NANYANG POLYTECHNIC
EGR204 Microcontroller Applications

Laboratory Session 1

Course: Diploma in Robotics and Mechatronics

Module: EGR204 Microcontroller Applications

Experiment: 1

Title: Using the 8051 Development Tools

Objective:

- The students will learn how to use the Silicon Laboratories Integrated Development Environment.

Learning Objectives:

- Demonstrate how to setup a working directory.
- Demonstrate how to create a project file and add and create files in the project.
- Demonstrate how to compile and link a project.
- Demonstrate how to connect to target system and run the project on it.

1. Introduction

Traditionally, program for embedded system are written in assembly language. However, the structured nature and ease of use of the C language make it a more suitable choice as a programming language for embedded application. Some aspects of the C programming language and standard libraries of the 8051 C Cross Compiler are altered or enhanced to address the peculiarities of the embedded target processor.

In this laboratory session, we will familiarise ourselves with the Silicon Laboratories Integrated Development Environment.

2. Setting up the working directory

Before we start creating software, let's setup a working directory as your personal workspace for the semester.

2.1 On your Windows desktop, create a new folder and call it "EGB204". To do it, you can right click on the mouse to open a pop-up menu, choose menu item "new", select menu sub-item "Folder", and you should see a new folder created on the desktop. Rename it as "EGB204".

2.2 This folder will be known as the **working directory** or **user directory**.

2.3 Next, navigate to folder "C:\users\header_files". You should see two files:

1) F200.h (header file specific for our hardware), and
2) FUNC.c (source code specific for our hardware).

2.6 Copy both files to your working directory. After this operation, you have completed setting up your working directory.

Note: <u>You are only required to setup this directory once for the entire semester.</u>

3. Starting the Silicon Laboratories IDE

Next, we are ready to learn the Silicon Laboratories Integrated Development Environment (IDE). A typical IDE comprises of

1) a text editor,
2) a compiler,
3) a linker, and
4) a debugger,

(or ways to invoke them from within the IDE application). On the computer desktop, you should see the following application icon for Silicon Laboratories IDE (figure 3.1):

Figure 3.1

Click on the icon and you should see the Silicon Laboratories IDE application running (figure 3.2).

