Lab 1

Introduction to Development Tools and Lab Software

Objectives

- Learn to use software development tools such as Integrated Development Environment (IDE) (Microchip Studio), and simulator (Proteus) for the AVR ATmega328P microcontroller.

Software

- Microchip Studio (Version 7) **OR** AVR Studio (Version 4)
- Proteus ISIS (Version 8.4)

Theory

One of the learning outcomes for the lab work during this semester is for students to gain proficiency in using microcontrollers while interfacing them with a multitude of devices and connecting them to a variety of networks. Microcontrollers are "programmable" chips. Users can program these chips to do a variety of tasks. The main objective of this course would be to learn how to effective program microcontrollers make them perform most common kinds of tasks. The particular chip students will program in these labs is the AVR ATmega328p microcontroller.

Programming would require the use of Integrated Development Environments (IDEs) with cross compilers built in. Cross compilers can compile code for platforms in-addition to those they are installed on. For example, most students will either use Windows (or Apple or Linux) based computers/laptops. A cross compiler installed on Windows can be used to generate code that will run on a microcontroller. This code has to be copied/burned on to the microcontroller for it to run (Figure 1.1).

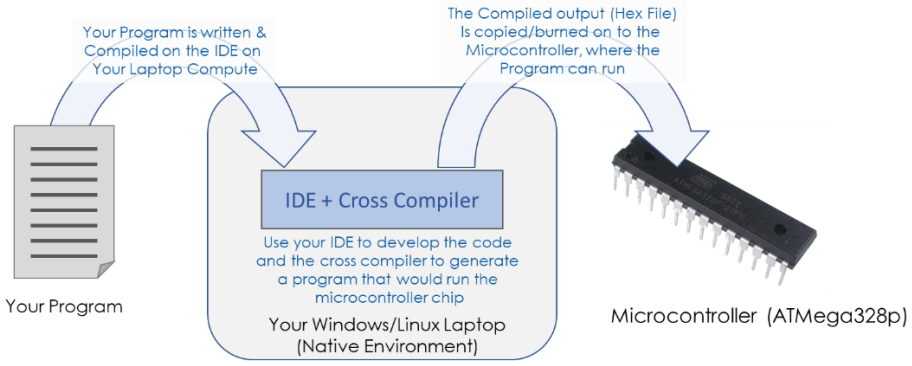

Figure 1.1 – Running cross compiled code on hardware

In case the actual hardware microcontroller is not available, the hardware can be emulated on a Modelling and Simulation software. In this software you can model

the microcontroller and then burn/write the program/hex file onto the model and simulate it on the software (Figure1.2).

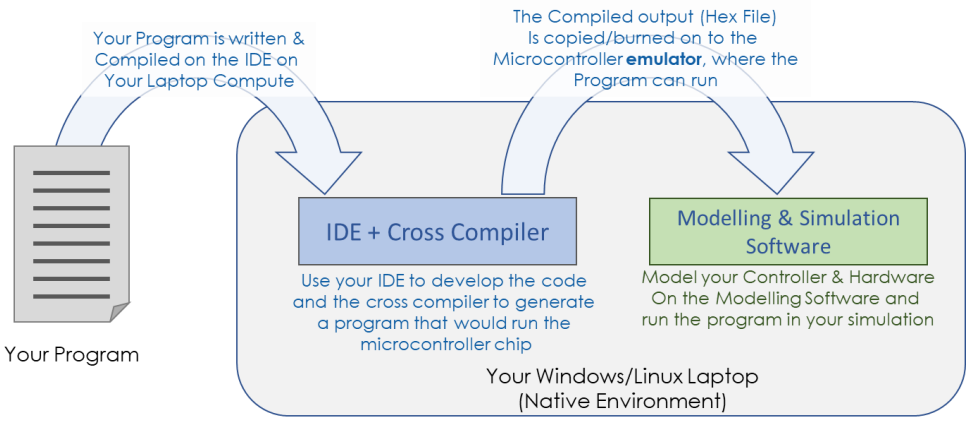

Figure 1.2 – Running cross compiled code on a simulator

In the lab we will be using either Microchip Studio or AVR Studio as the cross-compiling IDE and Proteus for modelling and simulation.

