UNIVERSITI SAINS MALAYSIA

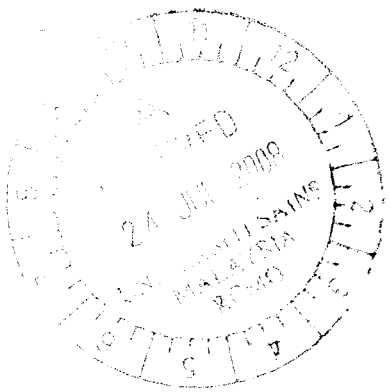
Laporan Akhir Projek Penyelidikan Jangka Pendek

Low Cost Data Acquisition System

by
Zaini Abdul Halim

2008

LOW COST DATA ACQUISITION SYSTEM



by

ZAINI ABDUL HALIM

Final Report
Short Term Grant
USM

May 2008

ACKNOWLEDGEMENTS

First and foremost, I would like to express my utmost gratefulness to God Almighty for giving me the strength, wisdom and perseverance in successfully accomplishing my research.

I would like to thank my research assistant, Mohd Pasha Abdul Razak, for helping me design some circuits. I also would like to thank to my friends, technicians, administrative staffs of Electrical and Electronic Department and many others who have been involved directly or indirectly throughout my research.

Last but not least, I would like to express my sincere gratitude to Universiti Sains Malaysia for the short term grant I received in pursuing this project.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

ABSTRACT

The purpose of this research is to develop a low cost data acquisition system. Microcontroller 8051 has been used to control the process of reading data from sensors and transfer the data to PC. 8051 is a low cost 8 bit microcontroller with 4Kbytes ROM, 128 bytes RAM, 32 I/O pins, 1 serial port and 6 interrupts source is suitable for simple application. The board can be interfaced to external device that produce voltage or current. Analog digital converter is used to convert the analog voltage signal to digital voltage signal. Current to voltage converter is used in order to convert the analog current signal to voltage signal. The voltage signal must be fed into analog digital converter in order to get the digital voltage signal. Digital voltage signal will be used in microconteller for processing purpose. Besides the input voltage and input current, the board is also equipped with some basic applications such as relay circuit, temperature sensor and potentiostat circuit. The potentiostat circuit has been tested to see the capability of the system. Magnasiun sulphate, natrium chloride and potassium fericyanide have been used in the experiments. After tested for three times continuously, the results show that the system is stable. There is no fluctuation in the reading. The reading can be visualized on PC every one second. The data also can be saved in Microsoft excel format for further analysis.

CHAPTER 1

INTRODUCTION

1.1 Data Acquisition System

Data acquisition is a process used to collect information to document or analyze some phenomena. A simple example of data acquisition system is logging the temperature of an oven on a piece of paper. As technology progressed, this type of process has been simplified and made more accurate, versatile and reliable through electronic equipment.

Resolution refers to the smallest signal increment that can be detected by a data acquisition system. Resolution is determined by the analog to digital converter. Hence Analog digital converter is the heart of most data acquisition systems. For example 12 bit analog digital converter will produce a system with 12 bit resolution, one part in 4096 resolution or 0.0244% of full scale.

Sample rate is the speed of data acquisition system which is typically given by the speed of the analog to digital converter. There are four types of data acquisition system:

1) Serial communication data acquisition system
2) Universal Serial Bus (USB) data acquisition system
3) Data Acquisition plug in board
4) Parallel port data acquisition system

RS232 is the most common standard for serial communication system. However it only supports communication to one device at a time and the transmission distance is only 50 feet. Another standard for serial communication is RS485. It is more flexible in

that it can support to more than one device at a time. Transmission distance can be up to 5000 feet.

The USB is a new standard for connecting data acquisition systems to a PC. There are some advantages of USB over serial port and parallel port, including higher bandwidth and the ability to provide power to the peripheral device. Since USB connections can supply power, only one cable is required to link the data acquisition device to the PC.

Computer plug-in board is another type of data acquisition system. The advantage of this system is high speed since it is connected directly to the computer bus. Each board installed in the computer is addressed at a unique input/output map location. The I/O map in the computer provides the address locations that are used by processor to access the specific device as required by its program.

Parallel port can also be used to connect data acquisition system to PC. The system can support very high sample rate. However the distance between the computer and the data acquisition device is limited to a few feet.

DAQ hardware without software is of little use and without proper controls hardware can be very difficult to program. Hence there is software as a user interface to acquire the data and to process the data. Data can be sent serially to PC. Besides microcontroller, FPGA also can be used to get the data from ADC and sent the data to PC continuously.

1.2 Motivation

Traditionally measurements are done on stand alone instrument of various type oscilloscope, multimeters, counters and others. However the need to record the measurements and process the collected data for visualization has become increasingly important. One way to measure signals and transfer the data into a computer is by using data acquisition board. Data acquisition begins with the physical phenomenon to be measured. This physical phenomenon could be temperature, pressure, fluid flow others. Sensors will convert the physical phenomenon into measurable signal like voltage or current. Although there are many data acquisitions system in the market and most of them are produced by a well known company, but they are costly. Hence this project proposed a prototype of low cost data acquisition system but still reliable in its application.

1.3 Research Objectives

The objectives of this research are:

1) to develop graphical user interface software for data acquisition system
2) to develop data acquisition board that can accept input voltage and input current
3) to equip the data acquisition card with some basic applications such as relay, temperature sensor and potentiostat circuit.

1.4 Research Scope

The goal of this research is to develop a low cost data acquisition system. The system can be interfaced to PC through serial port. Data can be saved in Microsoft Excel file for further analysis. Input data can be voltage or current. The input range for voltage

is 0V until 5 volt and the input range for current is from 4mA to 20mA. The board is equipped with some basic applications such as relay, temperature sensor and potentiostat. To evaluate the capability and applicability of the system, a series of experiment have been performed using potentiostat circuit.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Data acquisition system consists of two parts, hardware part and user interface software part. Hardware part consists of microcontroller circuit to control the whole process and signal conditioning circuit to condition the signal so that they are available to be processed by microcontroller unit. Other circuit can be interfaced to microcontroller such as temperature sensor and pressure sensor which is depends on user's application.

User interface software will provide a window to user to select which operation they are engage on certain time. Data will be displayed on the screen and will be saved for further process. Time and date also will be recorded for their reference.

2.2 The 8051 Microcontroller

The Intel 8051 is a Harvard architecture, single chip microcontroller which has developed by Intel in 1980 for use in embedded systems. It provides many functions such as CPU, RAM, ROM, I/O port, interrupt logic and timer in a single package. It has 8 bit data bus which means that it can access 8 bits of data in one operation. 8051 has 16 bit address bus in which it can access 2^{16} memory locations or 64KB each of RAM and ROM.

It has 128 bytes of on chip RAM and 4kB of on chip ROM. 8051 provides four bytes bidirectional input/output port, serial port, two evel interrupt priority and power saving mode.

The original 8051 core ran at 12 clock cycles per machine cycle, with most instructions executing in one or two machine cycle. With a 12MHz clock frequency, the 8051 could execute 1 million one cycle instruction per second or 500,000 two cycle instructions per second. Enhanced 8051 cores are now commonly used which run at six, four, two or even one clock per machine cycle, and have clock frequencies of up to 100MHz.

2.3 Analog Digital Converter

Most of signal coming from sensors are analog signal. These signals must be converted to digital signal for processing application. The analog signal will be changed into digital signal by Analog Digital Converter (ADC). Precision of the analog input signal converted into digital format is dependent upon the number of bits the ADC used. The resolution of the converted signal is a function of the number of bits the ADC uses to represents the digital data. The higher the resolution, the higher the number of divisions the voltage range is broken into and therefore the smaller the detectable voltage changes.

An 8 bit ADC gives 256 levels (2^8) of digital signal. If the full scale of the input signal is 5V, then the LSB for 8 bit ADC corresponds to $5/2^8 = 0.0195V = 19.5mV$ (Mazidi, 2000). Instead of microcontroller, FPGA can be used to get the data from ADC and sent the data to PC continuously.

2.4 Potentiostat

Potentiostats are widely used in electroanalytical techniques to identify, quantify and characterize redox active species including inorganic, organic and biochemical

spesies. Some examples of electroanalytical methods requiring potentiostatic control of the experiment include analysis of corrosion, materials properties and in vivo detection of biologicals such as glucose and catechol amines.

A potentiostat is an electronic instrument capable of imposing electrical potential waveforms across a working electrode relative to a reference electrode. These electrodes are contained in an electrochemical cell. The working electrode is the electrode where the potential is controlled and current is measured. The reference electrode is used in measuring the working electrode potential. A reference electrode should have a constant electrochemical potential as long as no current flows through it. In order to maintain a constant potential over all conditions, a third electrode which is called counter electrode is used to conduct current into and out of the cell. This current has to exactly balance the current generated at the working electrode.

The equivalent circuit of the three electrodes system in electrochemical is shown in Figure 2.1. The current generator responds to the analyte chemical reacting at the working electrode. The capacitor represents the double layer capacitance, which depends on the analytes. The resistor at counter electrode represents the solution resistance between the counter electrode and the current source. There is another resistor in series with the reference electrode, although the reference electrode does not generally conduct current.

With this background, it shows that three electrodes electrochemical sensor has the following properties: it has to maintain a fixed potential between working electrode and reference electrode, it is bipolar and will operate regardless of whether the current

flows to or from the working electrode and it must measure the current from working electrode, delivering a usable signal to an output terminal.

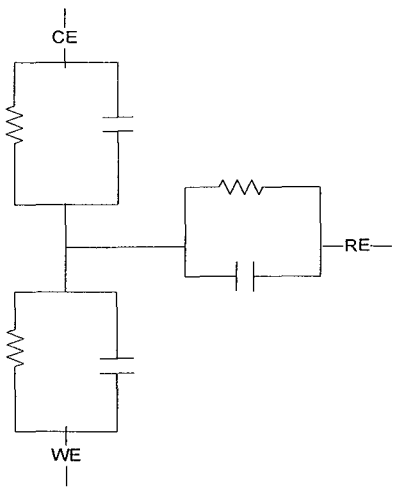


Figure 2.1: Equivalent circuit of the three electrode cell

The potentiostat circuit consists of two parts: control circuit with bias voltage and current measuring circuit. The potentiastat circuit is shown in Figure 2.2. The control circuit provides current to the counter electrode to balance the current required by the working electrode. Operational amplifier, with the low input bias current 50pA, low offset voltage is the basic component in this circuit. The inverting input of the operational amplifier is connected to the reference electrode and must not draw any significant current from the reference electrode.

When the circuit is turned on, the IC2 provides current to maintain the working electrode at the same potential as the reference electrode. A bias voltage is applied on the non-inverting of the IC2. Circuit stability and noise reduction in the control circuit relies on R1, R2, C1 and C2 as shown in Figure 2.2.

The current measuring circuit is a single stage operational amplifier (IC1) in a transimpedance configuration. The current is reflected across R4, generating an output voltage. The measuring circuit uses a combination of the load resistor R_{load}, internal operational amplifier resistance and the internal operational amplifier capacitance to establish an RC circuit. This RC circuit affects both the rms noise and the response time; the response time increase linearly with increasing R_{load} resistance while noise decrease rapidly with increasing R_{load} resistance.

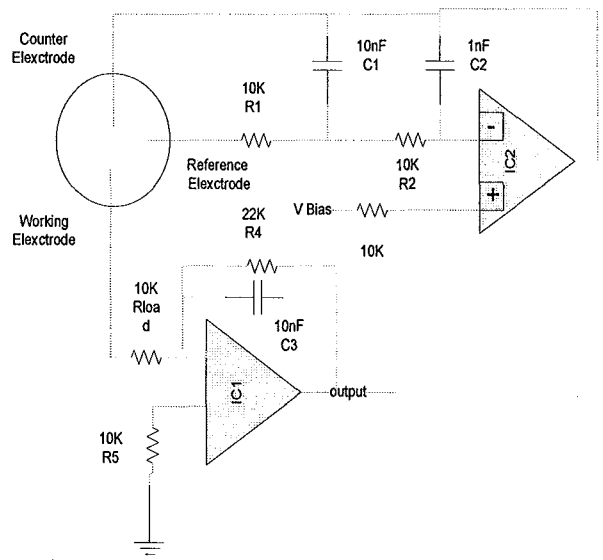


Figure 2.2: Potentistat circuit with input bias voltage.

2.5 Interfacing Software

Data acquisition board will need a medium to display the collected data. Examples of the medium are monitor screen of PC and LCD. Hence, software must be developed to interface the data acquisition board to the display medium for user's application. Assembly language, C language and VHDL can be used to program the LCD whereas C language and Visual Basic can be used to develop the interfacing software on PC.

CHAPTER 3

HARDWARE AND SOFTWARE DEVELOPMENT

3.1 Data Acquisition Board

The board consists of five parts which are microcontroller, potentiostat circuit, input current, relay device and temperature sensor. The embedded software are coded in assembly language (as shown in Appendix 1) to perform the data acquisition, serial communication with a PC and other house keeping task. Data will be sent to PC in one data frame format that consist of one byte of start data, 2 bytes of sensor's data and one byte of end of data.

3.2 Microcontroller Based Circuit

The whole system is controlled by 8051 microcontroller. The schematic for microcontroller is shown in Figure 3.1. The crystal with 11.0592MHz has been used as a clock source to microcontroller in this application. ADC0809 is an 8 bit analog digital converter. However the board will only provide 4 bit or 4 channels to user. The rest 4 bits are used by internal application. LTC1298 is a 12 bit analog digital converter and the board will provide 2 channels or 2 bits for user's application. The board provides serial port for user to interface to PC. Max232 has been used to convert the voltage level from TTL level to PC level for serial communication.

LM7812C is a voltage regulator to provide 12 volt power supply. 12 volt power supply will be used in potentiostat circuit.