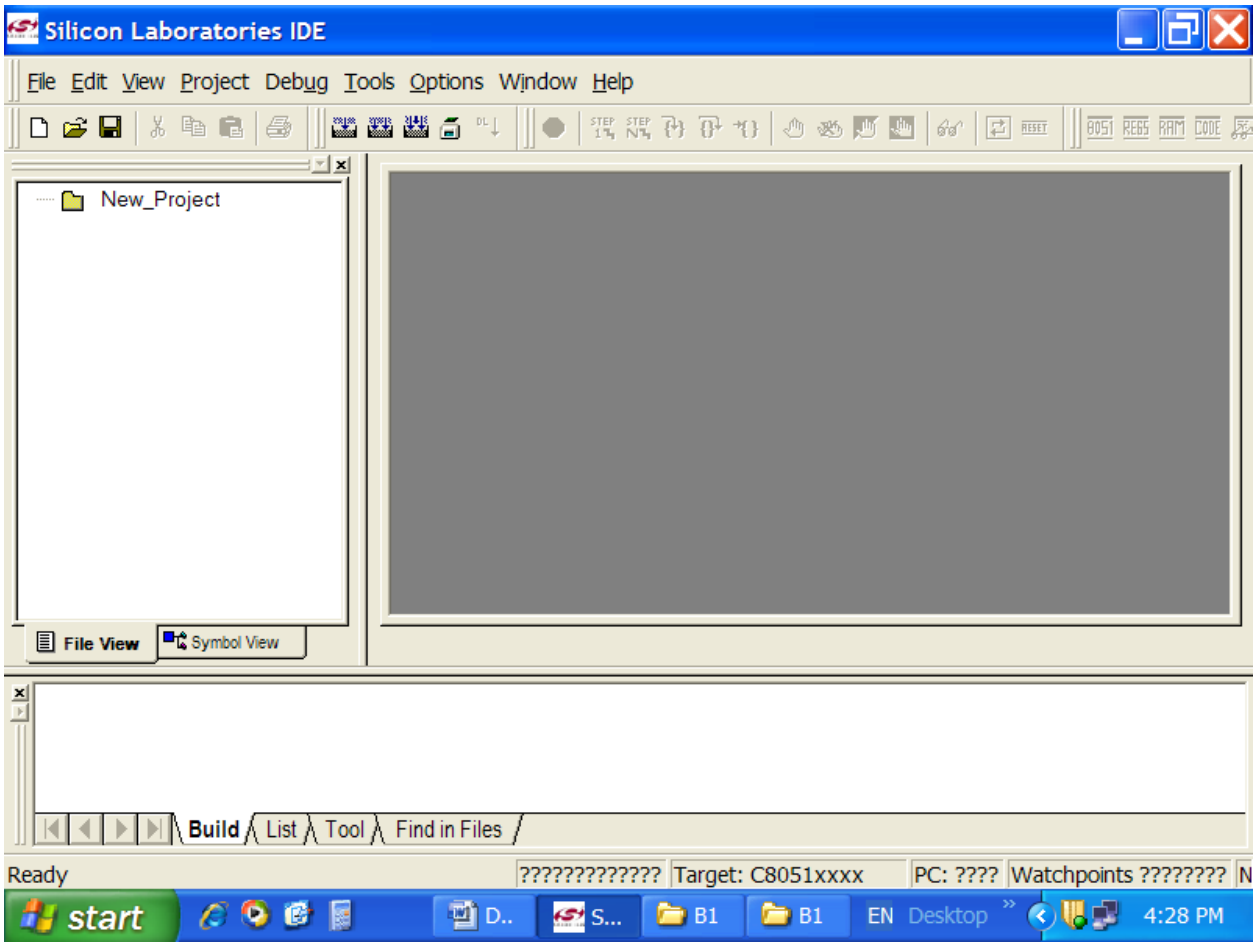

Figure 3.2

4. Creating a new project

While working on any C programming project, we will typically create many files. Silicon Laboratories IDE provides a convenient way to manage them by organising them into a project and allow files in it to be grouped into folders according to their file types. In this section, let's create a new project.