Pre-Lab Tasks

Task-1

You are strongly recommended to use your own laptop for all the laboratory tasks. In case you do not have a laptop, you will be able to use AVR Studio and Proteus on the computers provided in the MP/VLSI laboratory.

If you have a laptop, make sure that you download or acquire the following software from the MP/VLSI lab and install the following software.

Microchip Studio:

Microchip Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel SMART ARM-based and Atmel AVR microcontroller (MCU) applications. Studio 7 supports all AVR MCUs. The Microchip Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits.

Proteus

Proteus PCB design is used to design PCB and Proteus VSM is a virtual circuit simulator. Proteus VSM can simulate electric/electronic and microcontroller-based circuits. It supports number of microcontrollers available in the market.

In-Lab Tasks

Note:

As stated in the previous section, you can use either AVR studio or Microchip studio to compile and debug your code. However, it is recommended that you use Microchip Studio.

Task-1 (A) – Getting started with Microchip Studio

1. Launch Microchip Studio either by start menu or by selecting desktop icon. From start page new projects can be created and recently used projects

can be opened. The Start page can also be accessed from View → Start Page, or Alt V G. Click on new project (figure 1.3).

2. After selecting New Project, the project wizard will appear. AVR Assembler is selected for the assembly code and C/C++ is selected to compile C-code. In this tutorial, we will use C-code so select C/C++ and then select GCC C Executable Project. Write project name, select the path directory path to save files for this new project and Click ok (figure 1.4).

