

## Objectives

The primary objectives of this lab was to become acquainted with the Atmel Studio development environment and also the AT89C51AC3 chip and development board. This lab would be divided into four different parts, which included I/O module, Keypad, DAC converter, DAC motor driver and 12V DC motor. These are peripherals which have been constructed that can be connected to the development board.

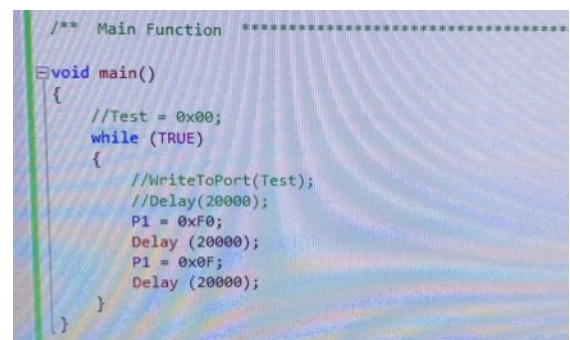
## Methodology

### Introduction

In this part, the code was provided, so I just needed to connect microcontroller with PC by USB port, and also connect it with I/O module. Connecting up the power supply and USB, and then connected Port 1 to digital in on I/O. And also used 15V power supply to supply I/O module. I initially had an issue when connecting the RS232 serial port with port, as FLIP give me an attention of timeout error. Tutor told me that I needed to press PROGRAM button, then press the RESET, and then release RESET, at last release PROGRAM. Because the microcontroller needed to be reset before connected to PC.

### Using ports to output data

The first program written in this section was a counter. Writing a program to set 4 port bits hi, wait for a length of time, then make those 4 bits lo, and set the remaining 4 bits hi. Have this repeating indefinitely. To achieve this, a main function shown in figure 1:  
P1 = 0xF0 means set first 4 bits hi, the other 4 bits lo.  
P2 = 0x0F means set first 4 bits lo, the other 4 bits hi.



```
/** Main Function *****  
  
void main()  
{  
    //Test = 0x00;  
    while (TRUE)  
    {  
        //WriteToPort(Test);  
        //Delay(20000);  
        P1 = 0xF0;  
        Delay (20000);  
        P1 = 0x0F;  
        Delay (20000);  
    }  
}
```

Fig. 1

The initial code provided the delay was too short, causing the display to be difficult to follow. This issue was corrected by increasing the delay from 50 to 20000.

Next, identify which ports are available to us as output.

As shown in the previous program, port 1 worked. In order to test the other ports, changing the code "P1" to "P0", "P2", "P3" and "P4" respectively. As result, P0, P2 and P3 could display the same output with P1. However, P4 was slightly different with the other ports. Only bits 0 to 4 could be used as outputs.

## Keypad Scanning

In this program, plug a numeric keypad into port 2, and an I/O module to receive the output on port 1. The code was given, but the value of P1 was missing. So I simply assign  $P1 = DepressedKey$ , and compiled the code, then tested the keypad. By pressed the number keys, the corresponding numbers would display in I/O. However, there were two different keys—\* and #. When pressed \*, displayed 0A; pressed #, displayed 0C.

## Digital to Analogue Conversion (DAC)

The first session of this program is to produce a sine wave and display on oscilloscope. The main code shown in figure.2.

Firstly, included math.h file because sin function was in there. Then, set step as 2048. After that, used a for loop in main() and assign the value to P1. (Because microcontroller has 8 bits, 00000000 = 0; 11111111 = 255) Assumed that there were 128 numbers for positive half sine wave, and the other 128 numbers for negative half sine.

For hardware setting, firstly connected up microcontroller to PC, then connected DAC module to Port 1. Using 15V power supply as DAC module power. Finally, connect oscilloscope to DAC Vout. The setting shown in figure 3.

```
/** Include Files *****/
#include "AT89C51AC3.h"
#include <math.h>

/** Main Function *****/
#define STEP 2048.0

void main()
{
    while (1)
    {
        for(int i =0; i<STEP;i++){
            int val = 127.0 * sinf(((2*pi)/STEP)*i) + 128.0;
            P1 = val;
        }
    }
}
```

