
Data Acquisition and Interfacing to Microprocessor

Tanvir Al Amin Popel

Student, Department of CSE, BUET

tanviramin@gmail.com

Table of Contents

This report contains our study about Data Acquisition systems and details of our idea and implementation of the low cost solution.

We would like to thank almighty that the project could be finished in time.

We would like to thank our respected teachers for their suggestion, support and constant help during the project.

Thanks to Digital Lab and personnel there for helping us during the project.

Thanks to all other persons and institutions, who helped us in various ways and

Finally we would like to thank our parents, family and friends for their support.

1 Introduction	3
2 Data Acquisition Systems	4
3 Project Idea	10
4 Component Specification	11
5 Circuit Diagram	13
6 Software Specification	14
7 Testing	15
8 Performance Analysis	17
9 Walkthrough	18
10 Future Works	19
Reference	19
Appendix A – Source Code	20

Signals contain information about a variety of things and activities in our physical world. Examples abound: Information about the weather is contained in signals that represent the air temperature, pressure, wind speed etc. The voice into a microphone provides an acoustic signal. To monitor the status of a nuclear reactor, instruments are used to measure a multitude of relevant parameters, each instrument producing a signal.

Signals from a transducer are generally analog. They can be described in terms of frequency spectrum.

To get information from a signal, we must process it. We may use the analog signal from a transducer and amplify it and process it. Or we can also use digital techniques. By using microprocessors or microcontroller-based technology we can offer more and more sophisticated processing. Moreover, if we want to automate a system that needs to use signals from various sources and then take some decision, we must convert it to digital. Then we can easily save the data, display it or perform any operation using it.

In this report we discuss the technical issues of a microprocessor-based data acquisition system which at first converts the analog data to digital and then interfaces it to a microprocessor.

The report is organized as follows:

Section 2 briefly describes data acquisition systems and some of their technical issues.

Section 3 introduces the idea of a project for a data acquisition system via digital techniques.

Section 4 lists the components we used.

Section 5 describes the circuit.

Section 6 describes the software operation and handling of data.

Section 7 describes the testing phase of the project.

Section 8 is an evaluation of the project. We have done a performance analysis on it.

Section 9 is a discussion about our future thinking about this project.

Section 10 is a walkthrough.

2.1 State of the Art

Digital Data acquisition means gathering signals from sources and digitizing the signal for storage, analysis, and presentation on a personal computer (PC). There are five components to be considered :

- 1) Transducers and sensors
- 2) Signals
- 3) Signal conditioning portions
- 4) DAQ hardware
- 5) Software

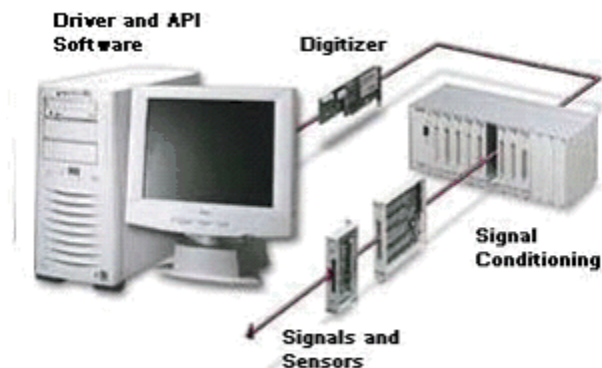


Figure 2.1 Data Acquisition System

2.1.1 Transducers

A transducer converts a physical phenomenon to a measurable electrical signal, such as voltage or current. Transducers are synonymous with sensors in DAQ systems. There are specific transducers for many different applications, such as measuring temperature, pressure, or fluid flow.

2.1.2 Signals

The appropriate transducer converts the physical phenomena into measurable signals. However, different signals need to be measured in different ways. For this reason, it is important to understand the different types of signals and their corresponding attributes.

Signals can be categorized into two groups:

Analog Signals

An analog signal can be at any value with respect to time. A few examples of analog signals include voltage, temperature, pressure, sound, and load. The three primary characteristics of an analog signal include level, shape, and frequency (Figure 2.2).