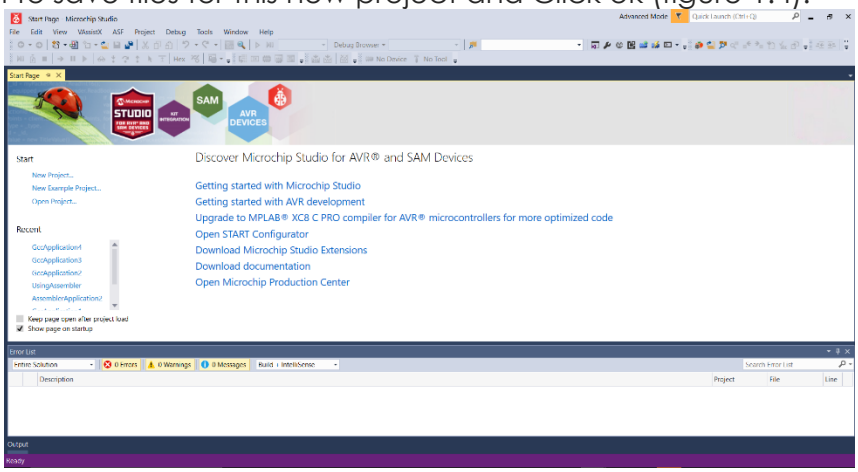

Figure 1.3 – How to create new project using Microchip Studio installed in Lab

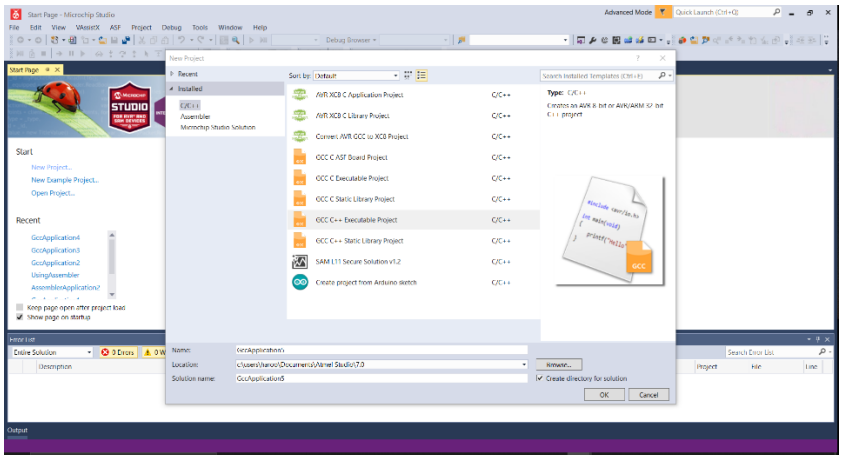

Figure 1.4 – Compiler selection and Project name description with location

3. In the Device Selection window, select ATmega328P. Click OK to create the project as shown in figure 1.5.

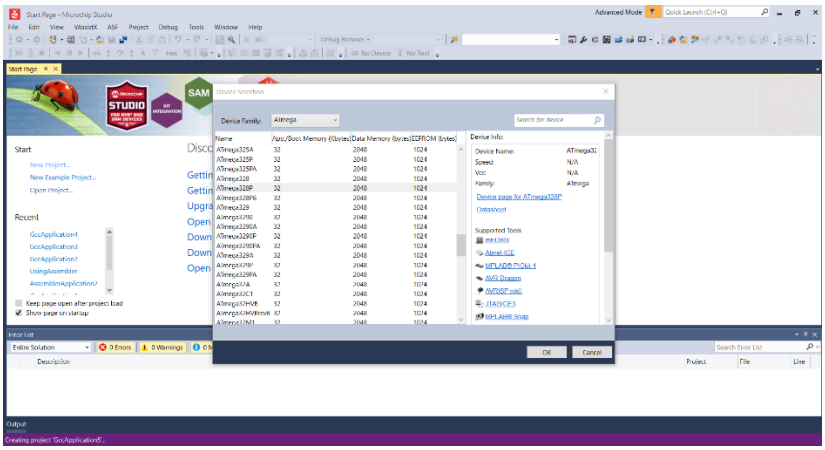

Figure 1.4 – AVR microcontroller Selection

4. Layout shown in figure 1.5 will appear on the screen. Layout will consist of source code window and message window. Write the code given below in the source window.

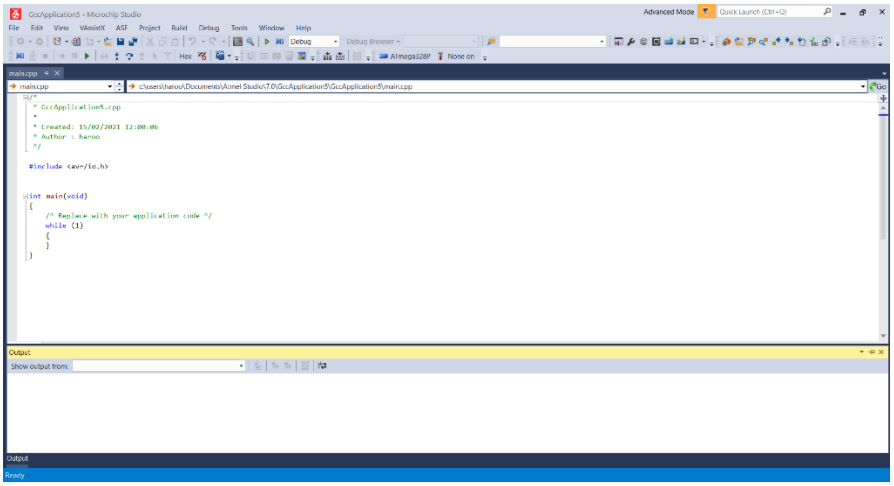

Figure 1.5 – Microchip Studio platform section wise overview

5. Press F7 button to build output files of the project including object file and hex file. Try it out with the code given below:

