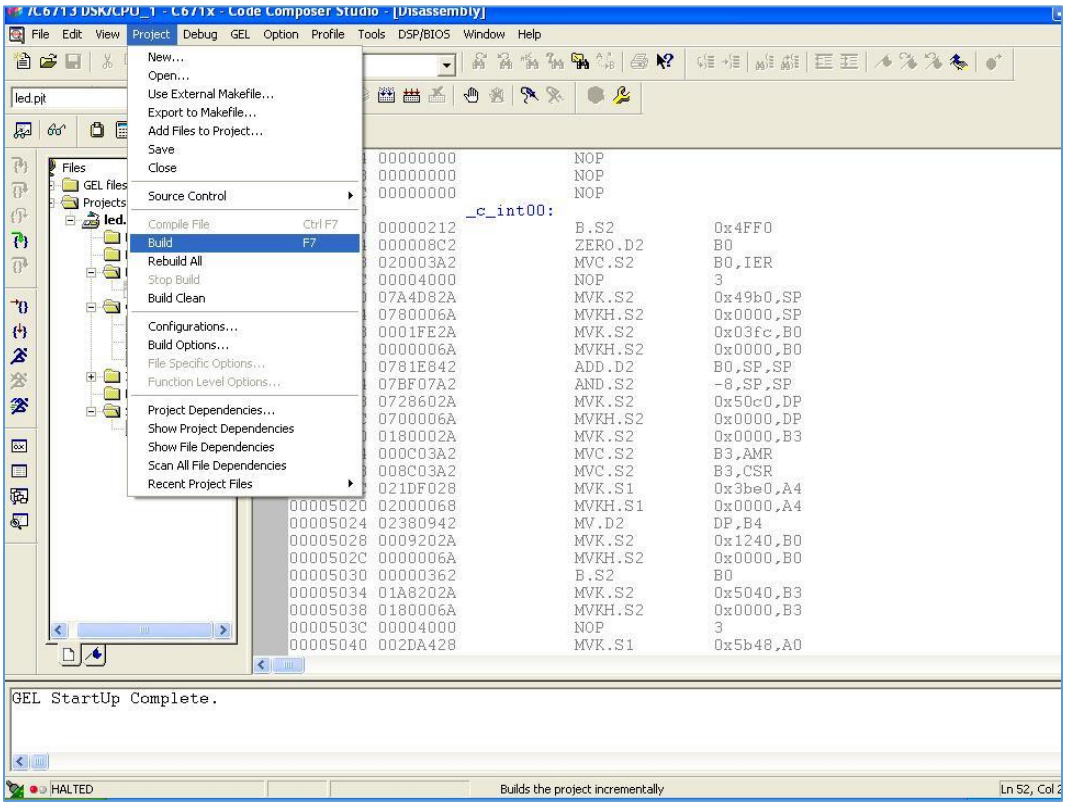


Figure 2.9

7. Run your project (See Figure 2.10) and observe the blinking sequence of LEDs.

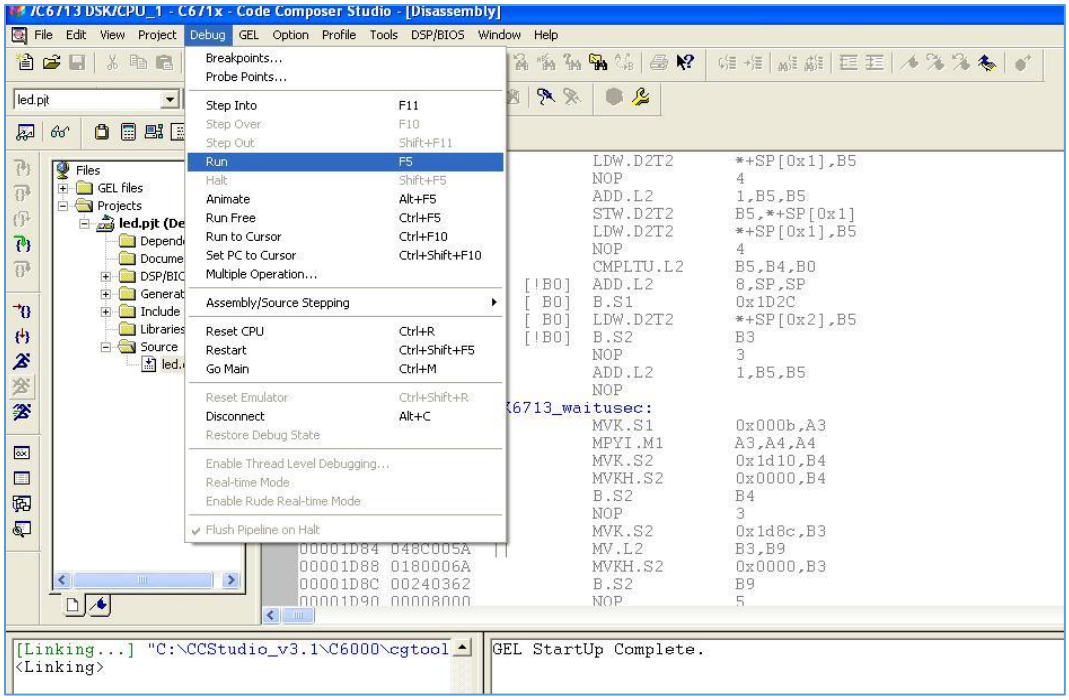


Figure 2.10

The Figure 2.11 and Figure 2.12 below show the LED blinking sequence. First, LED 1 blinks and then LED 3 lights up if DIP switch 3 is depressed.