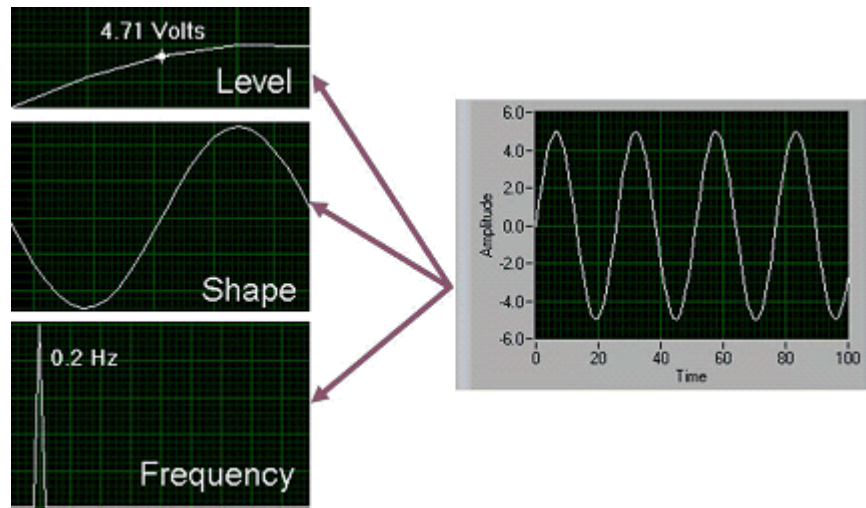


Figure 2.2. Primary Characteristics of an Analog Signal

Level

Analog signals can take on any value, the level gives information about the measured analog signal. The intensity of a light source, the temperature in a room, and the pressure inside a chamber are all examples that demonstrate the importance of the level of a signal. When measuring the level of a signal, the signal generally does not change quickly with respect to time. The accuracy of the measurement, however, is very important.

Shape

Several wave shapes exist - sine, square, saw tooth, and triangle.. Signals where shape is of interest generally change rapidly with respect to time, but system accuracy is still important. The analysis of heartbeats, video signals, sounds, vibrations, and circuit responses are some applications involving shape measurements.

Frequency

All analog signals can be categorized by their frequency. Unlike the level or shape of the signal, frequency cannot be directly measured. The signal must be analyzed using software to determine the frequency information. This analysis is usually done using Fourier Transform.

When frequency is the most important piece of information, it is important to consider include both accuracy and acquisition speed. Although the acquisition speed for

acquiring the frequency of a signal is less than the speed required for obtaining the shape of a signal, the signal must still be acquired fast enough that the pertinent information is not lost while the analog signal is being acquired. The condition that stipulates this speed is known as the Nyquist Sampling Theorem. Speech analysis, telecommunication, and earthquake analysis are some examples of common applications where the frequency of the signal must be known.

Digital Signals

A digital signal cannot take on any value with respect to time. Instead, a digital signal has two possible levels: high and low. Digital signals generally conform to certain specifications that define characteristics of the signal. The useful information that can be measured from a digital signal includes the state and the rate (Figure 2.3).

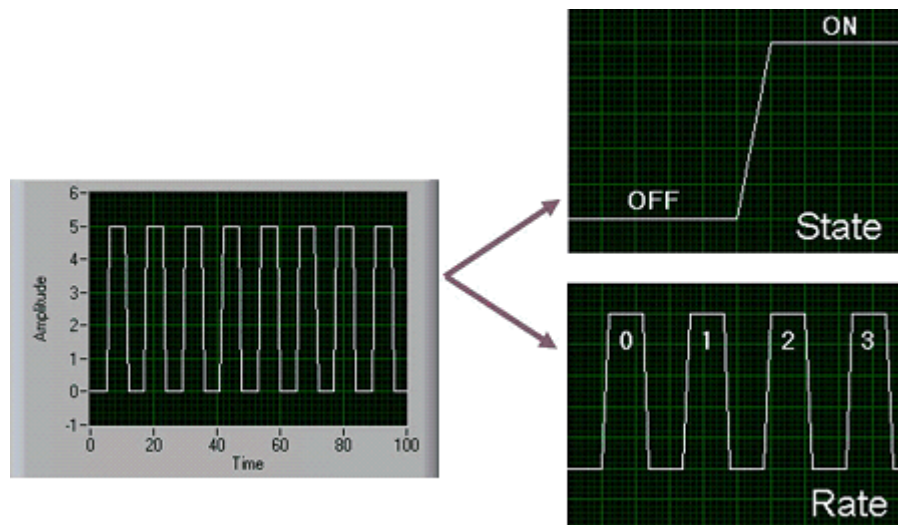


Figure 2.3. Primary Characteristics of a Digital Signal

State

Digital signals cannot take on any value with respect to time. The state of a digital signal is essentially the level of the signal – on or off, high or low. Monitoring the state of a switch – open or closed – is a common application showing the importance of knowing the state of a digital signal.

Rate

The rate of a digital signal defines how the digital signal changes state with respect to time. An example of measuring the rate of a digital signal includes determining how fast a motor shaft spins. Unlike frequency, the rate of a digital signal measures how often a portion of a signal occurs. A software algorithm is not required to determine the rate of a signal.