C-Code

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

int main(){
	// Write your code here
	DDRB = 0xFF;
	while(1){
		PORTB |= (1<<PB3);
		_delay_ms(1000);
		PORTB &= ~(1<<PB3);
		_delay_ms(1000);
	}
	return 0;
}
```

Task-1 (B) – Getting started with AVR Studio

1. Launch AVR Studio either by start menu or by selecting desktop icon. Then select Project Wizard form menu bar and select New Project as shown in figure 1.6.
2. After selecting New Project, two options will be displayed (AVR Assembler and AVR GCC). AVR Assembler is selected for the assembly code and AVR GCC is selected to compile C-code. In this tutorial, we will use C-code so select AVR GCC. Write project name (lab1 for this tutorial) and select the path directory path to save files for this new project (Select desktop and lab1 as folder name) and Click Next (figure 1.7).

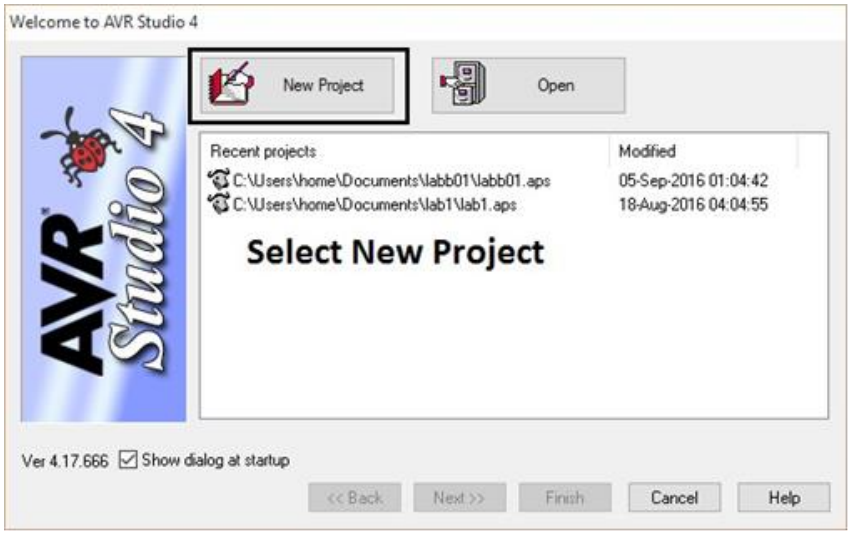

Figure 1.6 – How to create new project using AVR Studio installed in Lab