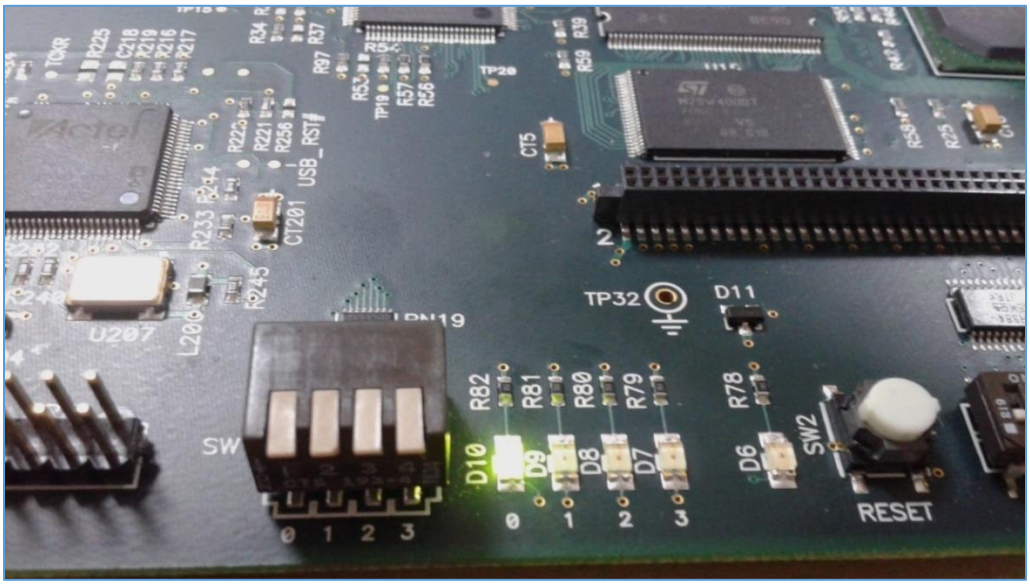


Figure 2.11

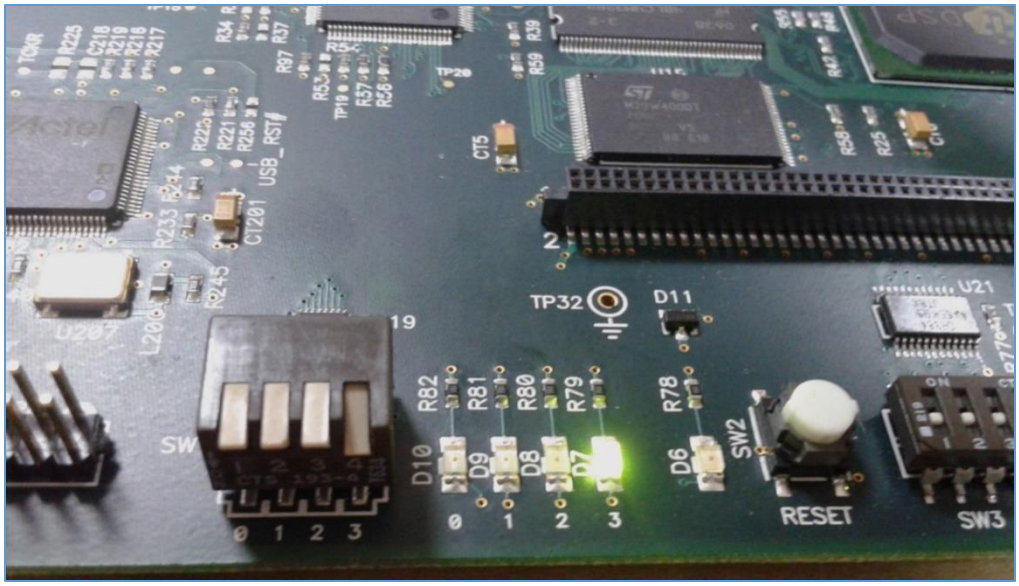


Figure 2.12

Exercise 2:

Write a C code that performs the following operation on LEDs on DSP Board. For each question attach the code in the space provided.

1. LED # 2 toggles (blinks) for the first 30 seconds. Then, the DSP Board monitors the state of DIP switches on DSP Board.

   If Switch # 1 is ON and Switch # 3 is OFF, it will turn LED # 1 and LED # 2 ON, otherwise all LEDs will be in OFF state.

2. The following LEDs turn ON for 5 seconds in the following sequence 3, 2, 1 (LED # 3 turns ON for 5secs and then turns OFF, then LED # 2 turns ON for 5 secs and then turns OFF, then LED # 1 turns ON for 5secs and then turns OFF).

   After that LED # 0 blinks for 15 secs and then this loop or sequence is repeated.

Post-Lab Exercise:

Write a paragraph in the space provided below on TMS320C6713 DSP board.

Main points to focus are:

I. TMS320C6713 DSP board main components

II. Features

III. Importance

IV. Applications

V. Latest TMS320C6XXX board available