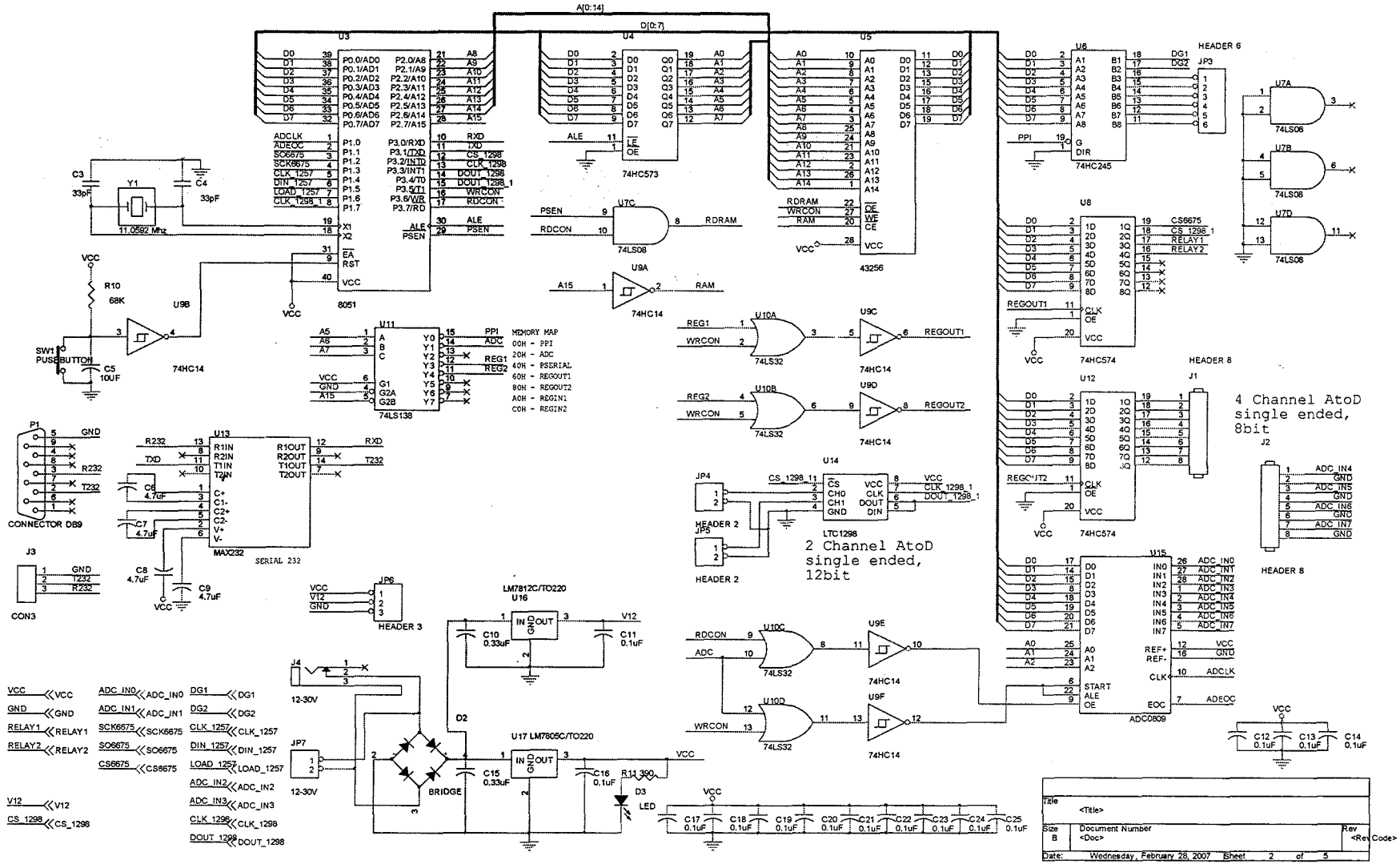


Figure 3.1: Microcontroller based circuit

3.3 Potentiostat Circuit

The potentistat circuit consists of two parts, control circuit with bias voltage and current measuring circuit. Figure 3.2 shows the schematic diagram for potentiostat circuit. LTC 1257 is used to provide bias voltage to the circuit and LTC1298 is a 12bit A/D converter, is used to convert the analog data to digital data for further processing. Operational amplifier OP296 is used in this application. The power supply for this device is 12V. The device with input bias current 50nAmp and the offset voltage is 300μV are suitable for this application.

3.4 Relay

The board provides two relays for user's application. The schematic diagram of the circuit is shown in Figure 3.3. MPSA13 is a NPN Darlington transistor which is used to turn on the relay. The outputs of the relays are connected to HEADER6 as shown in figure 3.

3.5 Input Current (4 to 20mA)

Max471 is a current sense amplifier and the input current will be connected to this device. The schematic of the circuit is shown in Figure 3.4. The output of the Max741 is connected to Max951 which is the operational amplifier. This operational amplifier has a unique output stage that enables it to operate with a low supply current while maintaining linearity under loaded, make it suitable for this application. The output of this operational amplifier is connected to ADC0809, channel 0.

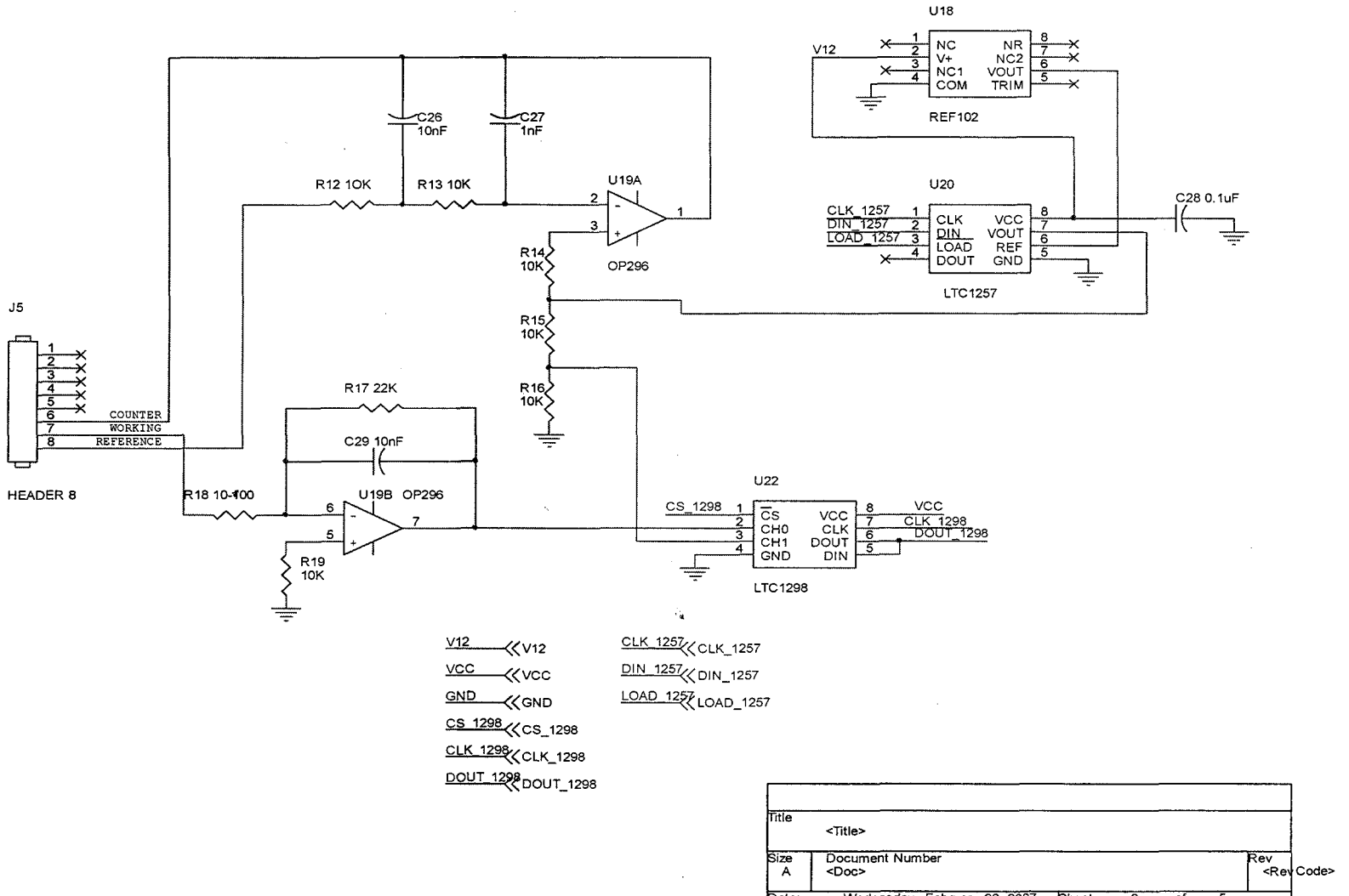


Figure 3.2: Potentiostat circuit

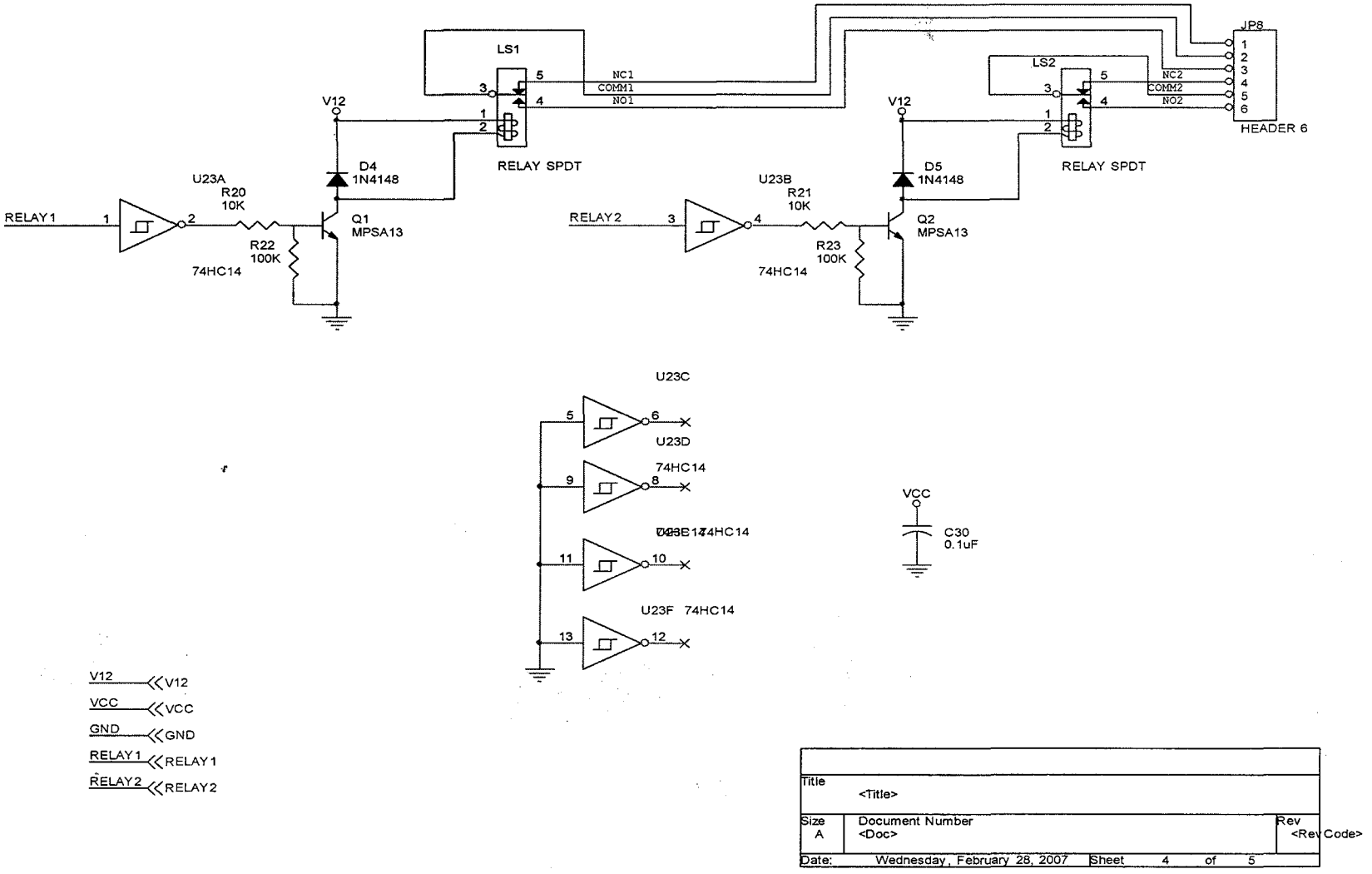


Figure 3.3: Relay circuit

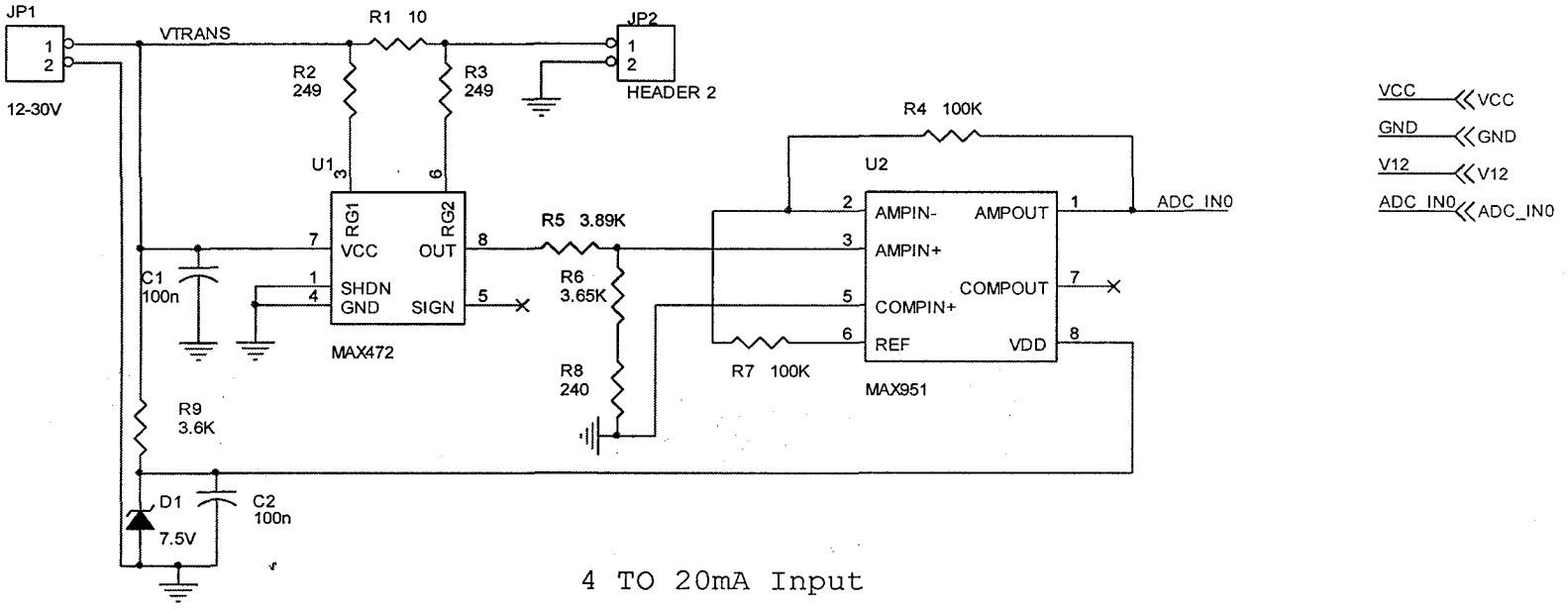


Figure 3.4: Input current circuit

3.6 Temperature sensor

There are three types of temperature sensor, which are K thermocouple, PT100 and PT200. The schematic diagram is shown in Figure 3.5. The outputs of these sensors are connected to amplifier and the aoutput of the amplifiers are connected to ADC0809.

3.7 User Interface Software

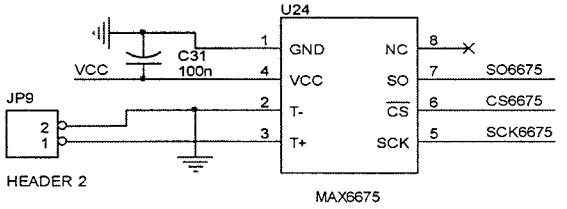
In user interface software, six channels analog to digital converter are available for user's application. Four of them are for 8 bit application and two of them are available for 12 bits application. User can see the reading on the screen and the data also provided in Microsoft excel format. Time and date are also visualized on the PC screen.