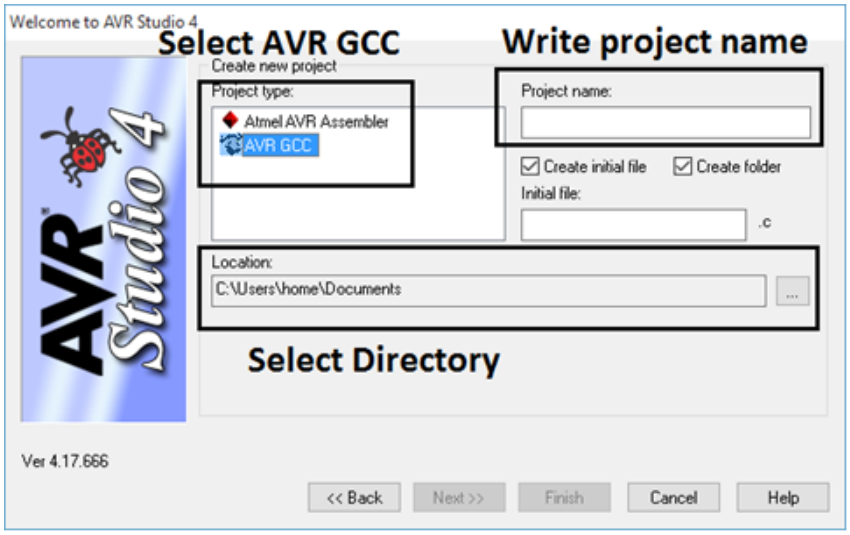

Figure 1.7 – Compiler selection and Project name description with location

3. In the next menu, select AVR Simulator in debug platform and ATmega328P in device menu. After selection click finish as shown in figure 1.8.

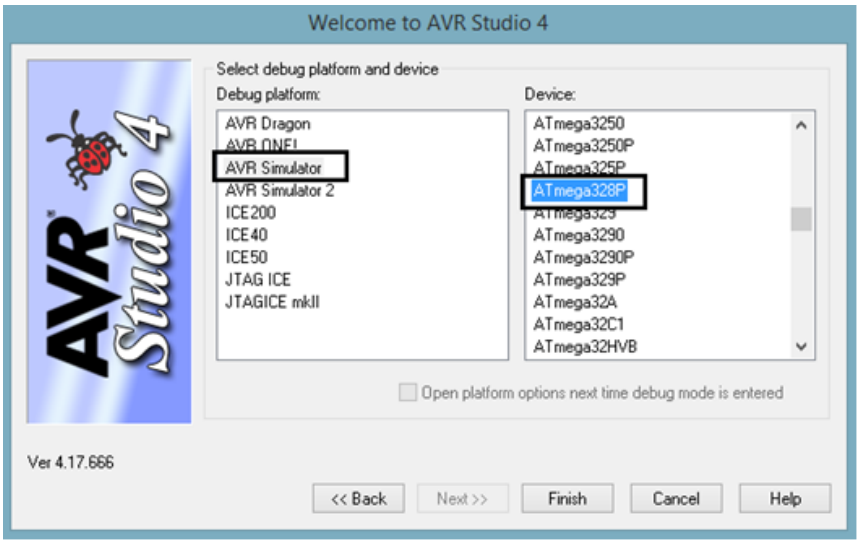

Figure 1.8 – AVR microcontroller Selection

4. Layout shown in figure 1.9 will appear on the screen. Layout will consist of source Window, I/O window, message window and I/O toolbar.

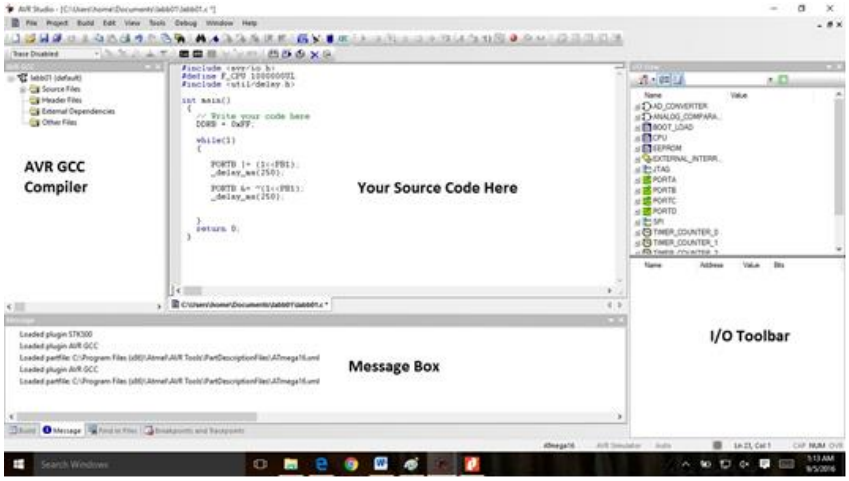

Figure 1.9 – AVR Studio platform section wise overview

5. Press F7 button to build output files of the project including object file and hex file. Try it out with the code given below:

C-Code

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

int main(){
	// Write your code here
	DDRB = 0xFF;
	while(1){
		PORTB |= (1<<PB3);
		_delay_ms(1000);
		PORTB &= ~(1<<PB3);
		_delay_ms(1000);
	}
	return 0;
}
```