2.1.3 Signal Conditioning Portions

Sometimes transducers generate signals too difficult or too dangerous to measure directly with a DAQ device. For instance, when dealing with high voltages, noisy environments, extreme high and low signals, or simultaneous signal measurement, signal conditioning is essential for an effective DAQ system. Signal conditioning maximizes the accuracy of a system, allow sensors to operate properly, and guarantees safety.

- Amplification
- Attenuation
- Isolation
- Bridge completion
- Simultaneous sampling
- Sensor excitation
- Multiplexing

2.1.4 DAQ Hardware

The DAQ hardware acts as the interface between the computer and the outside world. It primarily functions as a device that digitizes incoming analog signals so that the computer can interpret them.

2.1.5 Software

Software transforms the PC and the DAQ hardware into a complete data acquisition, analysis, and presentation tool. Without software to control or drive the hardware, the DAQ device will not work properly. Driver software is the layer of software that allows easy communication to the hardware. It forms the middle layer between the application software and the hardware.

2.2 Interfacing via Parallel Port

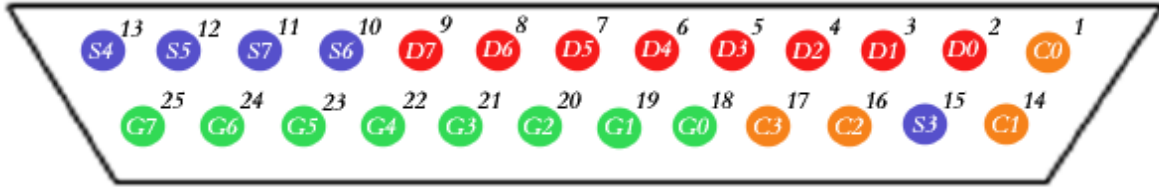


Fig 2.4: Parallel Port

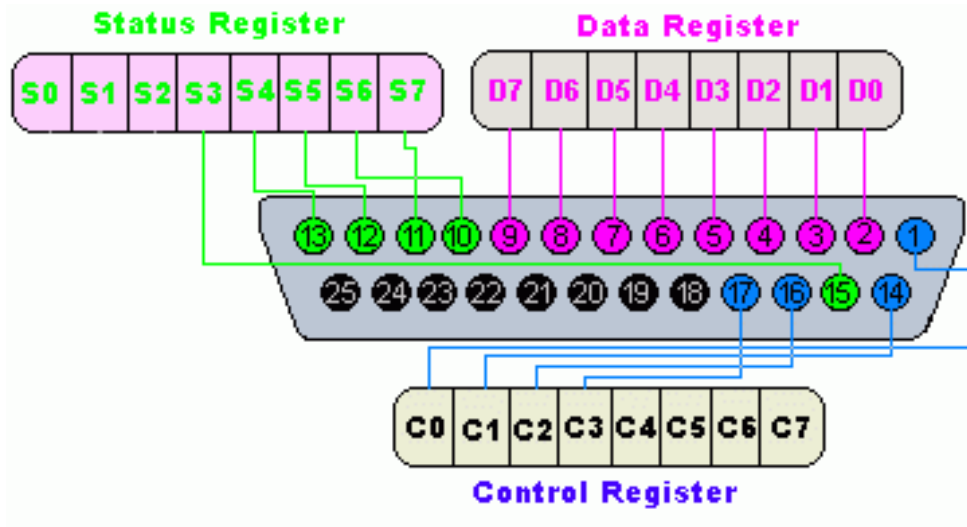
The lines in DB25 connector are divided in to three groups, they are (Figure 2.4)

- 1) Data lines (data bus) (RED)
- 2) Control lines (ORANGE)
- 3) Status lines (BLUE)

Pin No	Signal name	Direction	Register – bit	Inverted
1	nStrobe	Out	Control-0	Yes
2	Data0	In/Out	Data-0	No
3	Data1	In/Out	Data-1	No
4	Data2	In/Out	Data-2	No
5	Data3	In/Out	Data-3	No
6	Data4	In/Out	Data-4	No
7	Data5	In/Out	Data-5	No
8	Data6	In/Out	Data-6	No
9	Data7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper-Out	In	Status-5	No
13	Select	In	Status-4	No
14	Linefeed	Out	Control-1	Yes
15	nError	In	Status-3	No
16	nInitialize	Out	Control-2	No
17	nSelect-Printer	Out	Control-3	Yes
18-25	Ground	-	-	-

The registers found in standard parallel port are ,

- 1) data register
- 2) Status register
- 3) Control register



In an IBM PC, these registers are IO mapped and will have unique address.