Fig. 2

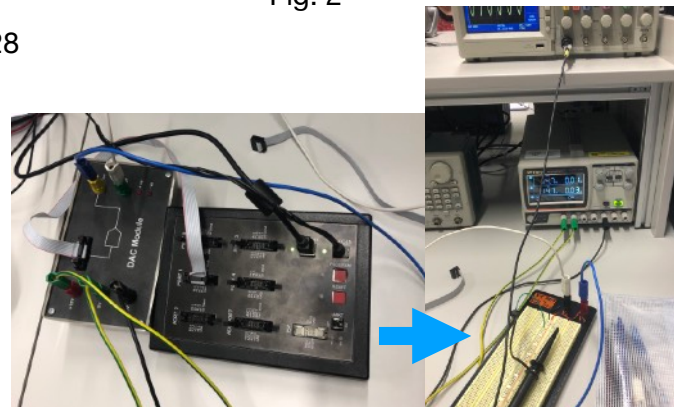


Fig. 3

The result display on the oscilloscope shown in figure. 4.

The second session of the program was to increase the frequency of the output. By simply changing  $i++$  into  $i=i+3$  (for example). Then sine wave was produced faster and with more period, which could be seen in figure 5.

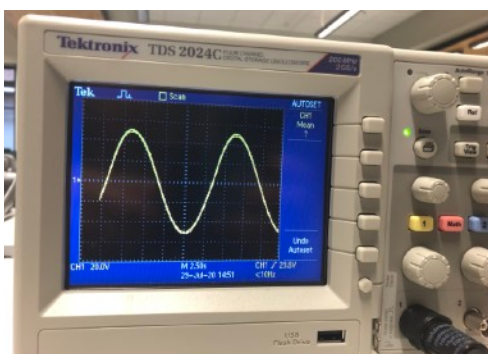


Fig. 4

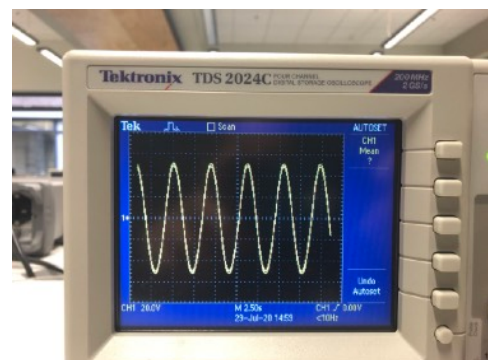


Fig. 5

## DAC Motor Control

The first session of the program was based on the previous lab session (DAC), set up an extra motor driver and a 12V DC motor. The connection shown in figure 6.

Since the wires were messy, in order to make connection clear, the schematic is provided in figure 7.

Then downloaded the code from the previous project and see how would the motor move. I initially had an issue when the program was running, the motor always ran forward, and the DAC always showed +Vout. No matter how hard I checked the code and connection, the issue still could not be found. Finally, when replaced a new DAC module, it worked. The DAC was broken.

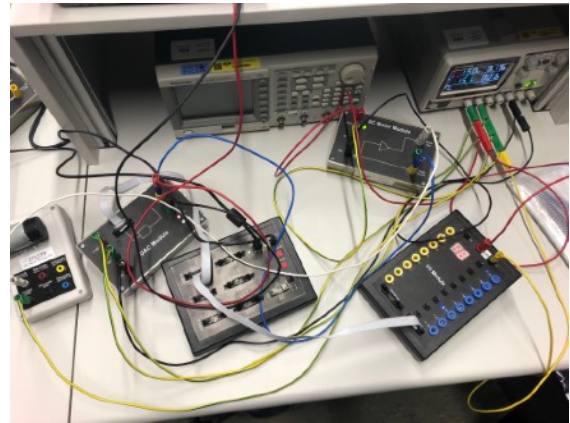


Fig. 6

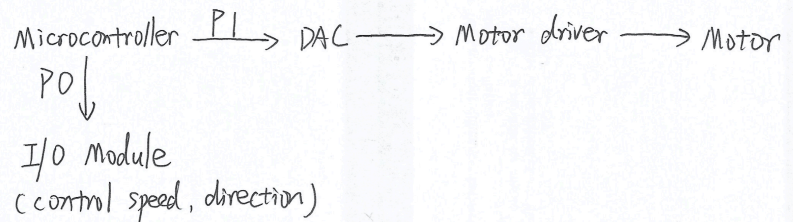


Fig. 7

As a result, the motor ran forward when the voltage was positive. Then it ran backward when the voltage was negative. And it repeated this infinitely. One interesting finding was that the motor had larger resistance when it ran backward than it ran forward.