4.1 Click menu item "Project" and select item "New Project…" as illustrated in Figure 4.1. You should see dialog "New".

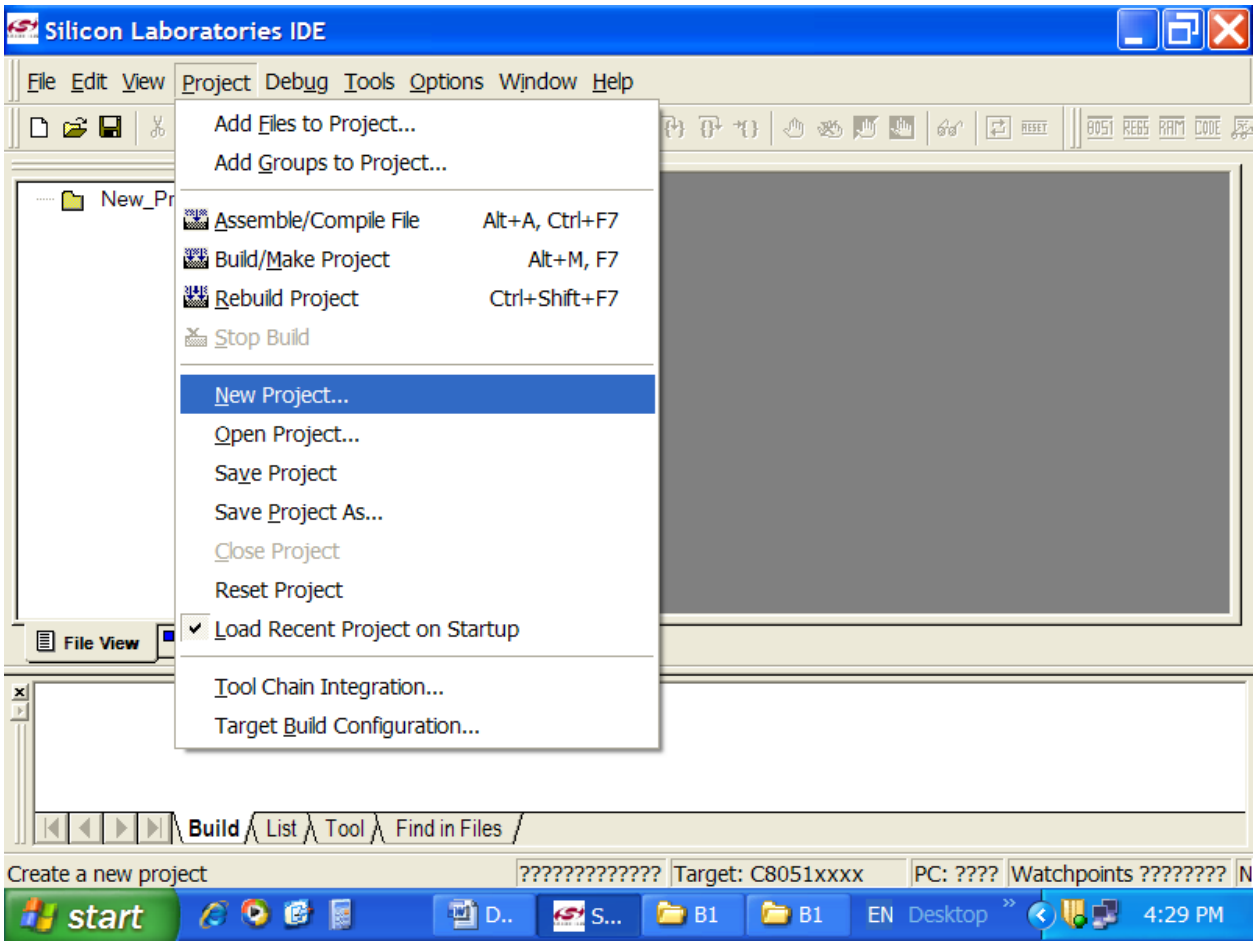

Figure 4.1

4.2 On the dialog "New", select or enter following values for the respective fields:

Device family: **C8051F2xx**
Project name: **<lab number>**
Location: **<your user directory>**
Project type: **Blank project**

Note: For naming your projects and files, please limit yourself to use alphabets, numbers and underscore. DO NOT use space or punctuations.

4.3 Click on button "OK" and you should see an empty project workspace created for you.

5. Adding existing files to the project

5.1 To add "F200.h" to the project, right-click on the folder "Header Files" and then, on the popup menu, select item "Add file to group Header Files", see figure 5.1. You should see dialog "Add File to project…" (see figure 5.2). In it, navigate to your working directory, select header file "F200.h" and click button "Open".