For potentistat circuit, data can be visualized on screen in a table format. There are four columns, which are time to record the real time, Channel 1 (V) that represents the input bias voltage, channel 1(nA) that represents the output current and channel 2 (V) that represents the output voltage. Data also can be saved in Microsoft Excell format.

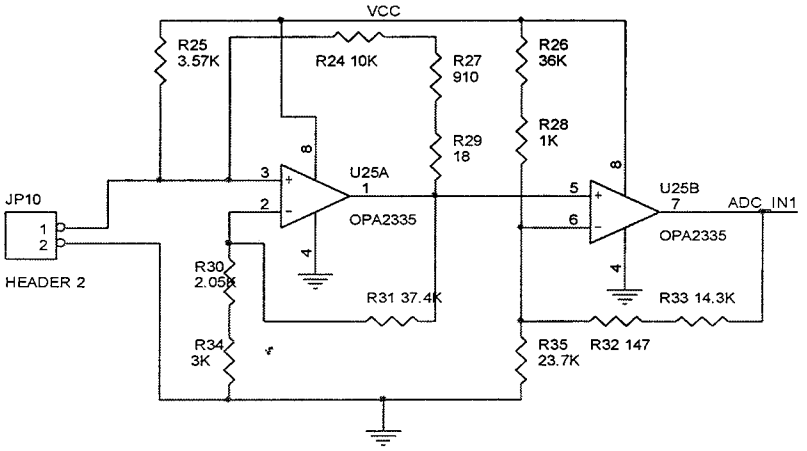
For temperature sensor application, user can select which type of temperature sensor they are using. The data will be displayed on screen in degree of Celcius. The data also can be captured in Microsoft excel format.

There are two options for relay, relay 1 or relay 2. Each relay has two statuses, either normally close or normally open. User can select whether to choose normally close or normally open, depends on their application.

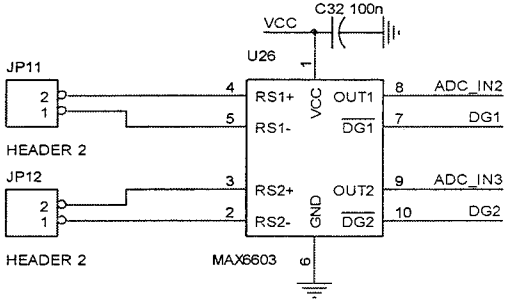
For input current interfacing, the range of the input current is 4 to 20mAmp. Output can be displayed in voltage on the screen. Data also can be visualized in Microsoft Excel format.



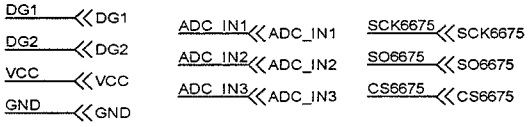
K Thermocouple 0 to 1024 Celcius



PT100 RTD Input 0 to 400 Celcius



PT200 RTD Input 0 to 1000 Celcius



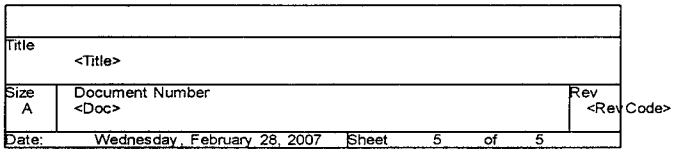


Figure 3.5: Temperature sensor circuit

The user interface software is programmed using Visual Basic Software. The serial port is enabled every one second to capture the data. In Visual Basic Software routine, the data frame is waited until it receives the start byte of data. Once the start byte is received, the software start capture the data until it get the end byte of data.

Figure 3.6 shows the flow chart to read data from analog digital converter and send the data to PC. ADC0809 has an 8 input channels and address are sent using microcontroller to read the right channel. Figure 3.7 shows the flow chart to read data from microcontroller and display the data in Microsoft Excell every one second. The flow chart is referring to potentiostat application.

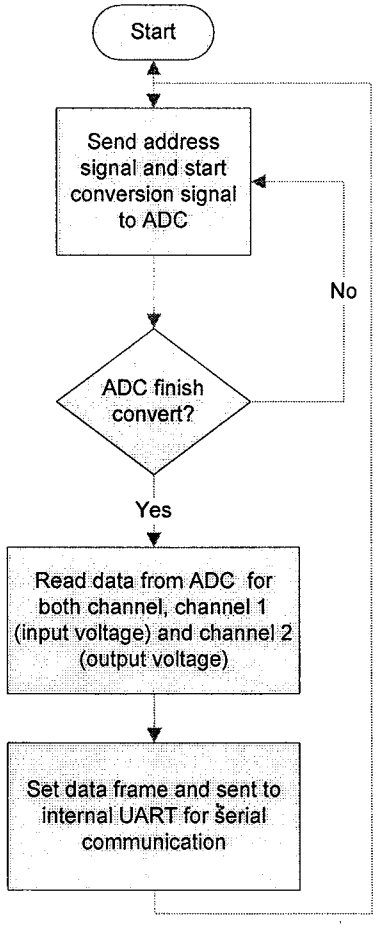


Figure 3.6 : Flow chart to read data from analog digital converter and send the data to PC

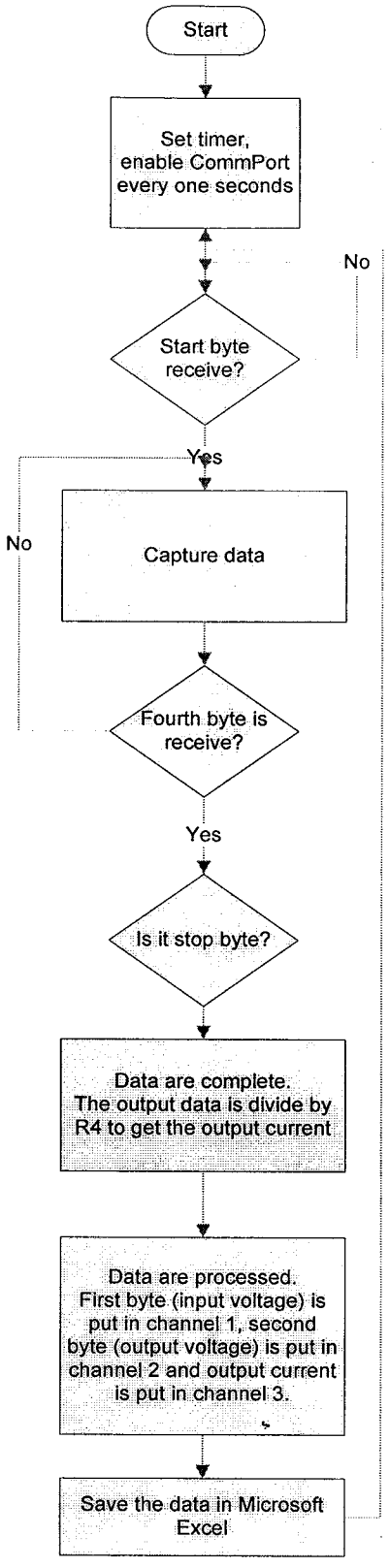


Figure 3.7: Flow chart to display the data on PC

CHAPTER 4

RESULT AND DISCUSSION

4.1 Data Acquisition Board

The layout circuit has been designed using Orcad software. The file has been sent to Asia Printed Circuit Sdn. Bhd. for PCB (printed circuit board) fabrication. The board is shown in Figure 4.1. The size of the board is 4 inch x 4 inch.

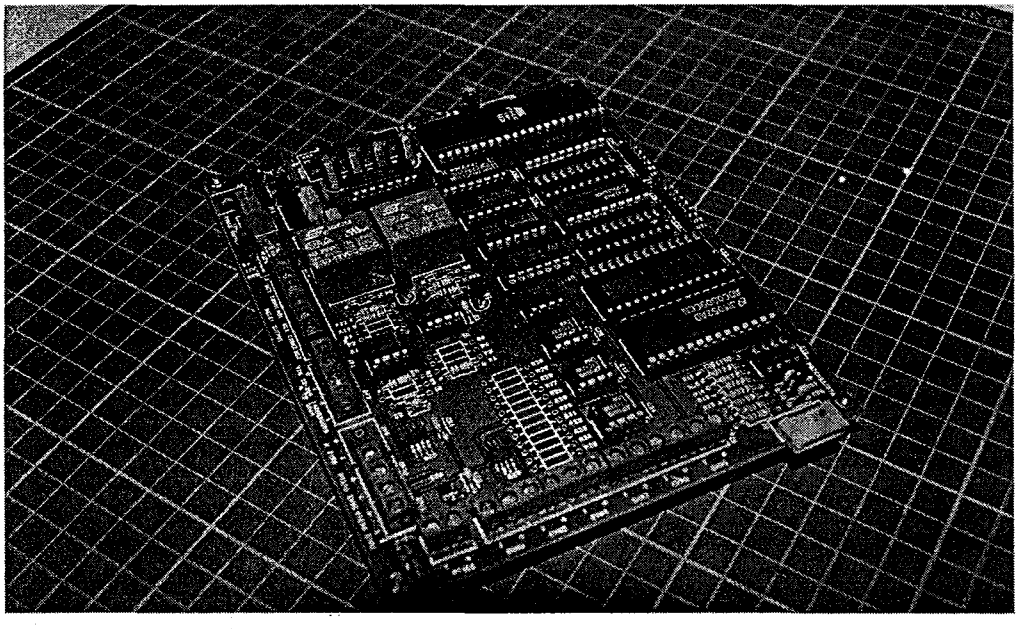


Figure 4.1: Data Acquisition Board

The major components of the data acquisition board are listed in Table 4.1.

Table 4.1: List of components' price

No	Part Number	Description	Price (RM)
1	AT89C51	8051 microcontroller	27.11
2	ADC0809	8 bit analog digital converter	19.51
3	LTC 1298 (2pcs)	12 bit analog digital converter	48.02 x2
4	LTC1257	12 bit DAC	38.84
5	OP296 (2pcs)	Operational Amplifier	12.98 x 2

6	74573	Latch	3.88
7	43256	RAM	54.23
8	74245	Bus Interface	2.62
9	74574 (2 pcs)	Decoder	6.54 x 2
10	7432 (2pcs)	Or Gate	2.58 x 2
11	Max 232	Voltage level converter	3.85
12	74138	Decoder	2.33
13	7408 (2 pcs)	And Gate	2.64 x 2
14	MPSA13 (2 pcs)	Transistor Darlington	0.55 x 2
15	Max 472	Amplifier current sense	15.32
16	Max951	Amplifier	16.79
17	OPA2335 (2pcs)	Operational Amplifier	17.84 x 2
18	Max 7775	Temperature to digital converter	55.91
19	Max 6603	Temperature to digital converter	70.05

Based on the price in Table 4.1, the total price of the major components is RM492.74. The price is based on one piece of components. The price will be reduced around 30% for higher volume (more than 25 pieces).

4.2 User Interface Software

The user interface software has successfully developed using Microsoft Visual Basic Software. The window is shown in Figure 4.2.

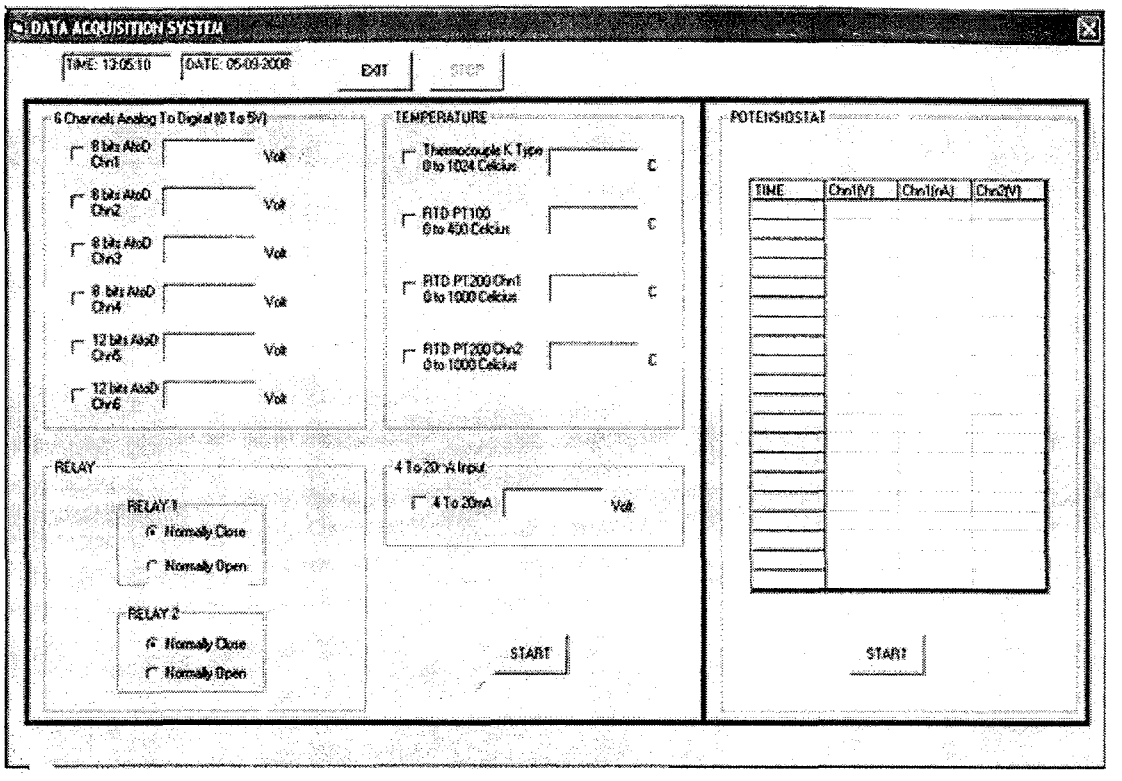


Figure 4.2: User Interface Software's Window

4.3 Testing

The potentiostat has been tested using potassium ferricyanide solution (K3F6(CN6)), magnesium sulphate (MgSO4) solution and natrium choloride (NaCl) solution. Three electrodes system is used. Working electrode, and counter electrode are designed using carbon material and reference electrode is designed using silver/silver chloride (Ag/AgCl). These electrodes are disposable electrodes.

The input bias voltage comes from voltage supply and it is increased by 0.2V every one second. The input voltage is connected to the amplifier IC2 and the output of amplifier is connected to counter electrode. This satisfies the basic requirement of a counter electrode, which needs to be able to provide any amount of current required by electrochemical activity at the working electrode.

The electrochemical current generated at the working electrode is converted into an equivalent voltage by operational amplifier IC1 which is a current to voltage converter and the voltage is measured using microcontroller circuitry. The reference electrode is connected to the inverting terminal of voltage follower circuit. The voltage follower circuit helps in maintaining a constant potential at the reference electrode without drawing much current.

Data are tested three times for every sample to ensure that the circuit is stable. Results summary are shown in Table 4.2 and Figure 4.3. In $MgSO_4$, NaCl and $K_3Fe(CN_6)$, the redox activity occurs within the following input voltage; 5.3955V to 5.4785V, 5.4785V to 5.5371V and 5.41014V to 5.45898V respectively. It also shows that there is no significant difference in output voltage for all experiments. It indicates that the circuit is stable.