Register	LPT1	LPT2
data register(baseaddress + 0)	0x378	0x278
status register (baseaddress + 1)	0x379	0x279
control register (baseaddress + 2)	0x37a	0x27a

Our Idea of a Data Acquisition System:

We plan about a project for Data Acquisition. Commercially high precision Analog to Digital Converter Chips are available. We use ADC 0804 LCN to convert analog signal to digital. ADC The ADC080X family are CMOS 8-Bit, successive approximation A/D converters which use a modified potentiometric ladder and are designed to operate with the 8080A control bus via three-state outputs. These converters appear to the processor as memory locations or I/O ports, and hence no interfacing logic is required. The differential analog voltage input has good common mode-rejection and permits offsetting the analog zero-input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

After converting the analog signal to digital we buffer it, and the digital signal is then fed to parallel port.

For data input in parallel port we can use status and control pins.

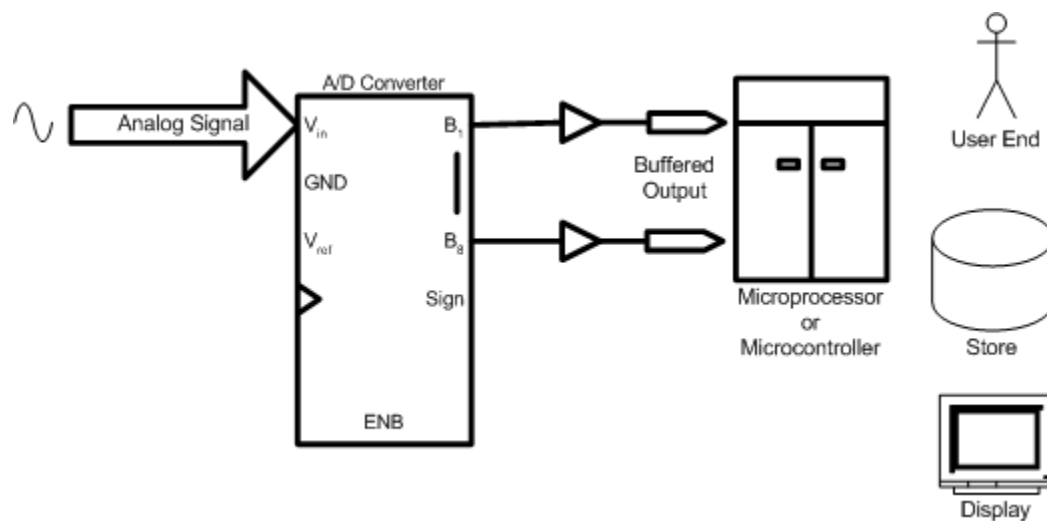


Fig 3.1 Idea of Our Project

For upper 4 bit nibble we can use status bits and for lower 4 bit nibble we can use control bits. In together 1 byte of data is fed each time.

In software part, we use Windows 98 operating system and Turbo C for the application program Turbo C has some built in functions for reading and writing to ports.

4.1 Major Components

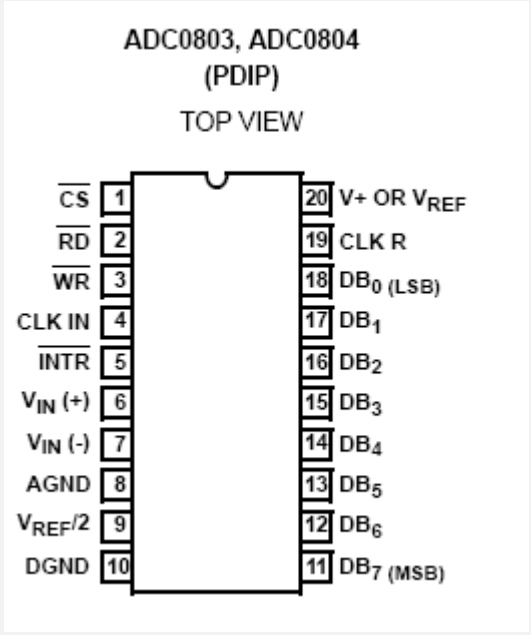
Name	Quantity	Comment
ADC 0804 LCN	1	Analog to Digital Converter CMOS IC. Must be handled with care  <p style="text-align: center;">ADC0803, ADC0804 (PDIP) TOP VIEW</p>
74 LS 373	1	TTL Low power Schottky 8 Bit Transparent Latch with Input / Output enable.
Parallel port adapter	1	Used for connectivity to parallel port. Generally attached with motherboard of a PC