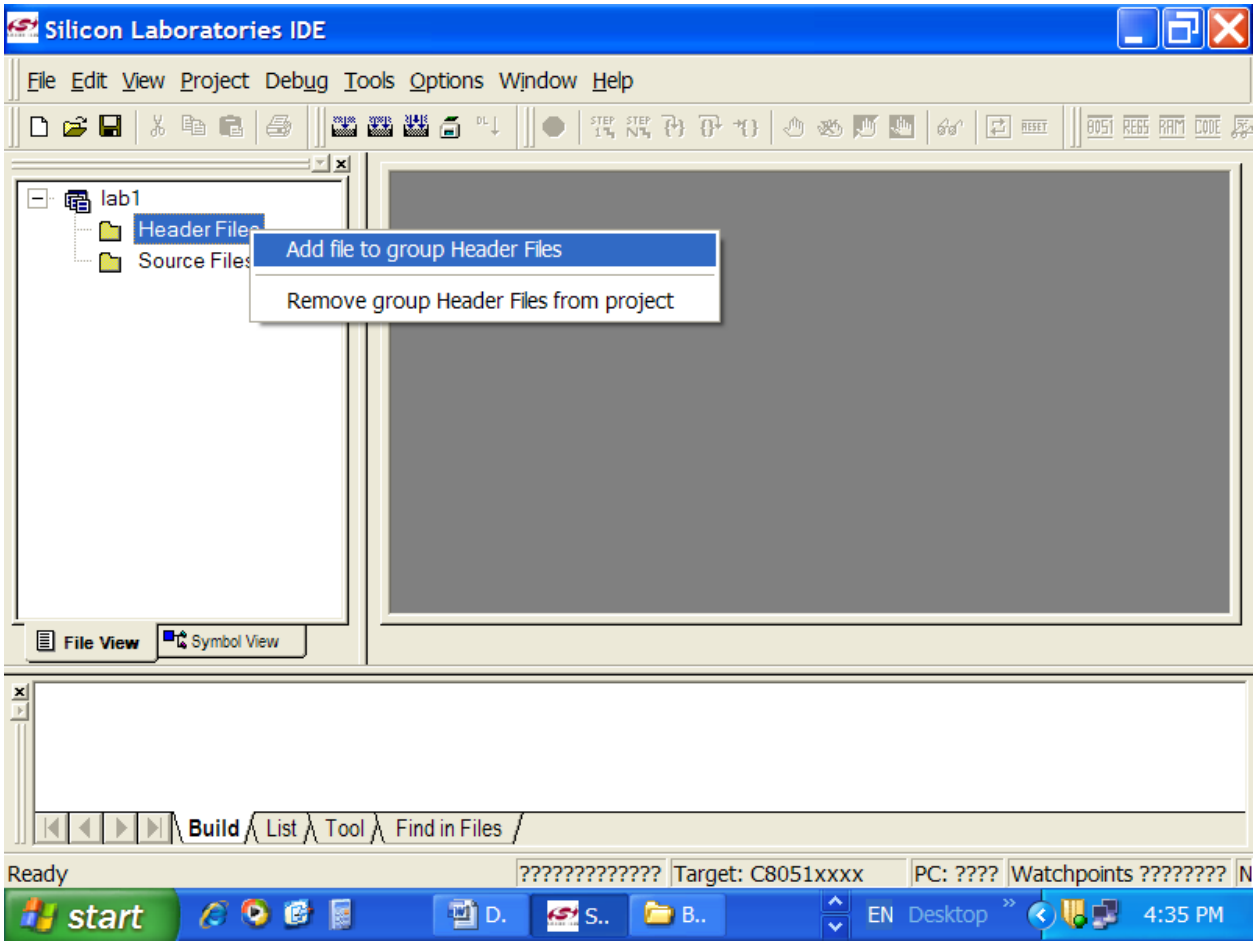

Figure 5.1

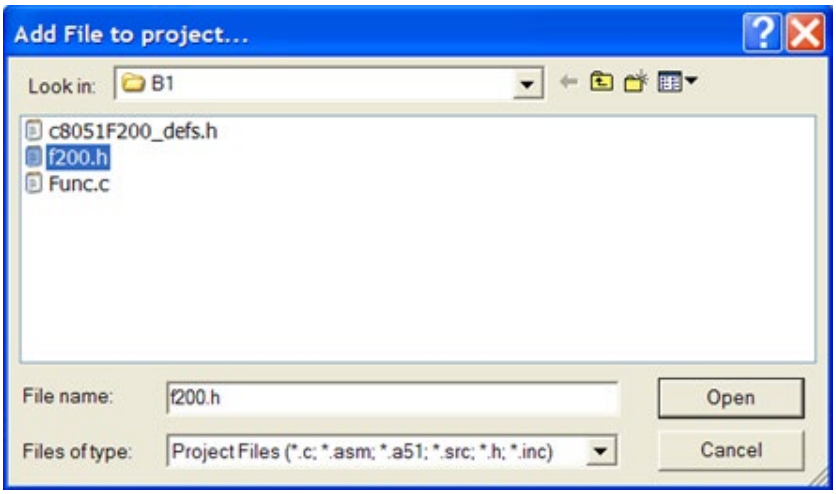

Figure 5.2

5.2 Next, try adding source file "FUNC.c" to folder "Source Files".

5.3 After adding a source file, we should add it to build list so that it is included in the build process, i.e. compilation and linking. Right click on source file "FUNC.c", on the popup menu, select item "Add Func.c to build" to include the source file.

5.4 By now, your IDE workspace should look like figure 5.3. Note the difference in the icons representing the header file and a "build-able" source file.

