ECEN301 : Embedded Systems
Lab 1 Submission

Daniel Eisen : 300447549

August 5, 2020

1 Objectives

The lab was to learn how to use Atmel Studio to develop c code for microcontroller (specifically the 8051), developing an understanding of how to control sending data in and out to the pins/ports and using a series of peripheral inputs. This is meant to be the basis for future labs.

2 Methodology

2.1 Introduction

To begin with, setting up a connection to, and confirming a c programming connection. Initially there was a problem with this, flashing that is, was Atmel was unable to automatically use BATCHISP to program after build. The solution was to "manually" connect and go through the programming steps using FLIP; establishing a COM port connection, loading hex file, and flashing the 8051.

2.1.1 Using ports to output data

The first task was basic input and output to and from the ports of the μC. This was done initially by loading a count.c program to display to the LED in the IO module. I was then observed that with no delay it was impossible to read without, for example slowing a recording down etc, so a delay function was used. Basically a non useful for loop.
This did however have a limit do to overflowing the passed 16 bit integer above 32768, so multiple calls are required for more delay. This could be improved by unsung a timer based delay.
Next was to toggle the output between 0b00001111 and 0b11110000. Again using a delay for human readability. This output was connected to the IO module to display on the LED.
Finally, to establish human input and μC output, two IO modules were used. One in EXT mode to receive input from P2 and one in INT mode to send data to P1. The software then connected the dataflow between these 2 ports so the input switches of one module controlled the LED output of the other.

2.1.2 Keyboard scanning

For more complex input, the KEYPAD module consisting of a 4x3 standard key panel input is used. This required the use of a provided library to sample the input byte from the keypad to be able to use that data within the program. This is where the first, and really only problem arose. I just could not get the provide precompiled .lib file working with Atmel Studio, never worked out what was wrong so had to end up using the .c and .h as just a part of the project. Hardware wise it was straight forward with the library outlining that the keypad be connected to P2, the IO module was then wire to P1 for output to the LEDs. The software called the sample function and writes to the IO module. Due to the nature of the sample function, a sequential if else ladder of value checks, multiple key inputs yielded unintended outputs, but otherwise operation was as expected and simple.

```
#include "AT89C51AC3.h"
#include "ECEN301LibSDCC.h"

void main()
{
        unsigned char key;
        while (1){
                key = sampleKeypad();
                P1 = key;
        }
}
```

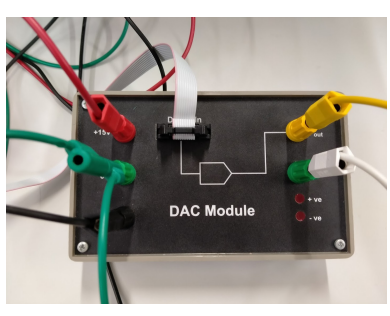
2.1.3 Digital to Analogue Conversion (DAC)

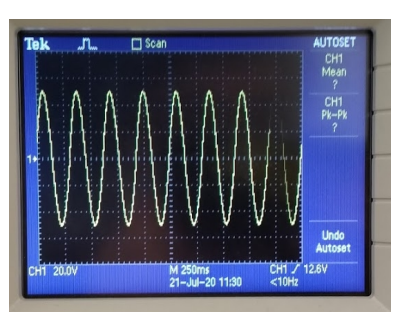
For the operation of external analogue peripherals, a digital signal from the controller must be converted to a voltage signal with a DAC. To demonstrate this a sine generate was written. Cantering the output values of the function around the proper point was most difficult and required quite the fiddling to get right.

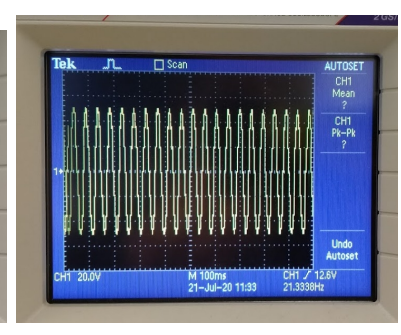
```
#include "AT89C51AC3.h"
#include <math.h>

#define STEP 2048.0

void main(void)
{
        /* Replace with your application code */
        while (1){
                for (int i = 0; i < STEP; ++i){
                        int val = 127.0 * sinf(((2*PI)/STEP)*i) + 128.0;
                        P1 = val;
                }
        }
}
```







2.1.4 DAC Motor Control

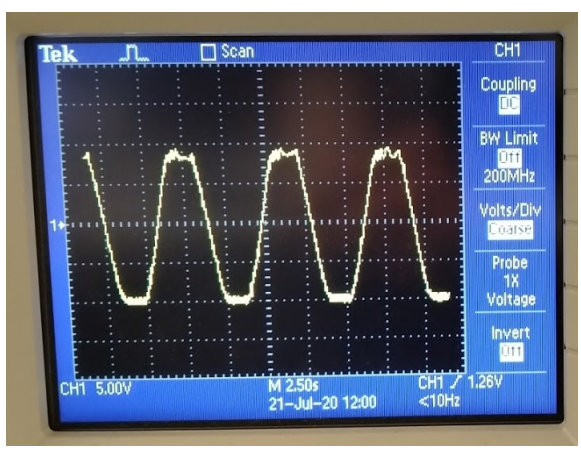
By utilising bit masking to separate the 8bit word of IO switch's into a speed control word (7bit) and a 1bit direction variable the motor speed and direction could be independently controlled. One issue how ever that the speeds in each direction must re inverse of each other as seen in the code.

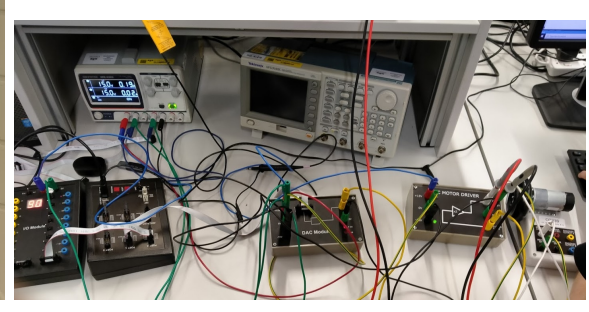
```
#include "AT89C51AC3.h"
void main(void)
{

        while (1) {
                char con = P2;
                char vel = con & 0b01111111; //mask speed value 0-127

                if (con & 0b10000000){
                        P1 = 128+vel;
                } else {
                        P1 = 128-vel;
                }
        }
}

//a,b) forward (469mV) @ 4, back (-407mV) @ 5
//c) 5V drives a 8.8V
```





3 Questions

1. The Object and executable files and object files are generated on compilation. The executable binary is the machine code that will be read and executed the controllers CPU and not really readable or understandable to us. The object file is not executable but is used only in compilation during the linking stage. The rel file is a generated object file from the assembler per src file, this is feed into the linage editor.

2. Port 0-3 can be fully used, with all 8-bits accessible. But P4.0 and P4.1 are the data lines of the UART connection the computer for programming, the switch (dis)connects these pins from the USB interface.

3. By default, unpressed the keypad output was 0c0b

4. When multiple keys would be pressed, the sampleKeypad function would output the key that was pressed and closer to the last value it checked against in the function.

5. The default should not be a valid value (unpressed), only allow for a single key pressed, have some knowledge of a multiple cycle press.

6. Hardware setup is shown is section 2.1.3

7. (a) forward was 469mV at switch position decimal 4
   (b) backwards was 407mV at switch position decimal 5
   (c)