Fig 4.1 : ADC 0804 Pinout

4.2 Discrete Components

Name	Quantity	Comment
10 K Resistance	3	1 Resistance used for internal pulse generator circuit. Rest 2 used for Obtaining +2.5V voltage.
10 uF capacitor	1	Power Supply spike capacitor
150 pF Capacitor	1	Pulse generator Circuit capacitor

4.3 Auxiliary Systems

Name	Quantity	Comment
MALE 25 DB Connector	1	Used for connecting the output to parallel port
Bread Board	1	Used for making the circuit.
+5 V Power Supply	1	Power supply for project
+2.5 Volt Power Supply	1	Auxiliary level shifter for input.

5 Circuit Diagram

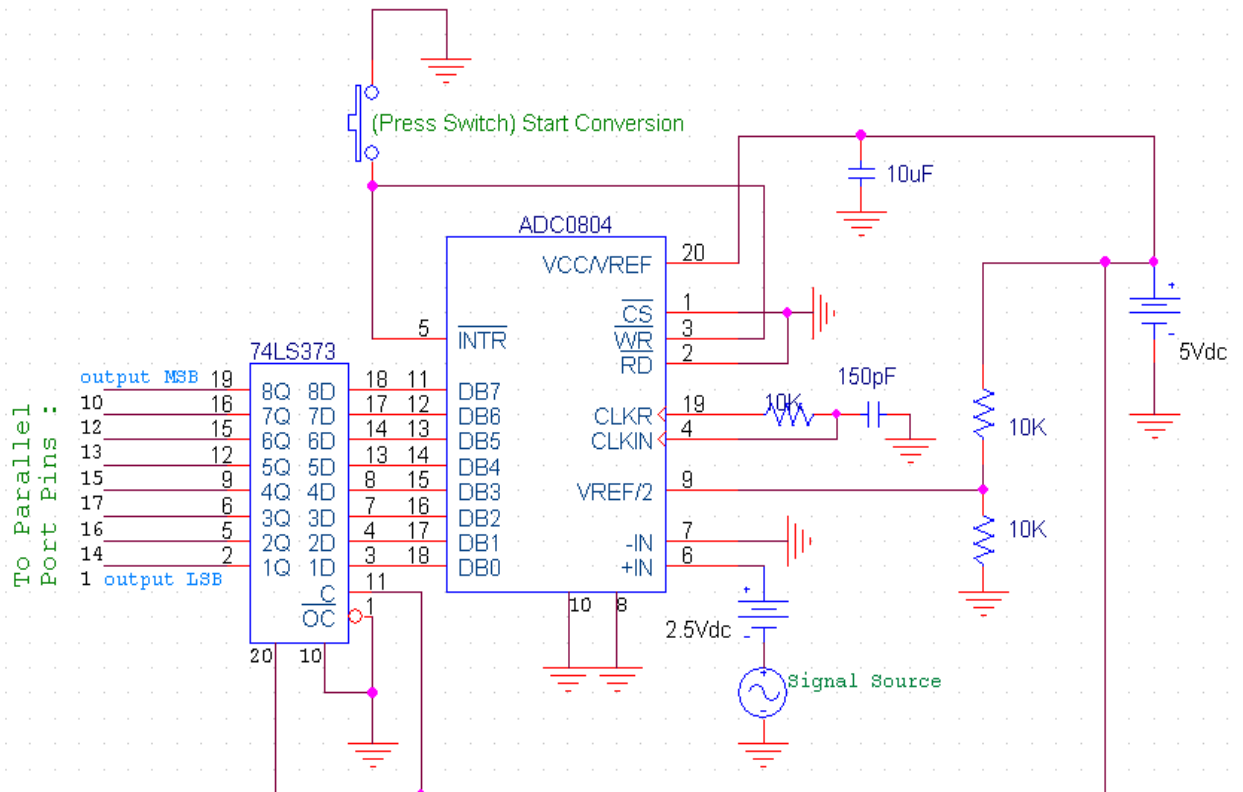


Fig 5.1 : The Total Circuit Diagram

This is the complete circuit diagram for the project. The ADC operates in Free Running Mode. Output of the Latch is Connected to Parallel Port pins 10,12,13,15,17,16,14,1 respectively from MSB to LSB. 10,12,13,15 are status pins and the rest 4 are control pins.

This circuit diagram is drawn with PSpice Capture A/D included with Orcad Family Release 9.2

6.1 Software to Read from Port

For reading 1 byte we need to read both status port and control port. From status port we get Upper Nibble of a byte. And from Control port we get the lower Nibble.