Table 4.2: Result Summary

Solution	**Experiment**	**Input Voltage (V)**	**Output Voltage (V)**
$MgSo_4$	1	5.3955	0.10986
	2	5.4541	0.10009
	3	5.4785	0.14892
NaCl	1	5.4785	1.26953
	2	5.5371	1.24999
	3	5.50292	1.25976
$K_3Fe(CN_6)$	1	5.41014	0.87646
	2	5.45898	0.65185
	3	5.41014	0.86669

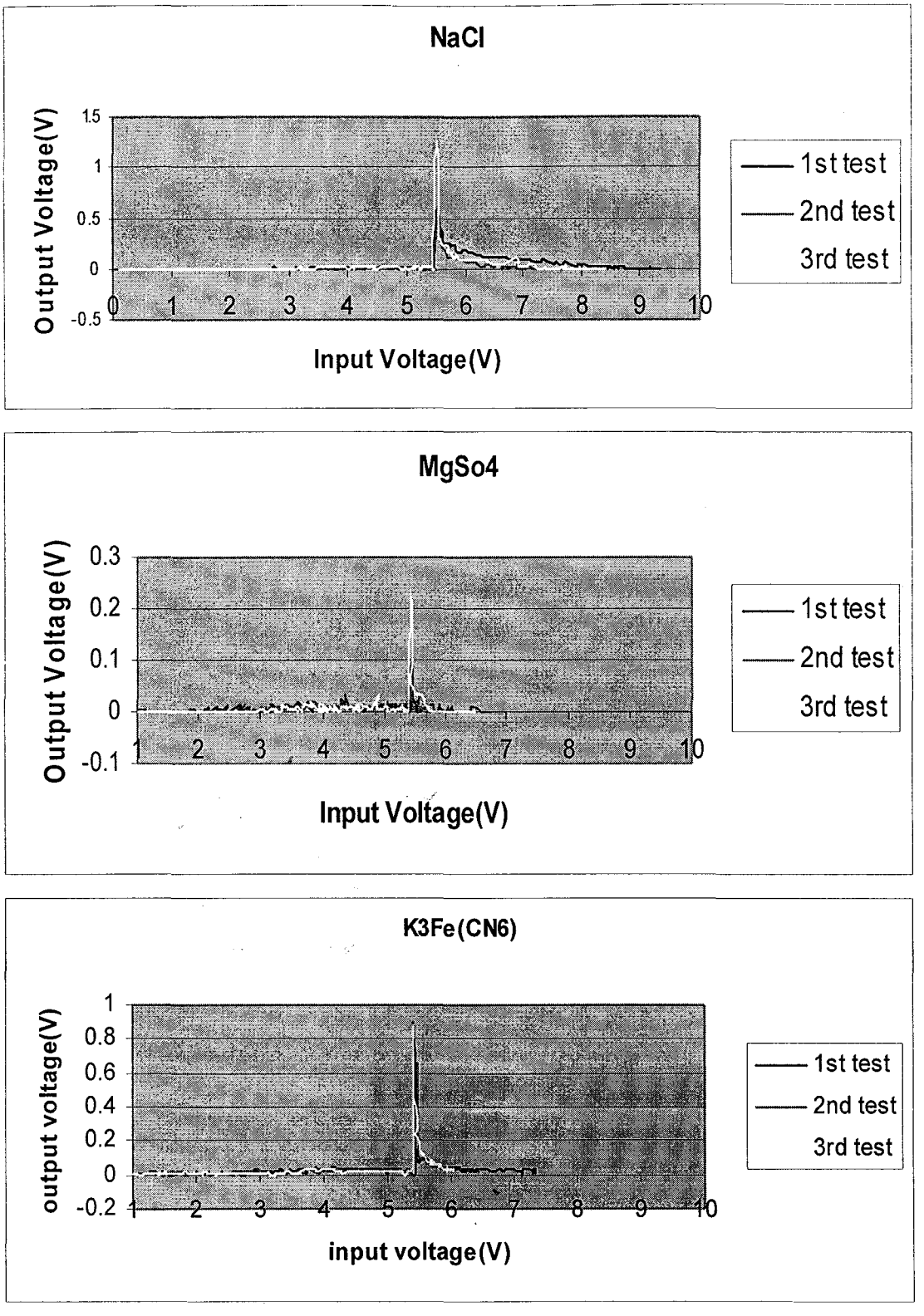


Figure 4.3: Output voltage for MgSO4, naCl and K3Fe(CN6)

CHAPTER 5

CONCLUSION

Data acquisition system has been developed in this project. Microcontroller 8051 has been used to control the whole application. Data can be visualized every one second on PC and the data can be saved in Microsoft Excel format for further analysis. Besides has input voltage and input current, the board also has been equipped with some basic applications such as temperature sensor, relay and potentiostat circuit. In summary, the features of the data acquisition system are as following:

1- Type K Thermocouple

- Data range 0°C to 1024°C.
- 12 bits, 0.25°C resolution.
- Cold junction compensation
- 1 channel

2- PT200 RTD Input

- Data range 0°C to 1000°C
- ±6°C(max) resolution
- ±5kV ESD Protection on RTD Inputs
- +16V overvoltage fault protection on RTD inputs
- 2 channels

3- PT100 RTD Input

- Data range 0°C to 400°C
- 10mV/°C data resolution

- 1 channel

4- Relay Output

- Contact rating- 7A 240VAC, 10A 28VDC
- Channels- 2 x form C
- Relay off time(typical) 4 ms
- Relay on time(typical) 3 ms

5- Analog Input

- 4 channel single ended, 8 bits
- 2 channel single ended, 12 bits
- Support 0-5V input range

6- 4 to 20mA Analog Input

- 1 channel
- 12 to 30V transducer supply

The total price of the board is approximately RM650. The price will be reduced by 30% of the total price if higher volume is considered.

LIST OF PUBLICATIONS

- Lower Memory utilization for Digital Data Acquisition and Transmission Systems, International Conference on Engineering Technology 2007, page 1-5.
- FPGA Implementation on Temperature Data Logger Using SHT75 Temperature sensor, Proceeding of The International Conference on Robotics, Vision, Information and Signal Processing ROVISP07,page 302-306.
- Universal Infrared Receiver For PC Media Player, 4th International Colloqium on Signal Processing and its Application, March 7-9 2008, Kuala Lumpur Malaysia, ISBN: 978-983-42747-9-5.page 652-656
- The development of an Inexpensive Portable Potentiostat, 4th International Colloqium on Signal Processing and its Application, March 7-9 2008, Kuala Lumpur Malaysia, ISBN: 978-983-42747-9-5.page 674-678.

Appendix 1

```
        cpu  "D:\ASMFILES\8051.tbl"
        incl "D:\ASMFILES\8051.inc"

;use ad ltc1298

CS_1298:        EQU     P3_2            ; chip select 1298 no.1
CLK_1298:       EQU     P3_3            ; clock 1298 no.1
DOUT_1298:      EQU     P3_4            ; dout 1298 no.1

CLK_1298_1:     EQU     P1_7            ; clock 1298 no.2
DOUT_1298_1:    EQU     P3_5            ; dout 1298 no.2

CLK_1257:       EQU     P1_4            ; clock d/a 1257
DIN_1257:       EQU     p1_5            ; din d/a 1257
LD_1257:        EQU     P1_6            ; load d/a 1257

ADCLK0809:      EQU     P1_0            ; clock adc0809
ADEOC0809:      EQU     P1_1            ; end of conv. 0809
;Bit Variable
;*************
                org 00
rxsbuf:         dfs     1
pot:            dfs     1

                org 28h
tempstatus:     dfs     1
temp1:          dfs     1
temp2:          dfs     1
temp3:          dfs     1
regda1:         dfs     1
regda2:         dfs     1
regout1:        dfs     1               ;status output port1
regout2:        dfs     1               ;status output port2
STACK:          equ     $

                org 0000h
                jmp start
                ORG 0003H
                ljmp EXTINT0
                org 000Bh
                ljmp T0ISR
                ORG 0013H
                ljmp EXTINT1
                ORG 0023H
                ljmp intserial
                org 0050h

;Program start here
;*******************
START:          mov sp,#stack
                mov TH1,#253
                mov TMOD,#00100001b     ;t1-baudrate, t0-16bit timer
                mov SCON,#01010010b     ;#01011110b
                setb tr1
                setb tr0
                setb ea
                setb es
                mov regout1,#0ffh
                mov regout2,#0ffh
                mov regda1,#00h
                mov regda2,#00h
                clr rxsbuf
                clr pot

loop:           jb pot,pontstrutine
                mov a,#"#"              ;start of data
                call transmit
```

```
                mov dptr,#24h           ;chn4 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#25h           ;chn5 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#26h           ;chn6 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#27h
                call START_CONV0809     ;chn7 0809
                call TRANS_ascii

                call ReadAtoD2          ;AtoD no2 LTC1298

                mov dptr,#20h           ;chn0 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#21h           ;chn1 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#22h           ;chn2 0809
                call START_CONV0809
                call TRANS_ascii
                mov dptr,#23h           ;chn3 0809
                call START_CONV0809
                call TRANS_ascii

                mov a,#"$"              ;end of data
                call transmit
                jmp chkrxsbuf

pontstrutine:   clr c
                mov a,#28h              ;28h = 100mv every 1 sec
                add a,regda2            ;lsb for d to a
                jnc nocarry
                inc regda1              ;inc msb if carry
nocarry:        mov regda2,a
                call delay_1sec
                call DtoA


                mov a,#"#"              ;start of data
                call transmit
                call ReadAtoD1
                mov a,#"$"              ;end of data
                call transmit

chkrxsbuf:      jnb rxsbuf,loop
                mov a,sbuf
                cjne a,#"A",BB          ;relay1 NC
                mov dptr,#60h
                mov a,regout1
                setb acc.2
                movx @dptr,a
                mov regout1,a
                clr rxsbuf
                jmp loop
BB:             cjne a,#"B",CC          ;relay1 NO
                mov dptr,#60h
                mov a,regout1
                clr acc.2
                movx @dptr,a
                mov regout1,a
                clr rxsbuf
                jmp loop
CC:             cjne a,#"C",DD          ;relay2 NC
                mov dptr,#60h
```

```
                mov a,regout1
                setb acc.3
                movx @dptr,a
                mov regout1,a
                clr rxsbuf
                jmp loop
DD:             cjne a,#"D",EE  ;relay2 NO
                mov dptr,#60h
                mov a,regout1
                clr acc.3
                movx @dptr,a
                mov regout1,a
                clr rxsbuf
                jmp loop
EE:             cjne a,#"E",FF
                clr pot
                jmp loop
FF:             cjne a,#"F",brloop
                setb pot
brloop:         jmp loop

delay_1sec:     mov temp1,#8            ;1 sec
                mov temp2,#161
                mov temp3,#115
d1sec:          djnz temp3,d1sec
                djnz temp2,d1sec
                djnz temp1,d1sec
                ret

delay_20ms:     mov temp1,#0ffh
dly_20MS:       mov temp2,#0ffh
                djnz temp2,$
                djnz temp1,dly_20MS
                ret


;***********************************************
;Rutin to read channel 0 and channel 1 AtoD no.1
;***********************************************
ReadAtoD1:      mov a,#11011111b        ;Din word for LTC1298,channel 0,single
ended
                call START_CONV1                ; A/D conversion routine
                mov a,r2
                swap a
                anl a,#0fh
                mov r5,a                ;msb in r5
                mov a,r3
                swap a
                mov r3,a
                mov a,r2
                swap a
                anl a,#0f0h
                add a,r3
                mov r6,a                ;lsb in r6

                mov a,#11111111b        ;Din word for LTC1298,channel 1,single
ended
                call START_CONV1                ; A/D conversion routine
                mov a,r2
                swap a
                anl a,#0fh
                mov r0,a                ;msb in r0
                mov a,r3
                swap a
                mov r3,a
                mov a,r2
                swap a
                anl a,#0f0h
```

```
                add a,r3
                mov r1,a                ;lsb in r1
                mov a,r5
                call TRANS_ascii
                mov a,r6
                call TRANS_ascii
                mov a,r0
                call TRANS_ascii
                mov a,r1
                call TRANS_ascii
                ret

;***********************************************
;Rutin to read channel 0 and channel 1 AtoD no.2
;***********************************************
ReadAtoD2:      mov a,#11011111b        ;Din word for LTC1298,channel 0,single
ended
                call START_CONV2        ; A/D conversion routine
                mov a,r2
                swap a
                anl a,#0fh
                mov r5,a                ;msb in r5
                mov a,r3
                swap a
                mov r3,a
                mov a,r2
                swap a
                anl a,#0f0h
                add a,r3
                mov r6,a                ;lsb in r6

                mov a,#11111111b        ;Din word for LTC1298,channel 1,single
ended
                call START_CONV2        ; A/D conversion routine
                mov a,r2
                swap a
                anl a,#0fh
                mov r0,a                ;msb in r0
                mov a,r3
                swap a
                mov r3,a
                mov a,r2
                swap a
                anl a,#0f0h
                add a,r3
                mov r1,a                ;lsb in r1
                mov a,r5
                call TRANS_ascii
                mov a,r6
                call TRANS_ascii
                mov a,r0
                call TRANS_ascii
                mov a,r1
                call TRANS_ascii
                ret


;**********************
;A/D no 1 LTC1298 RUTIN
;**********************
START_CONV1:    setb CS_1298            ;make sure CS is high
                clr CS_1298             ;CS goes low
                mov r4,#4               ;load counter
loop1:          rlc a                   ;rotate Din bit into carry
                clr CLK_1298            ;SCLK   goes low
                mov DOUT_1298,c         ;output Din bit to LTC1298
                setb CLK_1298           ;SCLK goes high
                djnz r4,loop1           ;next bit
```

```
                mov p1,#04              ;bit 2 becomes an input
                clr CLK_1298            ;SCLK goes low
                mov r4,#9               ;load counter
loop2:          mov c,DOUT_1298         ;read data bit into carry
                rlc a                   ;rotate data bit into acc
                setb CLK_1298           ;SCLK goes high
                clr CLK_1298            ;SCLK goes low
                djnz r4,loop2           ;next bit
                mov r2,a                ;stores MSBs in r2
                clr a                   ;clear acc
                mov r4,#04              ;load counter
loop3:          mov c,DOUT_1298         ;read data bit into carry
                rlc a                   ;rotate data bit into carry
                setb CLK_1298           ;SCLK goes high
                clr p1_3                ;SCLK goes low
                djnz r4,loop3           ;next bit
                mov r4,#04              ;load counter
loop4:          rrc a                   ;rotate right into acc
                djnz r4,loop4           ;next rotate
                mov r3,a                ;store LSBs in r3
                setb CS_1298            ;CS goes high
                ret

;***********************
;A/D no 2 LTC1298 RUTIN
;***********************
START_CONV2:    push acc
                mov dptr,#60h           ;address regout2
                mov a,regout1
                setb acc.1              ;make sure CS is high
                movx @dptr,a
                clr acc.1               ;CS goes low
                movx @dptr,a
                mov regout1,a           ;save back regout2
                pop acc
                mov r4,#4               ;load counter
loop5:          rlc a                   ;rotate Din bit into carry
                clr CLK_1298_1          ;SCLK   goes low
                mov DOUT_1298_1,c       ;output Din bit to LTC1298
                setb CLK_1298_1         ;SCLK goes high
                djnz r4,loop5           ;next bit
                ;mov p1,#04             ;bit 2 becomes an input
                clr CLK_1298_1          ;SCLK goes low
                mov r4,#9               ;load counter
loop6:          mov c,DOUT_1298_1       ;read data bit into carry
                rlc a                   ;rotate data bit into acc
                setb CLK_1298_1         ;SCLK goes high
                clr CLK_1298_1          ;SCLK goes low
                djnz r4,loop6           ;next bit
                mov r2,a                ;stores MSBs in r2
                clr a                   ;clear acc
                mov r4,#04              ;load counter
loop7:          mov c,DOUT_1298_1       ;read data bit into carry
                rlc a                   ;rotate data bit into carry
                setb CLK_1298_1         ;SCLK goes high
                clr p1_3                ;SCLK goes low
                djnz r4,loop7           ;next bit
                mov r4,#04              ;load counter
loop8:          rrc a                   ;rotate right into acc
                djnz r4,loop8           ;next rotate
                mov r3,a                ;store LSBs in r3
                mov dptr,#60h           ;address regout2
                mov a,regout1
                setb acc.1              ;CS goes high
                movx @dptr,a
                mov regout1,a           ;save back regout2
                ret
```

```
;***************
;ADC0809 RUTIN
;***************
START_CONV0809: SETB P1_1       ;AS INPUT PORT
                mov a,#0ffh
                movx @dptr,a            ;dummy utk generate wr
START_CLK:      cpl ADCLK0809
                jnb ADEOC0809,START_CLK
                movx a,@dptr
                ret

AD_OE:          EQU     P1_5            ; Output enable
AD_SC:          EQU     P1_7            ; Start conversion
AD_EOC:         EQU     P1_4            ; End of conversion
AD_ALE:         EQU     P1_7            ; address lacth enable
AD_CLK:         EQU     P1_6            ; clock
AD0:            EQU     P3_2            ; mutiplexer channel addressing
AD1:            EQU     P3_3
AD2:            EQU     P3_4
CONV_PORT:      EQU     P2              ; ADC0808 PORT
;******************
;D/A LTC1257 RUTIN
;******************
DtoA:           setb clk_1257
                setb din_1257
                setb ld_1257

                mov a,regda1            ;msb data in regda1
                rlc a
                rlc a
                rlc a
                rlc a

                clr clk_1257
                call chk_acc            ;B11
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B10
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B9
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B8
                setb clk_1257

                mov a,regda2

                clr clk_1257
                call chk_acc            ;B7
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B6
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B5
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B4
                setb clk_1257

                clr clk_1257
```

```
                call chk_acc            ;B3
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B2
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B1
                setb clk_1257

                clr clk_1257
                call chk_acc            ;B0
                setb clk_1257

                clr ld_1257
                ret

;rutin to write serial data 1257
chk_acc:        jnb acc.7,clrdin
                setb din_1257
                rlc a
                ret
clrdin:         clr din_1257
                rlc a
                ret


EXTINT0:        reti

EXTINT1:        reti

T0ISR:          reti

;****************************************
;SERIAL INTERRUPT
;****************************************
intserial:      jnb ti,rint
                clr ti
                reti
rint:           clr ri
                push acc
                push psw
                mov a,sbuf
                setb RXSBUF
exit_ss:        pop psw
                pop acc
                reti

ascii_table:    dfb     "0123456789ABCDEF"

;*****************
;PUT STRING TO CRT
;*****************
PUT_STRING:     clr A
                movC A,@A+dptr
                JZ ENDDIS
                lcall TRANSMIT
                inc dptr
                jmp PUT_STRING
ENDDIS:         ret

;********************
;convert hex to ascii
;********************
TRANS_ascii:    mov r4,a
                swap a
                lcall nibble
```

```
                add a,tempstatus
                mov tempstatus,a
                mov a,r4
                lcall nibble
                add a,tempstatus
                mov tempstatus,a
                ret
nibble:         push dpl
                push dph
                anl a,#0fh
                mov dptr,#ascii_table
                movc a,@a+dptr
                lcall TRANSMIT
                pop dph
                pop dpl
                ret


CRROUTINE:      mov a,#CR
                lcall TRANSMIT
                ret
NEWLINE:        mov a,#CR
                lcall TRANSMIT
                mov a,#LF
                lcall TRANSMIT
                ret

TRANSMIT:       CLR ES
                mov sbuf,a
                jnb TI,$        ;TRANSMIT
                clr TI
                SETB ES
                ret
;RECEIVE CHAR
RECEIVE:        jnb ri,XIT;RECEIVE
                clr ri
                mov a,sbuf
XIT:            ret
aschex:         lcall aschexbyte
                xch a,b
                lcall aschexbyte
                swap a
                orl a,b
                ret
asclist:        dfb     0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,0
                dfb     0Ah,0Bh,0Ch,0Dh,0Eh,0Fh
aschexbyte:     push psw
                push dph
                push dpl
                mov dptr,#asclist
                clr c
                subb a,#30h
                movc a,@a+dptr
                pop dpl
                pop dph
                pop psw
                ret

                end
```