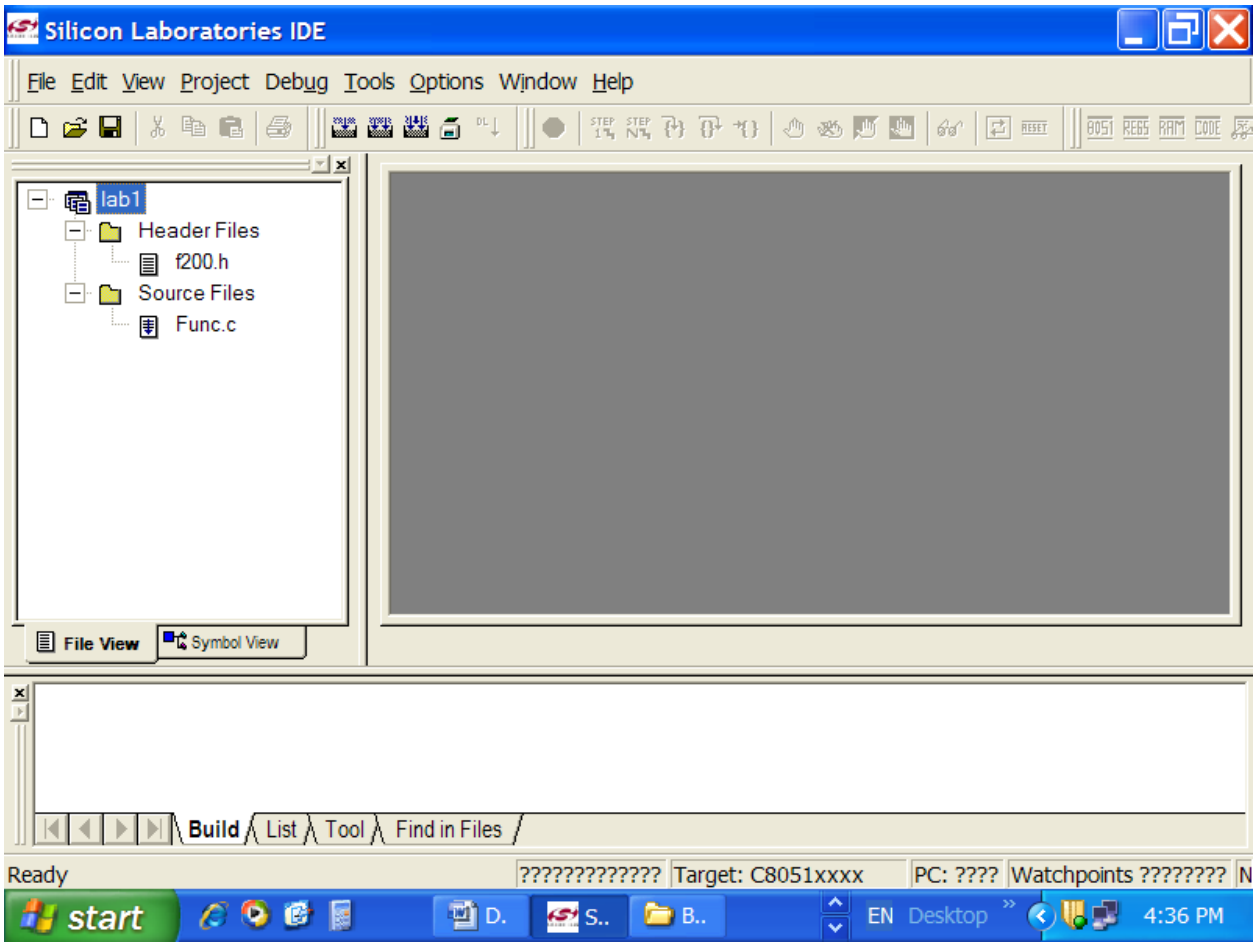

Figure 5.3

6. Adding new file to the project

6.1 To add a new file, click on the shortcut button shown in figure 6.1. You should see dialog "New" (see figure 6.2).

Figure 6.1

6.2 On the dialog "New", select or enter following values for the respective fields:

Files: **C Source File**
File name: **<lab number and exercise/assignment number>**
Location: **<your user directory>**
Add to project: **CHECK**
Add to build: **CHECK**

Note: For naming your projects and files, please limit yourself to use alphabets, numbers and underscore. DO NOT use space or punctuations.