For the status port we use S6(pin 10), S5(pin12), S4(pin 13), S3(pin 15). S6 being MSB. So we use a mask = 001111000 (bin) = 120 (dec).

Right shifting the masked value to 1 bit Left makes it the upper nibble of our result.

For control port we use C3(pin 17), C2(pin 16), C1(pin 14), C0(pin 1), where C3, C1, and C0 are active Low. As they are tri stated before taking data we at first make all these high, by sending 0x4 to 0x37A port.

After reading from Control port, we use a mask 0xF (=1111 in bin). We then XOR this value by 11(dec) = 1011 (bin). As C3,C1 and C0 are inverted and C2 is non-inverted this value is chosen to find the actual value.

```
int readPort ()
{
    int status;
    int control;
    int result_byte= 0;
    outportb(0x378+2, 0x4);
    status = inportb(0x378+1);
    control = inportb(0x378+2);
    status &= 120;
    control &= 15;
    status <<= 1;
    control ^= 11;
    result_byte |= status;
    result_byte |= control;

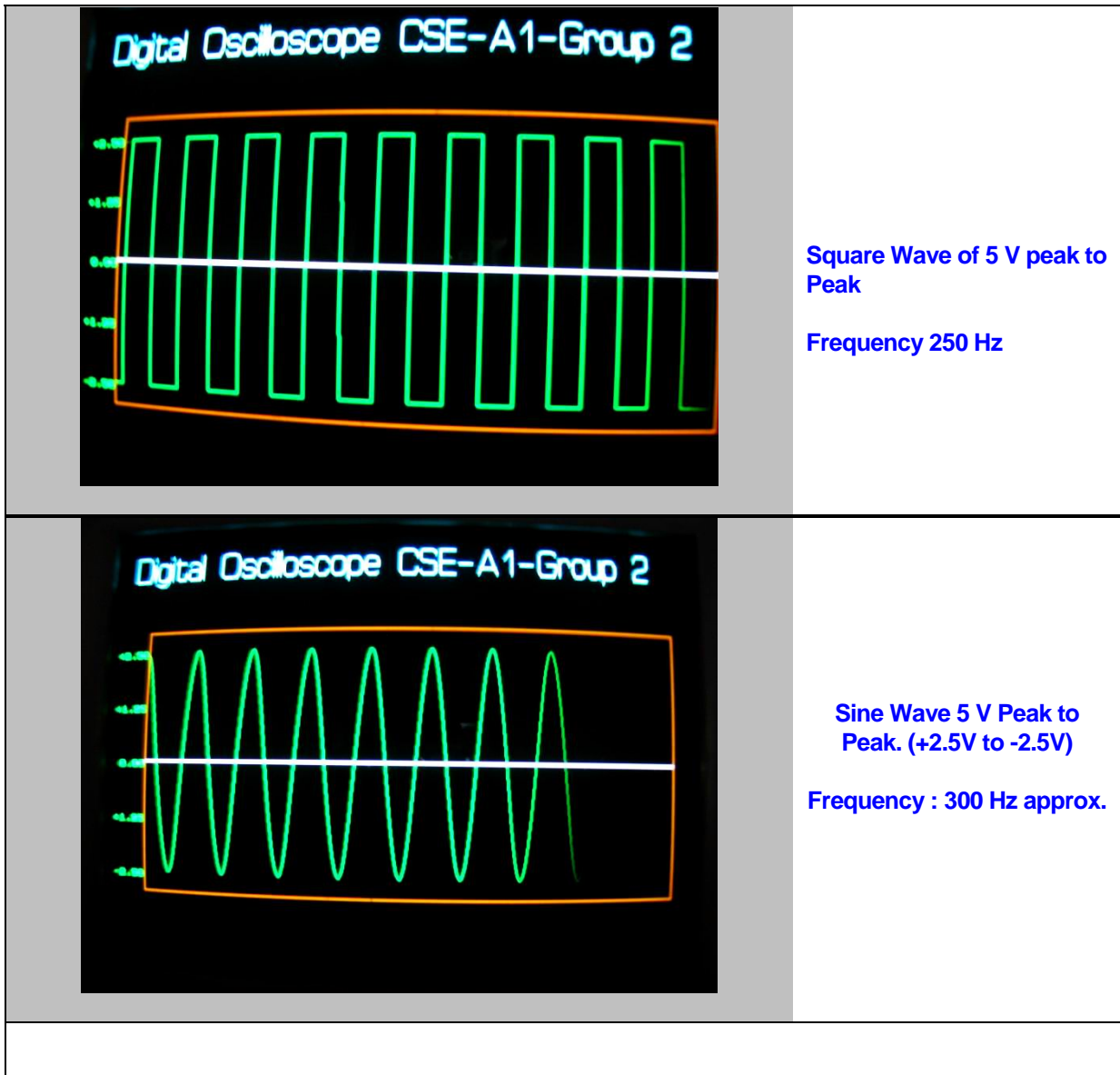
    return result_byte;
}
```