Appendix 2

```
    Dim oExcel As Excel.Application
    Dim oWB As Excel.Workbook
    Dim oWS As Excel.Worksheet
    Dim oRng1 As Excel.Range
    Dim oRng2 As Excel.Range
    Dim FlagHash As Boolean




Private Sub Check1_Click()

'If Check1.Value = 0 Then
'    Form1.BackColor = vbRed
'ElseIf Check1.Value = 1 Then
'    Form1.BackColor = vbBlue

'End If


End Sub




Private Sub Command1_Click()
MSComm1.Output = "E" '& Chr$(13)
Command1.Enabled = False
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
Timer2.Enabled = True
flagstart1 = True
flagstart2 = False
N = 1
M = 1
Set oExcel = New Excel.Application
        oExcel.Visible = False 'True ' <-- ** Optional **
        Set oWB = oExcel.Workbooks.Add
        Set oWS = oWB.Worksheets("Sheet1")
```

```
Cleanup:
   On Error Resume Next
   oExcel.DisplayAlerts = False

End Sub

Private Sub Command2_Click()
   Timer2.Enabled = False

   oExcel.Visible = True ' <-- ** Optional **
   Call oWB.Close(SaveChanges:=True)  ' <-- ** or True **
   Set oWB = Nothing

   oExcel.Quit
   Set oExcel = Nothing
   Command2.Enabled = False
   Command3.Enabled = True
      Command1.Enabled = True
      Command4.Enabled = True
End Sub

Private Sub Command3_Click()
   'Call oWB.Close(SaveChanges:=True)  ' <-- ** or True **
   Set oWB = Nothing
End
End Sub

Private Sub Command4_Click()
MSComm1.Output = "F" '& Chr$(13)
Command1.Enabled = False
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
Timer2.Enabled = True
flagstart1 = False
flagstart2 = True

MSFlexGrid1.Clear
MSFlexGrid1.Col = 0
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "TIME"
MSFlexGrid1.Col = 1
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "Chn1(V)"
MSFlexGrid1.Col = 2
MSFlexGrid1.Row = 0
```

```
MSFlexGrid1.Text = "Chn1(nA)"
MSFlexGrid1.Col = 3
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "Chn2(V)"
N = 1
M = 1
Set oExcel = New Excel.Application
      oExcel.Visible = False 'True ' <-- ** Optional **
      Set oWB = oExcel.Workbooks.Add
      Set oWS = oWB.Worksheets("Sheet1")

Cleanup:
   On Error Resume Next
   oExcel.DisplayAlerts = False

End Sub

Private Sub Form_Load()

Text1.Text = "DATE: " & Date$
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
MSComm1.InputLen = 1
MSComm1.PortOpen = True
N = 1
M = 1
flagserial = False
flagstart1 = False
flagstart2 = False

'dat$ = "123456789"
Command2.Enabled = False
MSFlexGrid1.Col = 0
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "TIME"
MSFlexGrid1.Col = 1
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "Chn1(V)"
MSFlexGrid1.Col = 2
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "Chn1(nA)"
MSFlexGrid1.Col = 3
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "Chn2(V)"

'Set oExcel = New Excel.Application
```

```
'       oExcel.Visible = False 'True ' <-- ** Optional **
'       Set oWB = oExcel.Workbooks.Add
'       Set oWS = oWB.Worksheets("Sheet1")

'Cleanup:
'   On Error Resume Next
'   oExcel.DisplayAlerts = False
End Sub
Private Sub MSComm1_OnComm()
Static rxAccept As Integer

'Timer2.Enabled = False

Select Case MSComm1.CommEvent

  Case comEvReceive
    rxTemp = MSComm1.Input
   If flagstart1 = True Then
      If flagserial = True Then
         Select Case rxAccept
            Case 0
               If rxTemp = "#" Then
                  rxAccept = 1
                  Exit Sub
               Else: Exit Sub
               End If
            Case 1
               If rxTemp <> "$" Then
                  rxdata = rxdata & rxTemp
                  Exit Sub
               Else
                  flagserial = False
                  If Check1.Value = 1 Then
                     rxdata1 = Mid(rxdata, 1, 2)
                     rxdata1 = Val("&h" & rxdata1) * 5 / 256
                     rxdata1 = Mid(rxdata1, 1, 7)
                     Text3.Text = rxdata1      'chn 1 in voltage
                  Else
                     Text3.Text = ""
                  End If

                  If Check2.Value = 1 Then
                     rxdata2 = Mid(rxdata, 3, 2)
                     rxdata2 = Val("&h" & rxdata2) * 5 / 256
                     rxdata2 = Mid(rxdata2, 1, 7)
                     Text4.Text = rxdata2      'chn 2 in voltage
```

```
Else
   Text4.Text = ""
End If

If Check3.Value = 1 Then
   rxdata3 = Mid(rxdata, 5, 2)
   rxdata3 = Val("&h" & rxdata3) * 5 / 256
   rxdata3 = Mid(rxdata3, 1, 7)
   Text5.Text = rxdata3      'chn 3 in voltage
Else
   Text5.Text = ""
End If

If Check4.Value = 1 Then
   rxdata4 = Mid(rxdata, 7, 2)
   rxdata4 = Val("&h" & rxdata4) * 5 / 256
   rxdata4 = Mid(rxdata4, 1, 7)
   Text6.Text = rxdata4      'chn 4 in voltage
Else
   Text6.Text = ""
End If

If Check5.Value = 1 Then
   rxdata5 = Mid(rxdata, 9, 4)
   rxdata5 = Val("&h" & rxdata5) * 5 / 4096
   rxdata5 = Mid(rxdata5, 1, 7)
   Text7.Text = rxdata5      'chn 5 in voltage
Else
   Text7.Text = ""
End If

If Check6.Value = 1 Then
   rxdata6 = Mid(rxdata, 13, 4)
   rxdata6 = Val("&h" & rxdata6) * 5 / 4096
   rxdata6 = Mid(rxdata6, 1, 7)
   Text8.Text = rxdata6      'chn 6 in voltage
Else
   Text8.Text = ""
End If

If Check11.Value = 1 Then
   rxdata7 = Mid(rxdata, 17, 2)
   rxdata7 = Val("&h" & rxdata7) * 5 / 256
   rxdata7 = Mid(rxdata7, 1, 7)
   Text9.Text = rxdata7      '4 to 20 mA
Else
```

```
    Text9.Text = ""
End If

If Check13.Value = 1 Then
    rxdata8 = Mid(rxdata, 19, 2)
    rxdata8 = Val("&h" & rxdata8) * 400 / 256
    rxdata8 = Mid(rxdata8, 1, 7)
    Text11.Text = rxdata8       'PT100 0-400c
Else
    Text11.Text = ""
End If

If Check14.Value = 1 Then
    rxdata9 = Mid(rxdata, 21, 2)
    rxdata9 = Val("&h" & rxdata9) * 1000 / 256
    rxdata9 = Mid(rxdata9, 1, 7)
    Text12.Text = rxdata9       'PT200 0-1000c chn1
Else
    Text12.Text = ""
End If

If Check15.Value = 1 Then
    rxdata10 = Mid(rxdata, 23, 2)
    rxdata10 = Val("&h" & rxdata10) * 1000 / 256
    rxdata10 = Mid(rxdata10, 1, 7)
    Text13.Text = rxdata10      'PT200 0-1000c chn2
Else
    Text12.Text = ""
End If

If M = 1 Then

Set oRng1 = oWS.Range("A" & CStr(M))
oRng1.Value = "TIME"

Set oRng1 = oWS.Range("B" & CStr(M))
oRng1.Value = "DATE"

Set oRng1 = oWS.Range("C" & CStr(M))
oRng1.Value = "Chn 1"
Set oRng1 = oWS.Range("D" & CStr(M))
oRng1.Value = "Chn 2"
Set oRng1 = oWS.Range("E" & CStr(M))
oRng1.Value = "Chn 3"
Set oRng1 = oWS.Range("F" & CStr(M))
oRng1.Value = "Chn 4"
```

```
                M = 2
                Else

               Set oRng1 = oWS.Range("A" & CStr(M))
                oRng1.Value = Time$

               Set oRng1 = oWS.Range("B" & CStr(M))
                oRng1.Value = Date$

                Set oRng1 = oWS.Range("C" & CStr(M))
                oRng1.Value = rxdata1
                Set oRng1 = oWS.Range("D" & CStr(M))
                oRng1.Value = rxdata2
                Set oRng1 = oWS.Range("E" & CStr(M))
                oRng1.Value = rxdata3
                Set oRng1 = oWS.Range("F" & CStr(M))
                oRng1.Value = rxdata4


                M = M + 1

                End If



                flagserial = False
                Timer2.Enabled = True
                Debug.Print rxdata
                rxdata = ""
                rxflag = True
                rxAccept = 0
                Exit Sub
              End If

          Case Else
             rxAccept = 0
             Exit Sub
          End Select
      Else 'baru
      End If 'baru
      Exit Sub    'baru

'for potensioststst
   Else    'baru
      If flagserial = True Then
         Select Case rxAccept
```

```
Case 0
   If rxTemp = "#" Then
      rxAccept = 1
      Exit Sub
   Else: Exit Sub
   End If

Case 1
   If rxTemp <> "$" Then
      rxdata = rxdata & rxTemp
      Exit Sub
   Else
      flagserial = False
      MSFlexGrid1.Col = 0
      MSFlexGrid1.Row = N
      MSFlexGrid1.Text = Time$

      MSFlexGrid1.Col = 1
      MSFlexGrid1.Row = N
      rxdata1 = Mid(rxdata, 1, 4)
      rxdata1 = Val("&h" & rxdata1) * 5 / 4096 '0.01960784
      rxdata1 = Mid(rxdata1, 1, 7)
      MSFlexGrid1.Text = rxdata1       'chn 0 in voltage

      MSFlexGrid1.Col = 2
      MSFlexGrid1.Row = N
      rxdata3 = (rxdata1 / 4.4) * 1000
      rxdata3 = Mid(rxdata3, 1, 7)
      MSFlexGrid1.Text = rxdata3       'chn 0 in current

      MSFlexGrid1.Col = 3
      MSFlexGrid1.Row = N
      rxdata2 = Mid(rxdata, 5, 8)
      rxdata2 = Val("&h" & rxdata2) * 5 / 4096 '0.01960784
      rxdata2 = Mid(rxdata2, 1, 7) * 2    'dapatkan balik nilai penuh(x2)
      MSFlexGrid1.Text = rxdata2       'chn 1 in voltage

      Set oRng1 = oWS.Range("A" & CStr(M))
      oRng1.Value = Time$
      Set oRng1 = oWS.Range("B" & CStr(M))
      oRng1.Value = rxdata1
      Set oRng1 = oWS.Range("C" & CStr(M))
      oRng1.Value = rxdata3
      Set oRng1 = oWS.Range("D" & CStr(M))
      oRng1.Value = rxdata2
      M = M + 1
```

```
                N = N + 1
                rxdata = ""
                rxflag = True
                rxAccept = 0
                'Debug.Print rxdata
                If N = 21 Then
                   N = 1
                   MSFlexGrid1.Clear
                   MSFlexGrid1.Col = 0
                   MSFlexGrid1.Row = 0
                   MSFlexGrid1.Text = "TIME"
                   MSFlexGrid1.Col = 1
                   MSFlexGrid1.Row = 0
                   MSFlexGrid1.Text = "Chn1(V)"
                   MSFlexGrid1.Col = 2
                   MSFlexGrid1.Row = 0
                   MSFlexGrid1.Text = "Chn1(nA)"
                   MSFlexGrid1.Col = 3
                   MSFlexGrid1.Row = 0
                   MSFlexGrid1.Text = "Chn2(V)"
                Else
                End If
                Exit Sub
             End If

          Case Else
             rxAccept = 0
             Exit Sub
       End Select

    Else 'try
    End If 'try

  End If  'baru

End Select

End Sub

Private Sub Option1_Click()
MSComm1.Output = "A" '& Chr$(13) ' Ensure that
End Sub

Private Sub Option2_Click()
MSComm1.Output = "B" '& Chr$(13) ' Ensure that
End Sub
```

```
Private Sub Option3_Click()
MSComm1.Output = "C" '& Chr$(13) ' Ensure that
End Sub

Private Sub Option4_Click()
MSComm1.Output = "D" '& Chr$(13) ' Ensure that
End Sub

Private Sub Timer1_Timer()
Text2.Text = "TIME: " & Time$
End Sub

Private Sub Timer2_Timer()
'If MSComm1.PortOpen = True Then
   'Exit Sub
   'Else
   'MSComm1.PortOpen = True
   'End If
'MSComm1.RThreshold = 1
flagserial = True
End Sub
```