The second session was to control the direction and the speed of the motor by using the I/O module. The program code is shown in figure 8.

Firstly, set bit 7 to 0, because it would be used to indicate direction. Then, used an if function to decide to go forward or backward. The code actually ran in an opposite function. When bit 7 on, the motor ran backward, when bit 7 off, ran reverse. By simply changing 0b10000000 in the if condition to 0b00000000, the motor can run in the other way.

```
#include "AT89C51AC3.h"

void main(void)
{
    /* Replace with your application code */
    while (1)
    {
        char con = P0;
        char vel = con & 0b01111111; //set bit7 to 0

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

Fig. 8

The minimum voltage required to get the motor moving in the forward direction was +0.17V. Switch setting shown in figure 9.

The threshold voltage required to get the motor moving in the backward direction was -0.18V. Switch setting shown in figure 10.



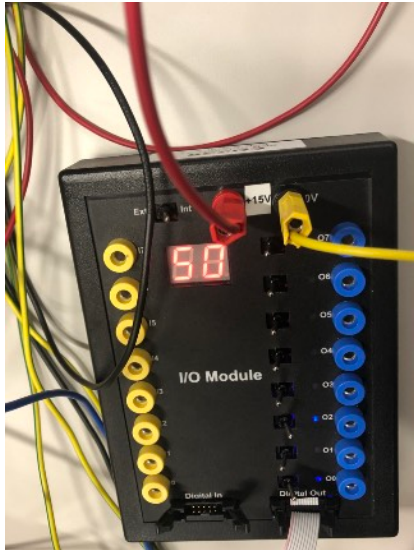


Fig. 9

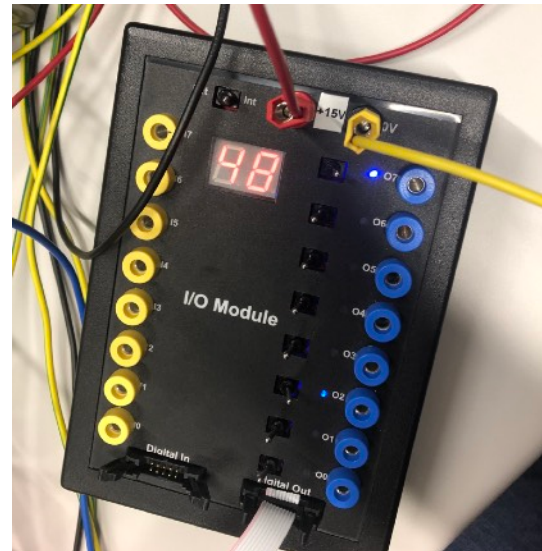


Fig. 10

In figure 9, switch on bit 2 and 0, remain the rest of bits off. Switch off bit 7, ran backward.

In figure 10, switch on bit 2, remain the rest of bits off. Switch on bit 7, ran forward.

The maximum obtainable DAC output voltage was 4.94V, the motor ran vary fast.

The minimum obtainable DAC out voltage was 19mV, the motor was very slow, nearly stop.

## Questions

**Q1:** object and executable file are created by the compiler (here is Atmel Studio), according to the main.c file. They are the file contain machine code, which is designed to be translated by the machine instead of people. The purpose of the object file is to connect a relocatable machine code in to executable file.

**Q2:** P0, P1, P2 and P3 can be fully used. P4 can be used for only bit 0-4. No port will affect the UART. If the UART switch off, microcontroller can not be connected into the PC.

### Keypad scanning

**Q1:** when there was no key depressed, the program would display 0d.

**Q2:** when multiple keys are pressed at the same time, bigger number would be displayed. For example, if press 1 and 2 at the same time, 2 will be displayed. No matter which one is pressed first.

**Q3:** issue is: when \* was pressed, display 0A; when # was pressed, display 0C.

**Q5:** answers please refer to the previous part of this report. (Digital to Analogue Conversion (DAC))

**Q6:** when motor ran forward, DAC output voltage was 4.92V at peak. Then drop gradually until 0V, the motor stop. Next, start to ran reverse, DAC output voltage was -4.89 at peak. Then, again drop to 0V, and started to do forward direction. Repeating this infinitely.