6.2 Software to display Graphically

Using several Functions from Turbo C++ 3.0 BGI graphics library, we implement the software, which shows the output like an oscilloscope. Full source code is at Appendix A

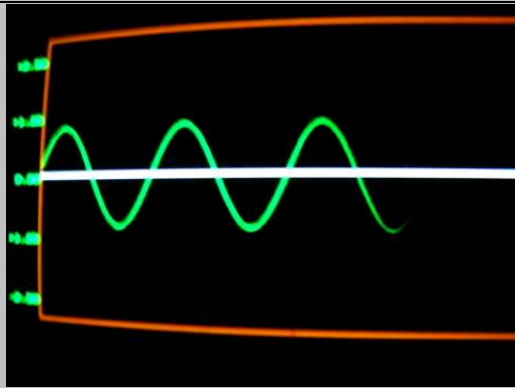
Section / Testing

The Completed project was tested with various frequency and amplitude sine wave, square wave and triangular wave taken from function generators. The snapshots of output during lab test of the project are given below.





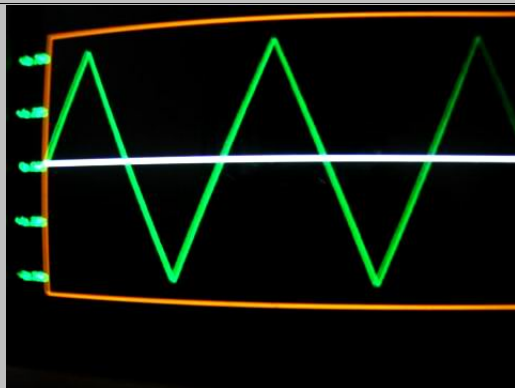
Triangular wave of 7 V Peak to Peak. As the ADC is calibrated to measure -2.5 V to +2.5 volt, Portions of the wave $>2.5V$ or $<-2.5V$ are clipped. Which gives it the shape of a Trapezoidal wave on display.



Sine wave 2.4 V Peak to Peak.

Frequency : 200 Hz

(+1.2 V to -1.2V)



Triangular Wave 100 Hz, 5 V Peak to Peak (+2.5 V to -2.5V)

Frequency : 150 Hz

Our Data Acquisition system can successfully acquire data and display the varying signal in the range +2.5V to -2.5 V.

Our system can successfully display signals of frequency up to 450 / 500 Hz without any visible error. Frequency >500Hz suffer from Data loss, and distorted signal is observed.

There are several issues for the development of this project :

- 1) The +2.5 V to -2.5V range can be easily expanded by using resistive ladder techniques in the input part.
- 2) If we use multithreaded operating system, and buffer the data in PC, frequency range of this acquisition system will increase. Because in DOS environment, we take a data by our software, then plot it and then again take another data. Data loss is occurred when we are plotting one point. Because graphical functions are time consuming. So a better idea is using multi threaded systems like Linux or Windows XP. Then data input from port and plotting the data will be separate and data loss will be less.
- 3) There are limitations for the ADC chip we used. By using a better ADC we can increase the frequency range.
- 4) A pc monitor refresh rate is in the range of 60 Hz to 100 Hz. So to view high frequency data, we need high refresh rate display.

This project was actually an educational project, and we built it experimentally. If we want commercial quality Data Acquisition system, then we need to consider about noise elimination, frequency range, data loss and hence design of the analog-digital part should be more fault tolerant. The software also needs to have a better design.

How to Start:

Software part:

1. Double Click SCOPE.EXE to run the program.

Hardware Part:

1. Connect the circuit with power supply.
2. Connect the (+) lead of the signal source with the wire labelled "Signal"
3. Connect the (-) lead of the signal source with the (+) point of a 2.5V DC Source
4. Connect the (-) of the 2.5V DC source to circuit ground.
4. For free running mode, connect the wire labelled "start" to ground and then disconnect it.
5. Now the ADC has entered Free Running mode

You should now see the waveshapes in the display.

Section 10 Future Works

In future we plan to research on how to increase the frequency range. We also think to improve the software. Such a Data Acquisition system can be a replacement to a costly oscilloscope in some cases.

For high frequency, oscilloscope is the only solution, but for medium to Low frequency such data acquisition systems are low cost solutions.