PAPERS PUBLISHED

Lower Memory Utilization for Digital Data Acquisition and Transmission Systems

M.C. Cheng[1] and **Z. Abdul Halim**[2]
Universiti Sains Malaysia
School of Electrical and Electronic Engineering
14300, Nibong Tebal, SPS, Penang, Malaysia

Abstract—**In most digital data acquisition systems, a digital sampler with high sampling rate may have employed to record the signals before being analyzed by an embedded processor, or stored into a memory buffer. Due to high sampling rate, the signals are recorded as redundant bits of '1's or '0's, which inadvertently increases the memory storage utilization. The data representation technique and memory cell design steps proposed in this paper is aimed to provide compression to the redundant binary bits. Specific purpose applications where the technique can be employed are digital logic analyzers, universal remote controls and data loggers.**

Index Terms— **data acquisition, memory, universal remote, logic analyzer**

I. Introduction

Digital signal sampling is a technique where a serial line of digital signals is continuously recorded by a sampler. Such application can be found in digital data acquisition systems, data loggers and digital transmission systems where the receiver has no proper information of the clock synchronization of the transmitter, and therefore high sampling rate has to be employed to preserve the accurate representation of the original digital signal's waveform. The proposed technique is a method to capture sampled data into memory systems without compromising the accurate representation of the digital signal during the reconstruction process.

Conventional technique used by a data acquisition system is to directly capture the sampled logic values into the memory cells as a series of long logic representation of the signal [1],[2]. However, by dividing the memory array into cluster of cells with proper designed value of its depth, and applying data representation technique to compress its data, the array of memory being occupied for digital storage can be minimized.

Manuscript received on January 31, 2007.

[1] Author is with Universiti Sains Malaysia, working in a R&D project on embedded processors. (phone: (60)12-582-8921, email: mun.chun@yahoo.com)

[2] Author is a researcher and lecturer in Universiti Sains Malaysia working in the field of microprocessors and embedded systems. (phone: (60)13-4331-506, email: eezaini@eng.usm.my)

II. Proposed Technique

A. Data representation of the Sampled Digital Data

Suppose a binary waveform is sampled at a rate which is high enough for the waveform reconstruction to produce an approximate identical binary waveform, a string of binary numbers are recorded.

Proposed technique of data representation for the string of binary number would be in the format of $x(y)$, where;

x = logic representation in logic 1 or logic 0.

y = the number (in decimal value) of sampling period (T_{SSR}) recorded for a specific binary number before the next transition to another binary number.

T_{SSR} = Uniform time period between the interval of signal sampling or signal reconstruction process. (unit: s)

For example, data representation for the reconstructed signal in Fig. 1 is 0(3), 1(2), 0(3), 1(3), ... , 1(2).

B. Memory Cell Architecture

Most memory array organization has evenly distributed number of columns (also known as width, represented by W number of bits) and number of rows (also known as depth, represented by D number of bits) in a W x D matrix. Designing the architecture of the memory to fit the sampled binary representation, the memory array may consist clusters of *information cell* as shown in Fig. 2(a). Each information cell can be used to fit information regarding the binary signal's waveform. Each information cell is defined by a matrix of w x L, where;

L = finite integer representing the level of the information cell used in the array or equivalent to the depth of the cell represented in number of the bits. L = 1, 2, 3, ...

w = finite integer representing the width of the information cell specified in number of bits. w = 1, 2 , 3, ...

It is most convenient for digital system designers to choose integer w as equals to the width of a memory unit used in a system to store the sampled data, mathematically,

$$w = W \tag{1}$$

Equation (1) will be used for the rest of the text. Typical values of W found in most memory unit for embedded systems are 8, 9 and 16.

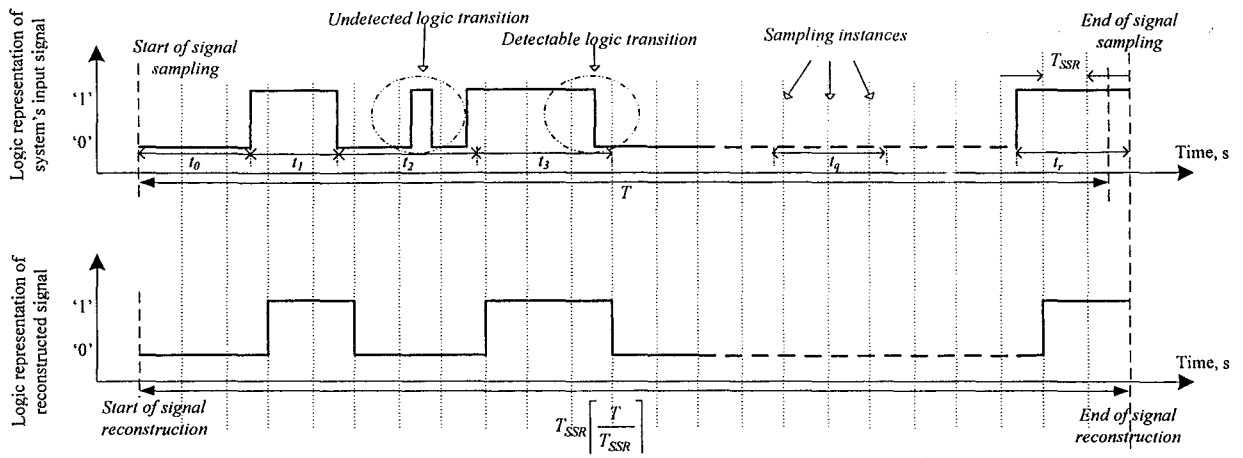


Fig. 1. Logic representation of a system's input signal and its corresponding reconstructed signal for a typical binary waveform.

C. *Data Representation Technique*

The proposed technique place major emphasis in reducing the memory utilization. For a string of binaries with data representation of

$$x_0(y_0), x_1(y_1), x_2(y_2), \ldots x_i(y_i), \ldots\ldots, x_n(y_n) \quad (2)$$

where, n = finite integer where the total number of information cells being used is equivalent to $(n+1)$.

The most significant bit of level L for the i^{th} information cell of matrix array w x L in Fig. 2(b) is used to store the logic level of bit x_i . While remaining bits in the memory array will be directly used to store the value y_i in either big-endian or little-endian format. The rest of the examples are based on big-endian format. The maximum value of y_i (ie. max(y_i)) that can be stored in the grey-shaded area is calculated as in (3).

$$\max(y_i) = 2^{wL-1} \quad (3)$$

The example for w = 8, L = 2 and $x_i(y_i) = 0(15350_{10})$ is also shown in Fig. 2(c). In this example only 2 bytes of the memory array has been used. However by conventional method of storing, a long string of 15350 bits of binary 0's would have been used.

D. *Capturing and Retrieving Data from Information Cell*

Data can be stored (in signal sampling process) or retrieved (in signal reconstruction process) from the cluster of information cells fitted in the memory array. These processes can be performed by a general purpose microcontroller or by a specifically designed embedded processor which is capable to perform the two algorithmic flow in Fig 3.

III. Constraint Of Proposed Technique

From earlier algorithmic flow chart of Fig. 3(a), every transition of information cell for storage will occur whenever the input logic has changed or the boundary condition in (3) has been reached. If the logic levels of the binary waveform would have fluctuate frequently so that the time interval between a valid logic transition is less than the value of T_{SSR}/wL, the transition to the next information cell would have occurred more frequently. Hence, the memory utilization

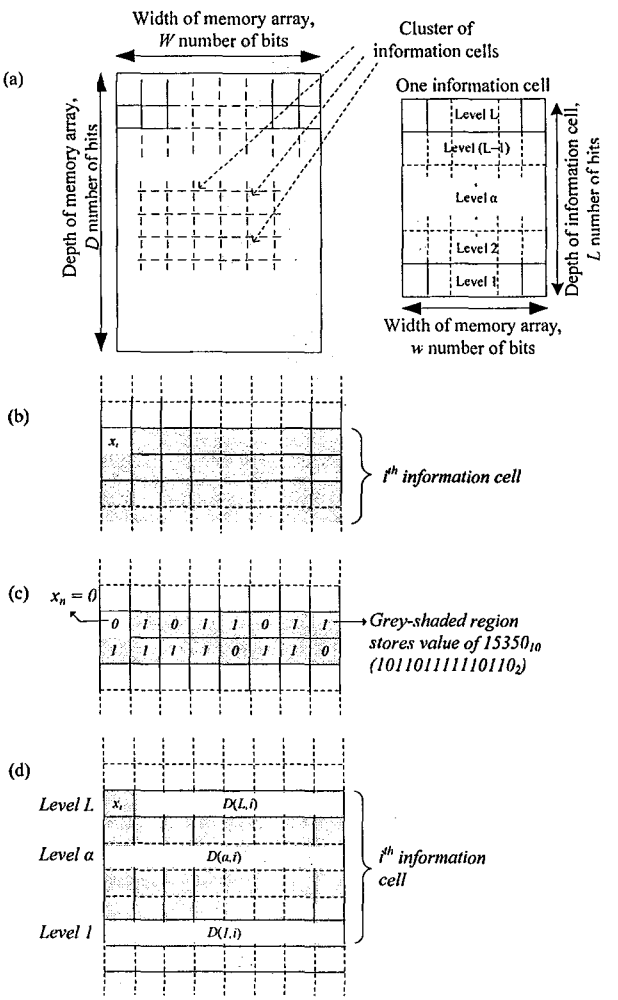


Fig. 2 (a) Memory array of WxL matrix and its cell architecture. (b) Structure of the i^{th} information cell. (c) Example of data 0(15350) being stored into one information cell with L=2. (d) Memory content of the cell in Level-α , represented in $D(\alpha,i)$ format.

would have been higher by using the proposed technique than the conventional method of direct storage. This section is devoted to mathematically derive the constraint of the proposed technique as compared to conventional technique.

Referring back to Fig. 1, assume an arbitrary waveform is recorded for a finite T length of time (unit: s), the total number of memory bits required to capture the binary waveform using conventional method of direct storage would be equivalent to the total number of sample counts conducted for the time interval of T. The total number of sample counts can be mathematically expressed as function $g(T,T_{SSR})$ given in (4).

$$g(T,T_{SSR}) = \lceil T / T_{SSR} \rceil \tag{4}$$

where $\lceil z \rceil$ denotes the ceiling function [3], [4] of the positive real number z. Refer to Appendix for further elaborations.

Within the range of T, the number of information cells $k(t_q)$, that is required to capture a signal with a length of time t_q, can be computed with the equation in (5).

$$k(t_q) = \left\lceil \frac{g(t_q, T_{SSR})}{\max(y_i)} \right\rceil \tag{5}$$

where t_q = time interval between the *detectable logic transition* of the logic level, and detectable logic transition is defined as change of logic level that can be detected by the digital signal sampler. (unit: s)

$q = 0, 1, 2, \ldots (q \le i)$

The number of bits used for storage is represented by function $h(w,L,t_q,T_{SSR})$ is equivalent to the number of bits inside each information cell times the total number of information cells being used, where the parameter L can be found with the method of later section. Mathematical expression is given in (6).

$$h(w,L,t_q,T_{SSR}) = wL\sum_{q=0}^{r} k(t_q) \tag{6}$$

The proposed method can be said to utilize lower memory space than direct storage when the condition in (7) can be fulfilled by choosing appropriate value of L or smaller value of T_{SSR}.

$$h(w,L,t_q,T_{SSR}) < g(T,T_{SSR}) \tag{7}$$

IV. Design Of Information Cell

The following sections are mathematical analysis to guide designers choosing the optimal value of the parameter L and manually encoding the binary values into each level of information cell for a known binary waveform.

A. Design of Parameter L for a Periodic Binary Waveform

In certain digital transmission systems (eg. infrared remote control systems), a repetitive message signal can be sent by the system. Series of a repetitive message signal constructs a series of periodic signals being sent. The transmission process can be achieved by retrieving data from a known cluster of information cells to the controller of the transmitter. The value of L has to be designed and optimized to provide minimum size of memory array.

Defining an arbitrarily periodic signal with time period $T_{message}$ (unit: s), and substituting $T = T_{message}$ into (4), the values of $h(w,L,t_q,T_{SSR})$ with respect to variation of L can be calculated using (6) and tabulated or graphically plotted as $h(w,L,t_q,T_{SSR})$ vs. L. If the *exact value* of T_{SSR} is not specified in design specifications, the value T_{SSR} can be gauged at any reasonable range (lower value of T_{SSR} yields higher accuracy of signal reconstruction) to yield minimum value of $h(w,L,t_q,T_{SSR})$ at any values of L, or else this step can be skipped. The value L is chosen as the value that yields the minimum value of $h(w,L,t_q,T_{SSR})$.