Task-3 – Getting started with Proteus

To learn basics about Proteus, we will simulate a microcontroller-based circuit in Proteus. We will use the same program for microcontroller which we compiled in the Microchip Studio Tutorial.

1. Launch Proteus from start menu or by desktop icon. Proteus layout shown below will appear on the screen as shown in figure 1.10. There is a root sheet to draw circuit. The device window will initially be empty. You need to pick components from the library. Proteus component library is equipped with lots of components and ICs including microcontrollers. To pick necessary components required for a circuit, click on small P in device window.
2. After Clicking Pick devices menu as in figure 1.11 will appear on the screen. There is a keyword prompt which can refine your search. Write ATmega328P in keyword prompt and result window will display components having keyword of ATmega328P. Double click ATmega328P component in result window to add it to device list.
3. As described in step 2, add following components needed for this tutorial:

a. Animated LED

b. 470 ohms resistor

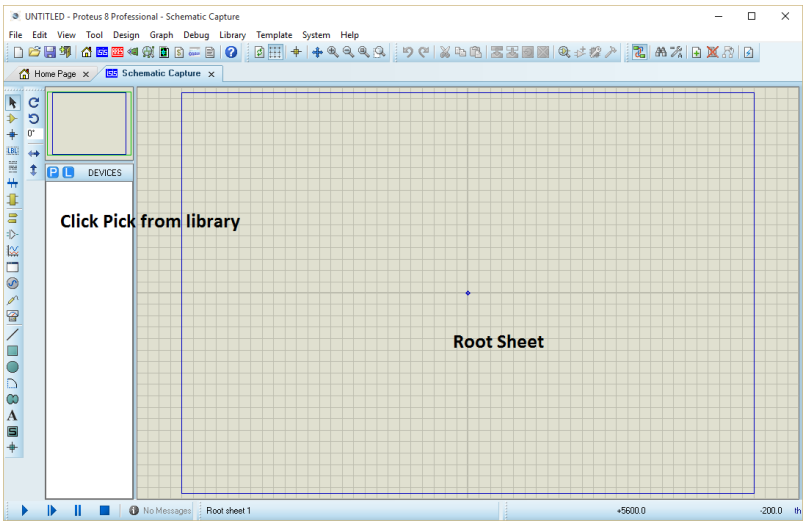

Figure 1.10 – Proteus ISIS overview of window

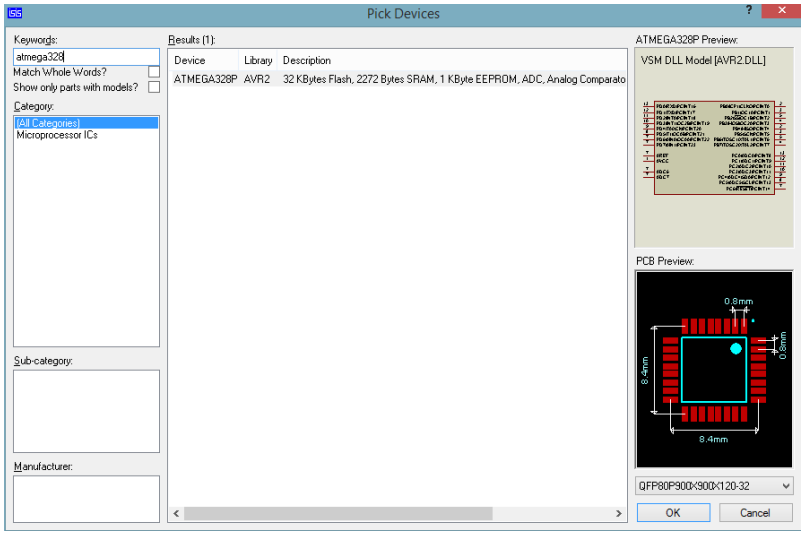

Figure 1.11 – Pick components & devices to use in the circuit

To add any component on root sheet, simply select that component from device window and click anywhere in the root sheet to place that component. To draw wire connection between two pins of any components, simply move mouse cursor to the tip of the pin and click. Then click on tip of the other pin you wanted to connect. Draw a circuit shown in figure 1.11 on Proteus root sheet by using components indicated above.