Reference

- [1] Sedra, Adel S. and Smith, Kenneth C., 2004-2005 *Microelectronic Circuits*. Fifth Edition. Oxford University Press.
- [2] *Microprocessor Data Handbook*, 2003 Edition, BPB Publications.
- [3] Taub Herbert and Schilling Donald, 1977, *Digital Integrated Electronics*, McGraw-Hill International Editions.
- [4] ADC0801/ADC0802/ADC0803/ADC0804/ADC0805 8-Bit μ P Compatible A/D Converters Data Sheet, November 1999, National Semiconductor Corporation
- [5] National Instruments <http://www.ni.com>

A Source Code

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<string.h>
#include<time.h>

/* Digital Oscilloscope Source Code. This software continuously reads values
from parallel port, and plots them graphically using Turbo C, graphics
functions.

Author : Md. Tanvir Al Amin Popel
http://popel.vze.com

For any problem, comments, and bug report : tanviralamin@gmail.com

*/

//#define SIMULATION_MODE

class ParallelPort{

    int prevData;
    int LIM ;

public:

    ParallelPort()
    {
        LIM= 128;
        prevData = rand()% LIM;
    }

    void writePort(int x)
    {

    }

    int readPort()

    {
#ifdef SIMULATION_MODE

        int x = rand() % 20;
        if(rand()%10 >= 5) x = -x;
        int temp = prevData+x;
```

```

        if(temp >= LIM)temp = prevData -x;
        else if(temp <= -LIM)temp = prevData -x;
        prevData = temp;
        return temp;

    #endif

    int status;
    int control;
    int result_byte= 0;
    outportb(0x378+2,0x4);
    status = inportb(0x378+1);
    control = inportb(0x378+2);
    status &= 120;
    control &= 15;
    status <<= 1;
    control ^= 11;
    result_byte |= status;
    result_byte |= control;
    return result_byte;

}

~ParallelPort()

{

}

};

class Oscilloscope{

    int height,width,startX ,startY;
    int screenHeight,screenWidth,penColor;
    int timeBase;
    int curPos;
    int prevPos;
    int prevX;

public:

    void setScreenPosition(int x,int y)
    {
        startX = x;
        startY = y;

    }
}

```

```

Oscilloscope(){

    height = 300;
    width = 600;
    startX = 20;
    startY = 100;
    int gdriver=DETECT,gmode,errorcode;
    initgraph(&gdriver, &gmode, "");
    errorcode = graphresult();

    if (errorcode != grOk) /* an error occurred */
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);          /* return with error code */
    }
    screenHeight = getmaxy();
    screenWidth = getmaxx();
    penColor = WHITE;
    curPos = 0;
    setfillstyle(0,0);
    timeBase= 1;

}

void drawWindow()
{

    setcolor(BROWN);
    rectangle(startX,startY,startX+width,startY+height);
    bar(startX+1,startY+1,startX+width-1,startY+height-1);
    setcolor(WHITE);
    line(startX+1,startY+height/2,startX+width-1,startY+height/2);
    setcolor(GREEN);

}

void feedData(int x)
{

    int newPos = startY + height/2 - x;

    if(curPos==0)

        putpixel(curPos+startX+1,newPos,penColor);

    else
    {
        line(curPos-timeBase+startX+1,prevPos,curPos+startX+1,newPos);
    }
    curPos+=timeBase;
    prevPos = newPos;
}

```

```

        if(curPos>=width-1)
        {

                setcolor(WHITE);
                //bar(startX+1,startY+1,startX+width-1,startY+height-1);
                bar(startX+1,startY+1,startX+width-1,startY+height/2-1);
                bar(startX+1,startY+height/2+1,startX+width-
1,startY+height-1);
                line(startX+1,startY+height/2,startX+width-
1,startY+height/2);
                setcolor(GREEN);
                curPos = 0;

        }

}

};

void drawTitle()

{

        char title1[] = "Digital Oscilloscope CSE-A1-Group 2";
        setttextjustify(CENTER_TEXT, TOP_TEXT);
        setcolor(LIGHTCYAN);
        setttextstyle(9,0,2);
        outtextxy(getmaxx()/2,0,title1);

}

void main()

{

        Oscilloscope osc;
        ParallelPort prt;
        osc.drawWindow();
        drawTitle();
        int y;
        clock_t nowClock,prevClock;

        while(!kbhit())

        {

                //clock1 = clock();
                y = prt.readPort();
                //y gives data in the range 0 to 255
                // but we need -128 to +127
                osc.feedData(y>>1);

        }

}

```