B. Design of Parameter L for an Random Binary Waveform

Random binary waveform can be found in application where the system is used to sample unpredicted pattern of binary waveform (eg. of application is a digital logic analyzer).

For the contemporary time, the proper design method of parameter L for an random binary waveform has not yet been fully been developed. However, it is wise to apply $L = 1$ as the binary waveform might fluctuate frequently, implicating in higher usage of memory space.

C. Formulae for Encoding

A known input logic for any time interval t_q can be mathematically encoded into the information cells. This can be done by calculating the each row's (w x 1) decimal representation of the memory matrix inside the information cells. Example of such application can be found in a remote control system, where the binary waveform of a message signal can be directly encoded into the memory array.

The information cell representing the logic level of x_i with an time interval of t_q can be fitted into one information cell if the condition (8) is met.

$$\max(y_i) > g(t_q, T_{SSR}) \tag{8}$$

Referring to Fig. 2(d), the level α's binary content of the i^{th} information cell can be represented in decimal form with the function represented by $D(\alpha,i)$. If condition (8) is met, the value of $D(\alpha,i)$ can be calculated easily with equation in (9).

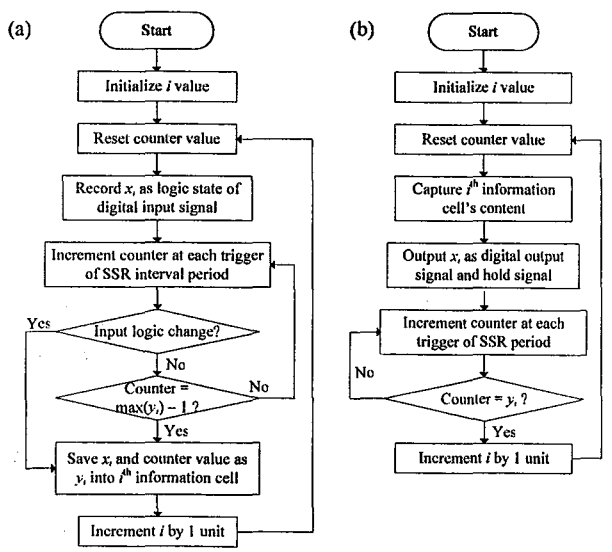


Fig. 3(a) Algorithmic flow for capturing digital input data to be fitted into the information cell iteratively (b) Algorithmic flow for retrieving data from information cell and reconstruction of the desired output's signal.

$$D(\alpha,i)=\left\lfloor \frac{g(t_q,T_{SSR})}{2^{w(\alpha-1)}} \right\rfloor \bmod 2^w \qquad (9)$$

where $\lfloor z \rfloor$ denotes the floor function [3], [4] of the real number z. Refer to Appendix for further elaborations and prove.

However if condition (8) is not met, more than one information cells will be used and the y_i section of the initial $(k(t_q)-1)$ number of cells will be entirely filled with strings of binary '1's or mathematically encoded as the decimal values calculated from the equations in (10);

$$D(\alpha,i+s)=2^w-1 \quad \text{for } \alpha = 1, 2, 3, \ldots, (L-1)$$
$$D(\alpha,i+s)=2^{(w-1)}-1 \quad \text{for } \alpha = L \qquad (10)$$

where s = {0, 1, 2, ..., $(k(t_q)-2)$}.

The last information cell is to be encoded with binaries calculated using (11).

$$D(\alpha,\beta)=\left\lfloor \frac{g(t_q',T_{SSR})}{2^{w(\alpha-1)}} \right\rfloor \bmod 2^w \qquad (11)$$

where $t_q' = t_q - (k(t_q)-1)\cdot \max(y_i)\cdot T_{SSR}$ and $\beta = i + k(t_q) - 1$. Refer to Appendix for further prove.

For information cells of L = 1, or for α = 1, equations in (9) and (11) can be simplified into (12).

$$D(1,\beta') = g(t_q'',T_{SSR}) \bmod 2^w \qquad (12)$$

where the corresponding sets used for substitution are (β', t_q'') = (i , t_q) and (β', t_q'') = (β , t_q') .

V. Design Of Information Cell Using A Hardware Example Of A Universal Remote Control System

Application example of a universal remote control system is chosen because it could demonstrate how the proposed technique could accommodate a multi-protocol embedded system, where message signals generated from the controller of infrared transmitter might not be generated by key logics, but a direct readout of data from the memory unit [5]. This is eliminates the need to hardware-built various key logics for unknown protocols that might not be available to the universal remote control designer [6].

A. Design of Parameter L for Philips RC5 Protocol

Philips RC5 protocol is available in [7]. A typical remote control key's message is a periodic binary waveform to be transmitted via the infrared transmitter after going through a modulator. Fig. 4 is a typical message of 14 bits per message constructed using Philips RC5 remote control's protocol.

High signal reconstruction rate would require greater memory usage, while low signal reconstruction rate would result in poor timing accuracy of reconstructed waveform. Specifying a minimum of 97.5% accuracy relative to the minimum signal length of 899µs of the RC5 protocol, the maximum value of T_{SSR} equals to 22.475µs. Choosing several values for T_{SSR} , the $h(w,L,t_q,T_{SSR})$ function can be calculated by applying equation in (6) to the problem in Fig. 4.

Results in Table I are calculated from;

$$h(8,L,t_q,T_{SSR}) = 8L\left(5\left\lceil\frac{\left\lceil\frac{899\mu s}{T_{SSR}}\right\rceil}{2^{8L-1}}\right\rceil+11\left\lceil\frac{\left\lceil\frac{1789\mu s}{T_{SSR}}\right\rceil}{2^{8L-1}}\right\rceil+\left\lceil\frac{\left\lceil\frac{89727\mu s}{T_{SSR}}\right\rceil}{2^{8L-1}}\right\rceil\right)$$

To meet minimum requirement and least memory array to obtain best accuracy, the value of L = 2 at T_{SSR} = 5µs should be chosen. And for a higher precision waveform, L = 3 at T_{SSR} = 0.1µs can be selected by the designer. It has to be noted that for higher values of L, the six sets of $h(8,L,t_q,T_{SSR})$ will merge to a asymptotic linear line. Employing higher reconstruction rate, will no longer bring significant improvement to accuracy, and depending on the applications, signal reconstruction frequency should be optimized with the power consumption of the particular digital system.

B. Efficiency Analysis

Choosing L = 2 at T_{SSR} = 5µs, only 34 bytes would have been used to store a message signal of 99.4438% timing waveform accuracy, relative to the smallest value of t_q (899 µs). However, using conventional method, signal reconstruction at such accuracy for $T_{message}$ = 114ms would require 2850 bytes (approximately 84 times higher memory usage).

C. Translating Message Signal to Information Cells

Upon designing L = 2 and making decision that T_{SSR} = 5µs, the known waveform is ready to be encoded into the information cells by applying equation in (9), and the summarized result can be found in Table II.

VI. Future Scope And Conclusion

The present proposed technique is undeniably capable to reduce redundancy of sampled logic values if the speed of the sampling clock applied is much higher than the speed of the input's signal variation. It is possible to eliminate this

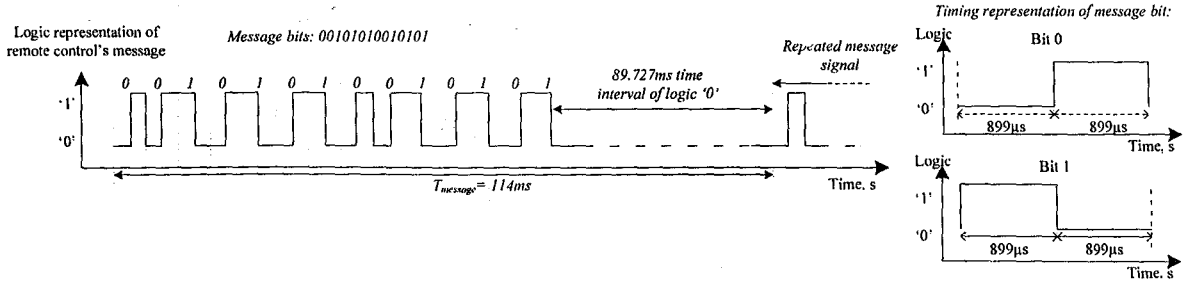


Fig. 4. Logic representation of a typical message generated from Philips RC5 remote control.

TABLE I
NUMBER OF BYTES REQUIRED AS A FUNCTION OF L AND T_{SSR}

T_{SSR}	Accuracy	$h(8,L,t_q,T_{SSR})$					
		L=1	L=2	L=3	L=4	L=5	L=6
20	87.7753	416	272	408	544	680	816
10	98.8888	784	272	408	544	680	816
5	99.4438	1472	272	408	544	680	816
1	99.8888	7160	304	408	544	680	816
0.5	99.9444	14280	352	408	544	680	816
0.1	99.9999	71240	704	408	544	680	816

Unit for T_{SSR} is in μs, Accuracy is measured in percentage (%), unit for $(8,L,t_q,T_{SSR})$ is in *number of bytes*.

TABLE II
ENCODED DATA FOR INFORMATION CELLS

q	t_q	Number of information cells	i	x_i	$D(2,i)$	$D(1,i)$
0	899	1	0	0	0	180
1	899	1	1	1	0	180
2	899	1	2	0	0	180
3	1789	1	3	1	1	102
4	1789	1	4	0	1	102
5	1789	1	5	1	1	102
6	1789	1	6	0	1	102
7	1789	1	7	1	1	102
8	1789	1	8	0	1	102
9	899	1	9	1	0	180
10	899	1	10	0	0	180
11	1789	1	11	1	1	102
12	1789	1	12	0	1	102
13	1789	1	13	1	1	102
14	1789	1	14	0	1	102
15	1789	1	15	1	1	102
16	89727	1	16	0	70	26

Unit for t_q is in μs. Encoded data in $D(2,i)$ and $D(1,i)$ are the decimal representation for the values in binary.

limitation issue. The suggested hint for improvement is to hybrid both conventional technique and the new technique, in other words, improving the algorithmic flow of Fig. 3(a) and Fig. 3(b).

APPENDIX

1) Ceiling function $\lceil z \rceil$:

Ceiling function for a real number z is equaled to the lowest value of integer greater than z . Optional computation formula for $z = A/B > 0$ can be given as; $\lceil A/B \rceil = 1 + (A - A \bmod B)/B$ where mod is a function that can be described as; A *mod* B gives the remainder on division of A by B.

2) Floor function $\lfloor z \rfloor$:

Floor function for a real number z is equaled to the highest value of integer lower than z . Optional computation formula for $z = A/B > 0$ can be given as; $\lfloor A/B \rfloor = (A - A \bmod B)/B$

3) Proof of Equation (9):

For the information cell of level α, each unit of $D(\alpha,i)$ represents the maximum value than can be fitted into the level $(\alpha-1)$ row. Hence, the total number of samples counted for time interval of t_q that is represented by level α and above, is a multiple of $\left\lfloor \frac{g(tq,T_{SSR})}{2^{w(\alpha-1)}} \right\rfloor$. Since, each row is capable to represent the value up to 2^{w-1}, therefore, the value of $D(\alpha,i)$ is given by the remainder of division on $\left\lfloor \frac{g(tq,T_{SSR})}{2^{w(\alpha-1)}} \right\rfloor$ by 2^w.

4) Proof of Equation (11):

Since the values of y_i for the earlier $(k(t_q)-1)$ number of information cells has been fully loaded with the value of $\max(y_i)$, the remaining time section to be encoded into the last information cell using the equation in (9). This can be performed by substituting the variable t_q with t_q' . t_q' is the remaining time section that has not been encoded into the earlier information cells.

REFERENCES

[1] Schumacher, M. LeCroy SA, Geneva, Switzerland, "Advantages of long memories" *6th IEEE Instrumentation and Measurement Technology Conference*, 1989. IMTC-89. pp 156-158, April 1989.

[2] Kerry Beenstra, Krishna Rangasayee, Alan L. Herrmann, "Enhanced Embedded Logic Analyzer", *United States Patent*, Patent No.: US6247147B1 June 2001.

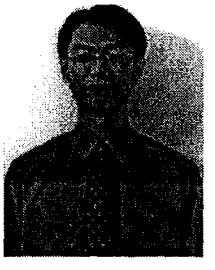
[3] Hallard T. Croft, Kenneth J. Falconer, Richard K. Guy, *Unsolved Problems in Geometry*. New York: Springer-Verlag, p. 2, 1991.

[4] Ronald L. Graham, Donald E. Knuth, Oren Patashnik, "*Integer Functions*", Concrete Mathematics: A Foundation for Computer Science, 2nd ed. Reading, MA: Addison-Wesley, pp. 67-101, 1994.

[5] Van Dootingh, "ESPIRIT Home Systems Project", IEEE Transactions on Consumer Electronics, Vol. 36, No. 3 pp 612- 618, August 1990.

[6] J. Sathyan and A. R. Ramakrishnan, "A Unique Self-Contained Universal Remote Control", *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 4, pp 1151-1155, November 2004.

[7] Paul Seerden, "Application Note", *AN10210 Using the Philips 87LPC76x microcontroller as a remote control transmitter*, Philips Semiconductors, May 2003. Available: http://www.nxp.com/acrobat_download/applicationnotes/AN10210_2.pdf



M.C. Cheng will be receiving B.Eng. in Electronics Engineering from Universiti Sains Malaysia (USM) in April 2007.

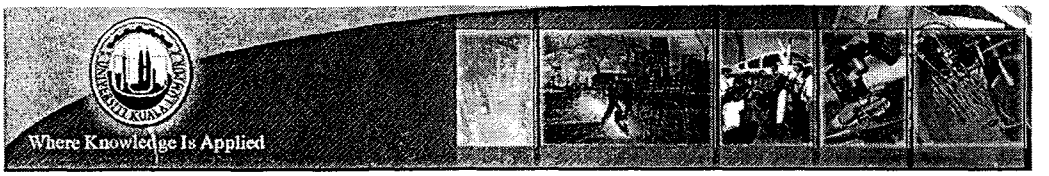
He received training in RTL development using FPGA from Intel, Penang in May 2006 and will be joining back to Intel as a System Validation Engineer after graduation.

Mr. Cheng Mun Chun has completed an R&D project in developing a new embedded processor using FPGA targeting for the new universal remote control market, and his present focus is in optimizing memory utilization for data acquisition systems.

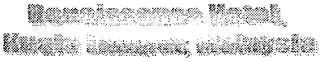
Z. Abdul Halim received B.Sc (Eng) and M.Sc (Eng) in Electrical & Electronic Engineering in 1997 and 2000 from USM.

In April 1997, she joined Test Department of Applied Magnetic Malaysia Sdn. Bhd. as a product engineer. She has been a researcher and lecturer for USM, School of Electrical & Electronic Engineering since April 2004.

Mdm. Zaini Abdul Halim's area of interest include digital circuit and hardware design using Microcontrollers, Microprocessor and FPGA. Her current activities have been focused on developing data acquisition system using FPGA.