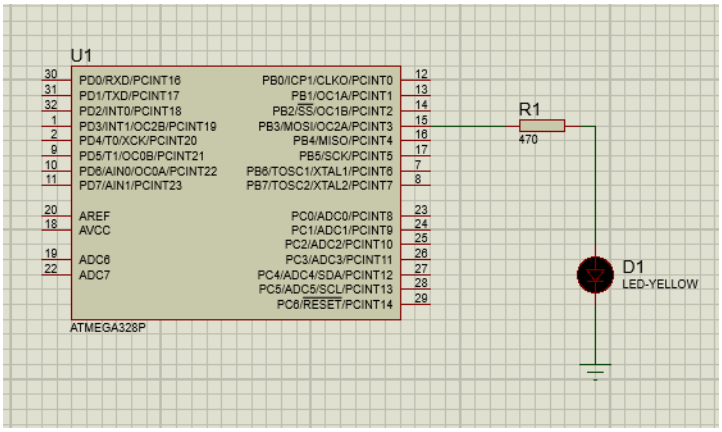

Figure 1.12 – Circuit Diagram for In-Lab Task 2

4. On the left side of Proteus layout, different modes can be selected. To add ground terminal to the circuit, click terminal mode from buttons on left side of screen and then select ground terminal. You can change characteristic properties of any component. To change, simply double click the required component. An edit component menu will appear for that particular component where you can change the characteristic properties of that component.
5. You also need to load your required program into the microcontroller memory as shown in figure 1.13. Program is loaded into microcontroller memory through hex file. Double click microcontroller edit menu will have prompt for load program. Select the hex file generated from Microchip Studio tutorial. This hex (Lab1.hex) file can be found in the Lab1 directory you created on desktop for the Microchip Studio tutorial.
6. Now run the simulation from the bottom of Root Sheet and show result to your lab instructor.

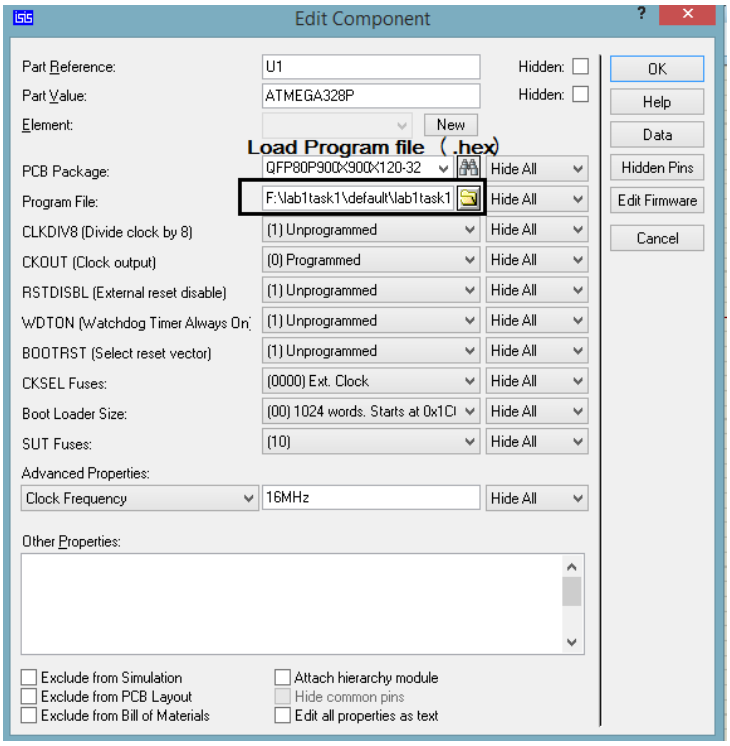

Figure 1.13 – Edit component menu

Task-3

Build the following code in Microchip Studio and simulate it on Proteus. Identify and correct the errors and warnings and rebuild the code to generate a hex file.

C-Code

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

//This program toggles all pins of
//Port D continuously with some delay
int main(){ // Write your code here
      DDRD = 0x00;
      while(1){
            PORTD = 0x00;
            _delay_ms(250);
            PORTD =0xFF;
            _delay_ms(250);
      }
      return 0;
}
```

Critical Analysis / Conclusion (To be filled in by the student)

Lab Assessment (To be filled by the lab-instructor)

Pre-Lab	/5	/25
In-Lab	/5	
Results	/5	
Viva	/5	
Critical Analysis	/5	

Comments:

Instructor Name

Instructor Signature