HIT332: Embedded and Mobile Systems
Practical 2 Notes

Shane Reynolds

February 9, 2018

Contents

1 Introduction & Background

The intention behind this brief set of notes is to provide guidance on how well the practicals and projects for HIT332: Embedded and Mobile Systems achieve their intended outcomes. There are 5 practicals in total, and 3 projects. This set of notes will cover Practical 2. The practicals (and projects) make use of a development board created by Damien Hill and Ben Saunders of Charles Darwin University. The main component of the board is the Atmel ATmega1281 16au 16MHz, 8-bit microcontroller. The development board can be seen in Figure 1.

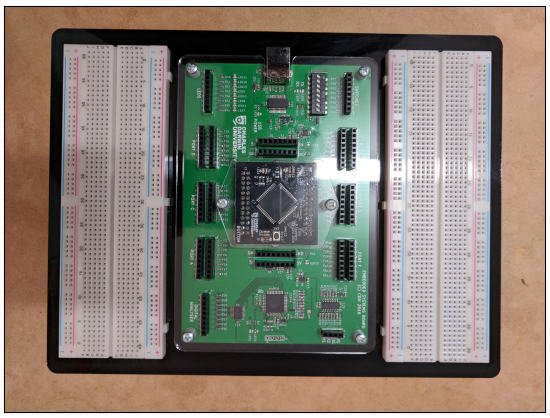

Figure 1: The development board which is used in practicals and projects for HIT332: Embedded and Mobile Systems

It must be highlighted that these notes have been developed using the board outside of its intended ecosystem. The board is made to be used on CDU's Casuarina Campus in one of the Engineering

computer labs. These labs have the appropriate software installed in the correct file paths. These notes have been written using software installed on a personal machine which CDU does not control. Furthermore, the components (other than the development board) used to complete the exercises were sourced independently from a local electronics supplier, independently of CDU. The notes will be highlighted where there has been a significant departure from the intended experience.

2 Connecting to the USB and Installing the Driver

Connecting the development board to USB installed the driver as indicated in the instructions. The only point of difference was that the COM3 port was specified for the connection, as opposed to COM15, however, this behaviour is acknowledged in the practical instructions.

3 Checking out the Hardware

The LEDs were connected to the 5V supply and worked as intended - this can be seen in Figure 2. Further, the LEDs were connected to the 5V dip switches, which also worked as intended, shown in Figures 3 and 4.

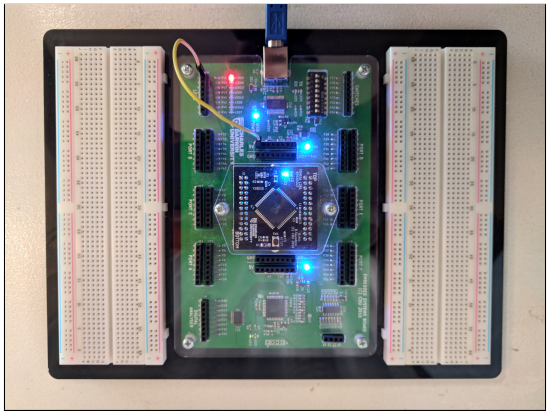

Figure 2: Five volts supplied to LED.

4 Writing the source code

The launch editor button located on the CDU Toolbox Lanucher did not work since Notepad++ was located in a different file path to the one implemented in the launcher. This was expected behaviour given the intended ecosystem is not being used. The instructions for using Notepad++ were fit for purpose, and the `main.c` developed can be seen in Figure 5.

```
1 #include <avr/io.h>
2
3 int main (void){
4 DDRD = 0xFF;
5 PORTD = 0xAA;
6 while(1){}
7 }
```

Figure 3: The implementation of the Hello, world! introductory program in `main.c`

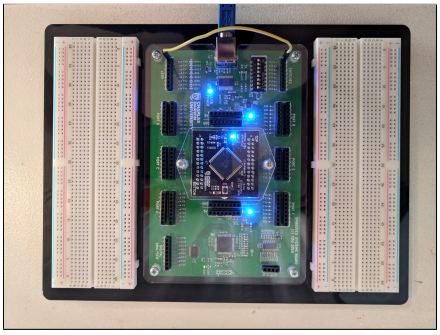

Figure 4: Dip switch in the off position, LED is off.

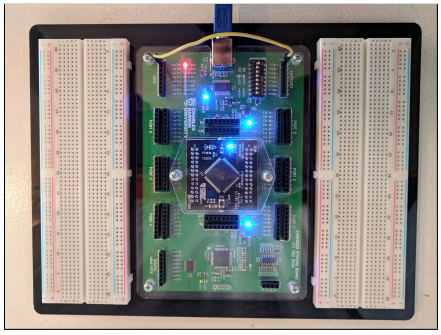

Figure 5: Dip switch in the on position, LED is on.

5 Compiling and Linking

Problems were encountered using the instructions for compiling and linking. This was due to the practical instructions assuming a specific file path for the `avr-gcc` command, and was expected. The use of the commands were tried in both the cmd environment and git bash which is a shell environment provided when installing the Windows version of git. To get around the issues encountered, the full file path to `avr-gcc` was used to access the command. This became time consuming and a short cut using an alias specified in the `.bash_profile` was created. The alias can be seen below:

```
alias avr_gcc="/c/Users/Shane\ Reynolds/Documents/CDU_embeddedSystems/CDUEmbeddedToolbox/avr_tools/bin/avr-gcc.exe"
```

This alias allowed the command to be called with the required arguments for compiling and linking.

6 Creating HEX and Programming the Processor

The same problems seen in Section 5 were encountered when trying to create hex and program the processor. To get around this more aliases were implemented in `.bash_profile`. The alias for `avr-objcopy` was:

```
alias avr_objcopy="/c/Users/Shane\ Reynolds/Documents/CDU_embeddedSystems/CDUEmbeddedToolbox/avr_tools/bin/avr-objcopymake.exe"
```

The alias for `avr-dude` was as follows:

```
alias avr_dude="/c/Users/Shane\ Reynolds/Documents/CDU_embeddedSystems/CDUEmbeddedToolbox/avr_tools/bin/avr-dude.exe"
```

Once the ATmega1281 was programmed, and the pins were connected to the required LED, the device performed as expected - seen in Figure 6.

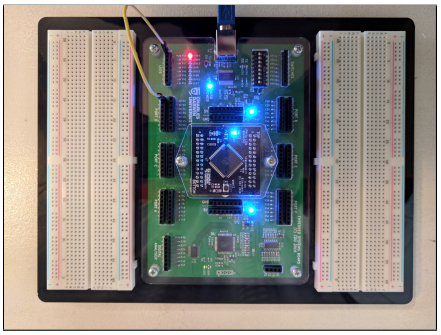

Figure 6: ATmega1281 programmed with `main.c` which places a high on pin 0 for PORTD.