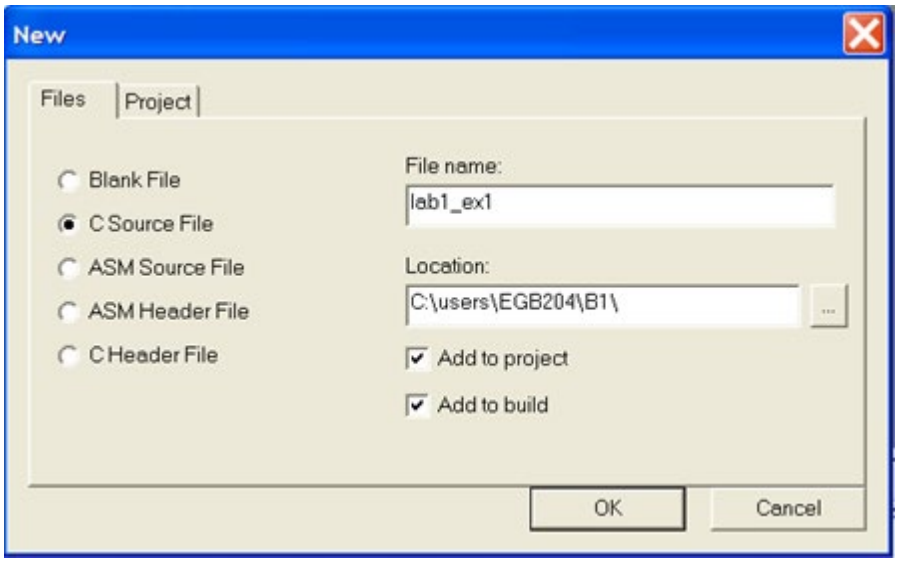

Figure 6.2

6.3 Click on button "OK" and you should see an empty file created for you (see figure 6.2).

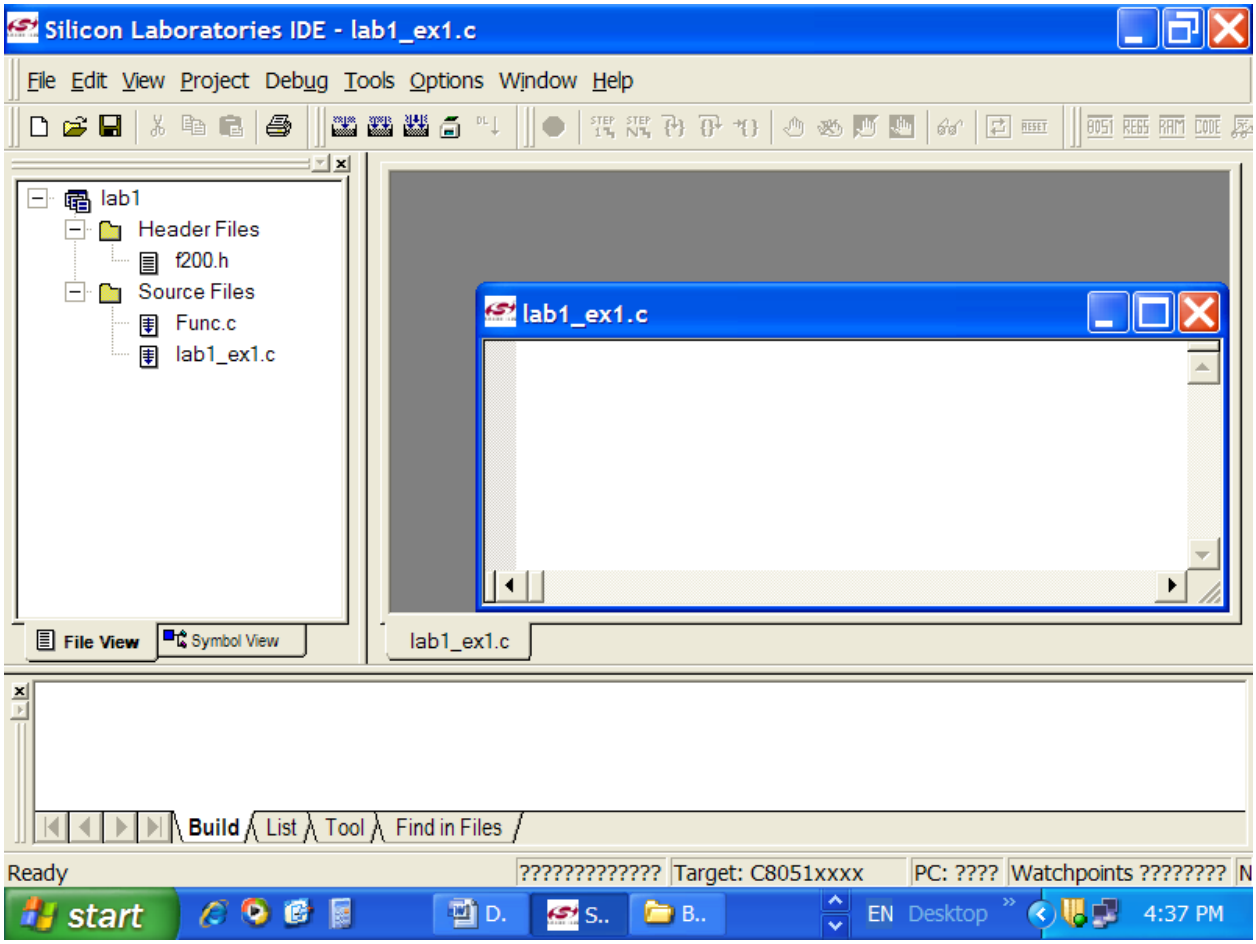

Figure 6.2

6.4 By now, your working directory should consist of four files (see figure 6.3).

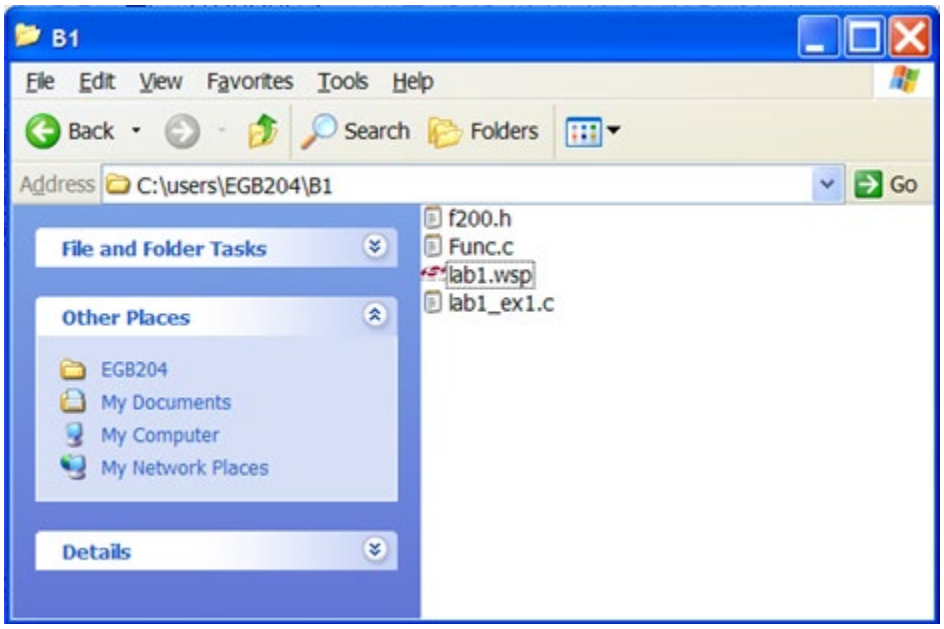

Figure 6.3

7. Coding, building and downloading

7.1 Type in the code listed in table 7.1 into the new file you just created.

```
#include <f200.h>

//----- Delay Subroutine -------
void delay(unsigned long duration)
{
      while ((duration--)!=0);
}

void setSystem();      //  refer to func.c for initialization for microcontroller chip

void main()
{
      unsigned int wait=60000;
      setSystem();                  // Initialize u-controller internal registry and clock
      P1=0x00;
      for(;;)
      {
            if (P02==0) wait=10000;
            if (P03==0) wait=30000;
            if (P04==0) wait=60000;

            P1=0x00;
            delay(wait);
            P1=0xff;
            delay(wait);
      }
}
```

Figure 7.1

7.2 To compile the code, click on the highlighted shortcut button shown in figure 7.2.

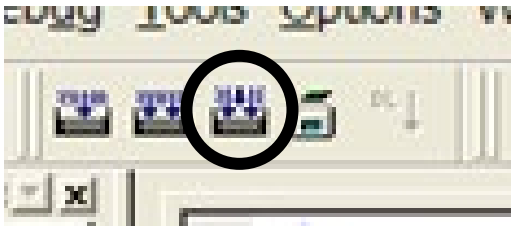

Figure 7.2

If your build process is successful, you will see the following messages in the output window:

LINK/LOCATE RUN COMPLETE WARNINGS (0) ERRORS (0)

And your working directory should look like figure 7.3.

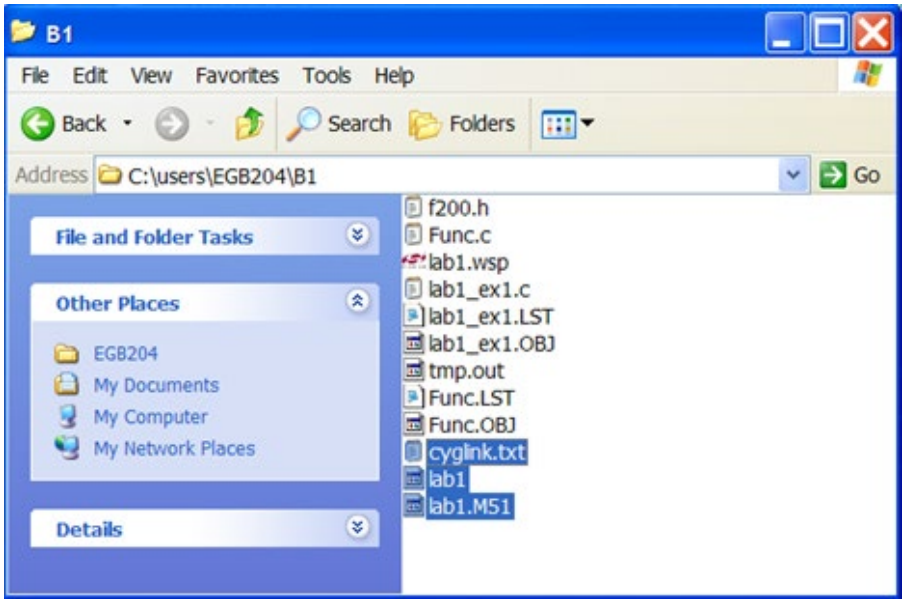

Figure 7.3

7.3 Next, we are ready to download the binary file to the target system. Connect your hardware to the host system by plugging the USB cable into the USB Debug Adapter on the target system. Next connect the power cable to the target system.

7.4 To establish the connection, click on the highlighted shortcut button shown in figure 7.4.

Figure 7.4

If successful, the shortcut button for downloading will be enabled, see figure 7.5.

Figure 7.5

If not, try configuring the connection options by clicking on menu item “Options”, selecting item “Connection Options”, and checking “USB Debug Adapter” in the serial adapter field (see figure 7.6).

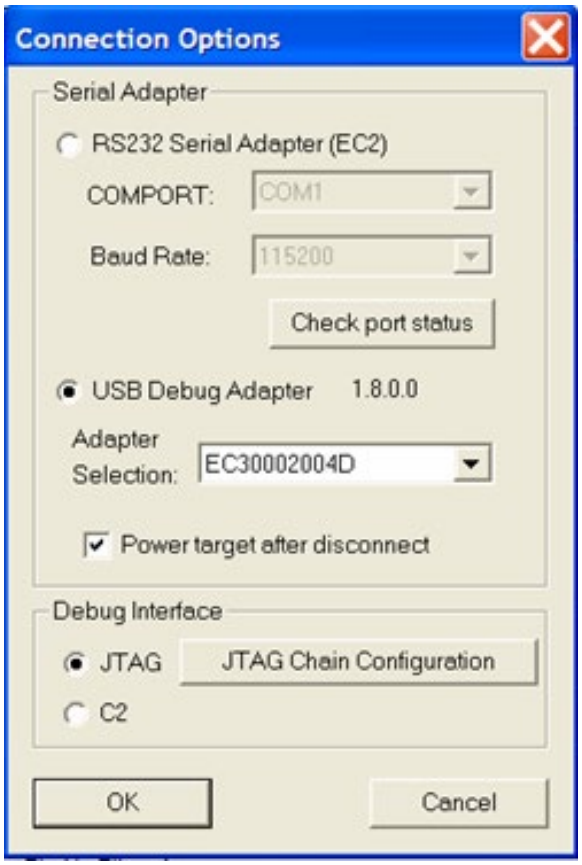

Figure 7.6

7.5 To perform a code download, click on the shortcut button for downloading. If successful, the shortcut button for running the code in target system will be enabled (in green), see figure 7.7.

Figure 7.7

7.5 To run the program, click on the shortcut button for running the code. If successful, the shortcut button will turn to be red in colour.

7.6 To stop running the program, click on the shortcut button (in red).

8. Program analysis

Figure 8.1 is a partial schematic of the target system. It shows how the input/output (I/O) ports 0 and 1 of the microcontroller 8051 are being connected to the outside world, i.e. switches and LEDs.

8.1 Based on the program listed in Figure 7.1, identify the switches connected to P0.2, P0.3 and P0.4.

8.2 Based on the program listed in Figure 7.1, identify the LEDs connected to P1.0 and P1.7.

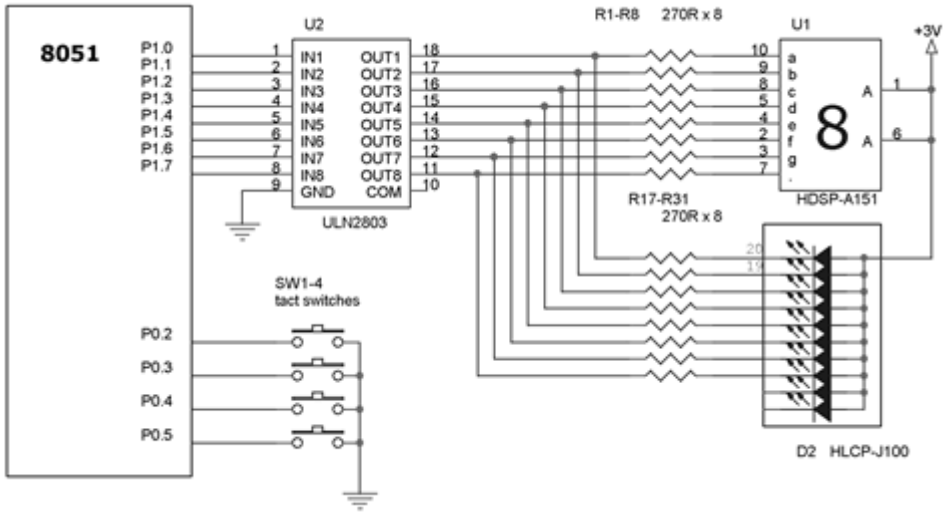

Figure 8.1

- The End -