11-13 December 2007



11-13 December 2007

Introduction

ICET 2007 is the first Engineering Technology Conference organized by Universiti Kuala Lumpur. This conference would be the main platform for researchers, academician, technologists and engineers to share and highlight their research findings particularly works that related to technology based research. Historically research has been focused on several fields e.g. pure science based research, engineering based research and social science research. Pure science research is considered the biggest group covering all types of research. Engineering based research has been well established to cater industrial needs and most of the research works were collaborated with industries to solve and improve their production yields. A new sub-group called technology based research has emerged Technology based research has similarity with other research groups. The difference is the aim of the research which focuses on the generation of new products and also build-up technopreneurs to commercialize the product.

The objectives of the conference are:-

• To bring together engineering technology expertise; professional from the industry, academia and government to discourse on research and development, professional practice, business and management in the engineering technology fields.

• To enable interdisciplinary collaboration between engineering technologists in the academic and industrial fields as well as networking internationally

Scope of topic

The ICET2007 focus on the following fields:

- Manufacturing Technology
- Advanced Materials
- Aviation
- Automotive & Marine
- Biotechnology
- Electrical/Electronics/Telecommunication & Networking
- Engineering Technology: Pedagogy and Training
- Information and Computing Technology
- Image Signal Processing Automation
- Mechatronics /Automation & Robotics

Organized by:

UNIVERSITI KUALA LUMPUR

In Collaboration With:

University La Rochelle

IPFW, USA

Institute Teknologi Bandung

Supported by:

MARA

TOURISM MALAYSIA

Visit Malaysia 2007

MeCD

The Development of an Inexpensive Portable Potentiostat

Z. Abdul Halim[a], O.Sidek[b], M.ravichandran[c]
School of Electrical&Electronic Engineering, Universiti Sains Malaysia Engineering Campus, 14300 Nibong Tebal, Malaysia
[b] Collaborative Microelectronic Design Excellence Centre (CEDEC), Universiti Sains Malaysia Engineering Campus, 14300 Nibong Tebal, Malaysia
[c]Department of Medical Microbiology and Parasitology, Health Campus, Universiti Sains Malaysia, 16150 Kubang Kerian, Kelantan, Malaysia.

Abstract: This paper presents a design and construction of a small, simple and inexpensive programmable potentiostat. The proposed potentiostat system consists of a signal input part including digital analog converter, a signal detection part including transimpedance amplifier based circuit and 8 bit analog digital converter and a signal control part including 89C52 microcontroller. The system is cost effective especially for individual or small group experiments. It is also suitable for independent student project involving field portable electrochemical instrumentation. The system has been tested using potassium ferricyanide. Data management are developed using Visual Basic software. Data are collected every one second and can be visualized in Microsoft Excell for further analysis.

1. INTRODUCTION

Potentiostats are widely used in electroanalytical techniques to identify, quantify and characterize redox active species including inorganic, organic and biochemical spesies [1]. Some examples of electroanalytical methods requiring potentiostatic control of the experiment include analysis of corrosion, materials properties and in vivo detection of biologicals such as glucose and catechol amines.

A potentiostat is an electronic instrument capable of imposing electrical potential waveforms across a working electrode relative to a reference electrode [2]. These electrodes are contained in an electrochemical cell. The working electrode is the electrode where the potential is controlled and where the current is measured. The reference electrode is used in measuring the working electrode potential. A reference electrode should have a constant electrochemical potential as long as no current flows through it. In order to maintain a constant potential over all conditions, a third electrode which is called counter electrode is used to conduct current into and out of the cell. This current has to exactly balance the current generated at the working electrode.

The equivalent circuit of the three electrodes system in electrochemical is shown in figure 1. The current generator responds to the analyte chemical reacting at the working electrode. The capacitor represents the double layer capacitance, which depends on the analytes. The resistor at counter electrode represents the solution resistance between the counter electrode and the current source. There is another resistor in series with the reference electrode, although the reference electrode does not generally conduct current.

With this background, it shows that three electrodes electrochemical sensor has the following properties: it has to maintain a fixed potential between working electrode and reference electrode, it is bipolar and will operate regardless of whether the current flows to or from the working electrode and it must measure the current from working electrode, delivering a usable signal to an output terminal.

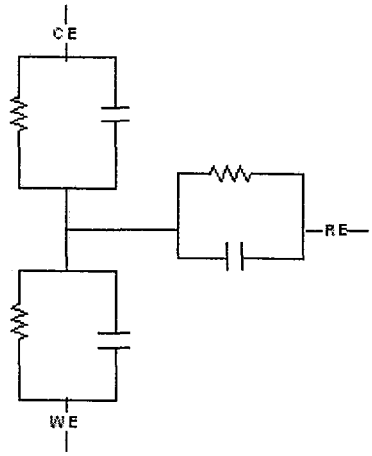


Figure 1: Equivalent circuit of the three electrode cell

4th International Colloquium on Signal Processing and its Applications, March 7-9, 2008, Kuala Lumpur, Malaysia.

ISBN: 978-983-42747-9-5

2. POTENTISTAT CIRCUIT

The potentiostat circuit consists of two parts: control circuit with bias voltage and current measuring circuit. The potentiastat circuit is shown in figure 2. The control circuit provides current to the counter electrode to balance the current required by the working electrode. Operational amplifier, OPA380A with the input bias current 50pA, offset voltage 25μV and power supply range 2.7V to 5.5V is the basic component in this circuit. The inverting input of the operational amplifier is connected to the reference electrode and must not draw any significant current from the reference electrode.

When the circuit is turned on, the IC2 provides current to maintain the working electrode at the same potential as the reference electrode. A bias voltage is applied on the non-inverting of the IC2. Circuit stability and noise reduction in the control circuit relies on R1, R2, C1 and C2 as shown in figure 2.

The current measuring circuit is a single stage operational amplifier (IC1) in a transimpedance configuration. The current is reflected across R4, generating an output voltage. The measuring circuit uses a combination of the load resistor R_{load}, internal operational amplifier resistance and the internal operational amplifier capacitance to establish an RC circuit. This RC circuit affects both the rms noise and the response time; the response time increase linearly with increasing R_{load} resistance while noise decrease rapidly with increasing R_{load} resistance.

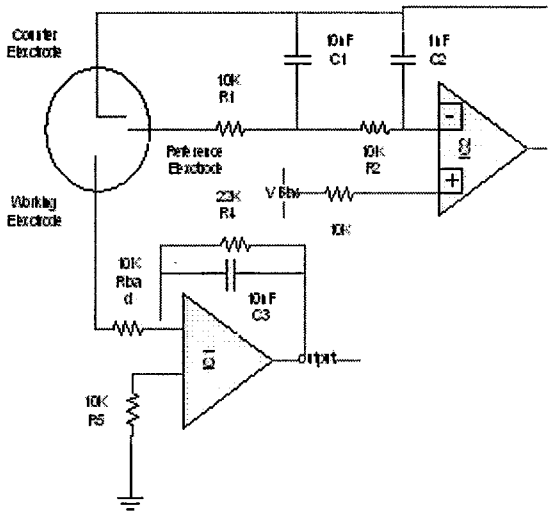


Figure 2: Potentistat circuit with input bias voltage.

3. THE PROCESSING UNIT

This unit performs the sensor data acquisition, to capture the data and display in Microsoft Excel format for further analysis. The processing unit consist of two parts, hardware and data management.

The main part of the hardware is shown in figure 3 [4]. The Philips 89C52 microcontroller with built in UART (Universal Asynchronous Receive Transmit) buffer is used as the embedded processor, based on its low power operation, cost effectiveness, programming features and extensive support base. Other components are 8 bit analog digital converter (ADC0809) that is required to convert the analog data from potentiostat to digital data, an RS232 port for communicating with PC and a real time clock (RTC) for use during data logging mode. The summary of list components is shown in table 1.

The embedded software are coded in assembly language to perform the data acquisition, serial communication with a PC and other house keeping task. These are relatively straightforward routines.

Data will be sent to PC in one data frame format that consist of one byte start of data, 2 bytes sensor's data and one byte end of data. The two bytes sensor's data are input bias voltage and output voltage from working electrode.

Graphical user interface is programmed using Visual Basic Software. The serial port is enabled every one second to capture the data. Time and date also are recorded and data can be saved in Microsoft Excel format for further analysis. In Visual Basic software routine, the data frame is waited until it receives the start byte of data. Once the start byte is received, the software start capture the data until it get the end byte of data. The output current is calculated by using output voltage and the value of resistor R4 as shown in figure 2.

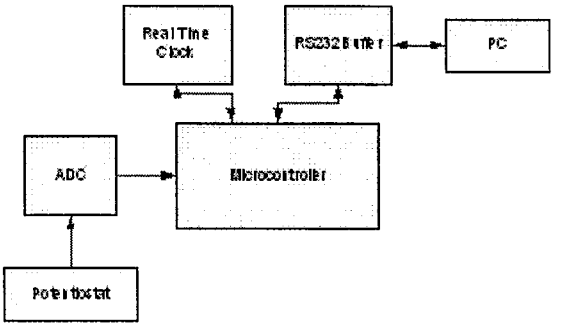


Figure 3: Block diagram of processing unit.

Table 1: A list of the components

4th International Colloquium on Signal Processing and its Applications, March 7-9, 2008, Kuala Lumpur, Malaysia.

ISBN: 978-983-42747-9-5

No	Device	Description
1	8952A	Intel's 8 bit microcontroller
2	Max232	To convert RS232 voltage level to TTL voltage levels and vise versa
3	Max660	Voltage converter
4	ADC0808	Analog digital converter
5	OPA380A	Operation Amplifier

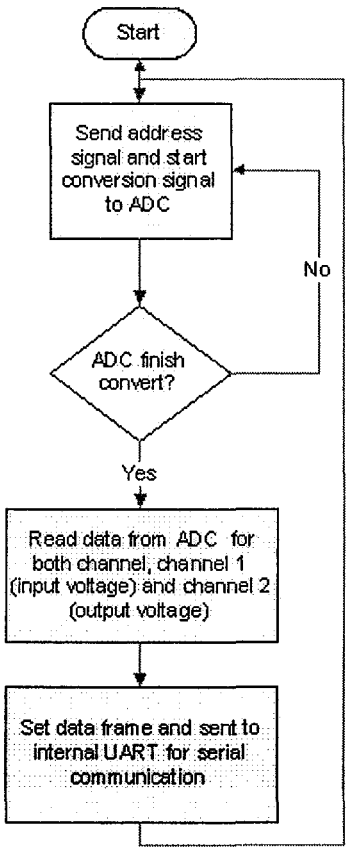


Figure 4: Flow chart to read data from analog digital converter and send the data to PC.

Figure 4 shows the flow chart to read data from analog digital converter and send the data to PC. ADC0809 has an 8 input channels and address are sent using microcontroller to read the right channel. Figure 5 shows the flow chart to read data from microcontroller and display the data in Microsoft Excell every second. Data are display in channel 1, channel 2 and channel 3 that represent input voltage, output voltage and output current respectively. Figure 6 shows the prototype of the potentiostat circuit .

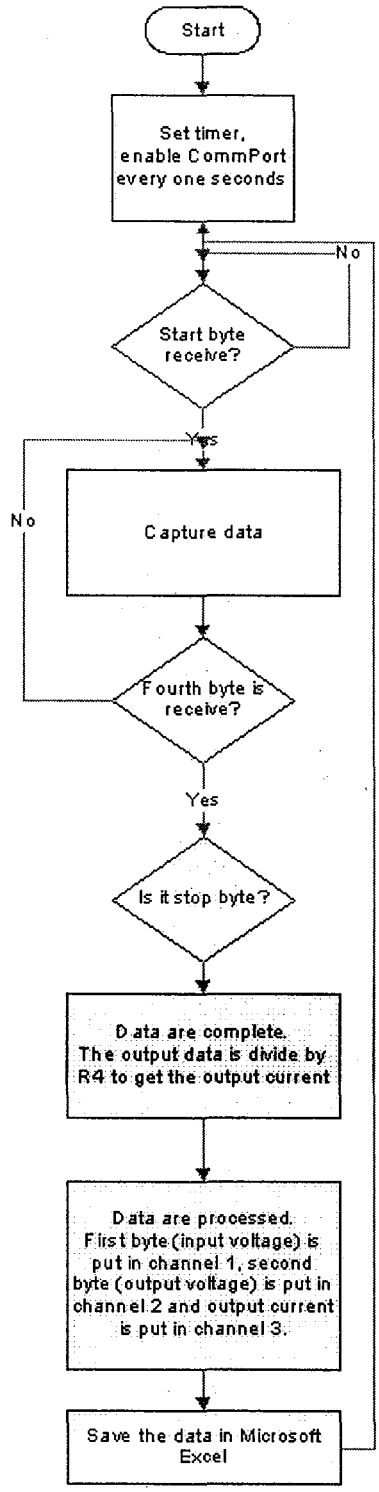


Figure 5: Flow chart to read data from microcontroller and display the data on PC.

ISBN: 978-983-42747-9-5

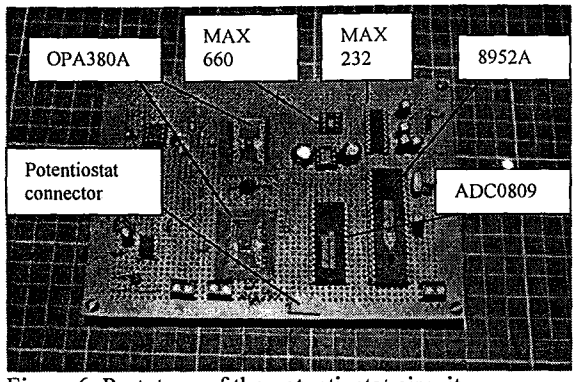


Figure 6: Prototype of the potentiostat circuit

4. EXPERIMENTAL

The input bias voltage is connected to noninverting terminal of operational amplifier, IC2 as shown in figure 2. The output pin of IC1 is connected to 8 bit analog digital converter that is controlled by microcontroller. For testing the potentiostat, potassium ferricyanide solution ($K_3F_6(CN_6)$), magnesium sulphate ($MgSO_4$) solution and natrium choloride (NaCl)solution are prepared.

Three electrodes system is used [3]. Working electrode, and counter electrode are designed using carbon material and reference electrode is designed using silver/silver chloride (Ag/AgCl). These electrodes are disposable electrodes.

The input bias voltage comes from voltage supply and it is increased by 0.2V every one second. The input voltage is connected to the amplifier IC2 and the output of amplifier is connected to counter electrode. This satisfies the basic requirement of a counter electrode, which needs to be able to provide any amount of current required by electrochemical activity at the working electrode.

The electrochemical current generated at the working electrode is converted into an equivalent voltage by operational amplifier IC1 which is a current to voltage converter and the voltage is measured using microcontroller circuitry [5]. The reference electrode is connected to the inverting terminal of voltage follower circuit. The voltage follower circuit helps in maintaining a constant potential at the reference electrode without drawing much current.

The input voltage (from power supply) and the output voltage from voltage follower circuit are connected to analog digital converter. Microcontroller will read the signal and send the data in data frame format to PC for further analysis. The data frame consists of a start byte, data from input voltage, data from output voltage and end of data. In Visual Basic program, the input voltage is put in channel 1 and the output voltage is put in channel 2.

Table 2: Result Summary

Solution	Experiment	Input Voltage (V)	Output Voltage (V)
$MgSo_4$	1	5.3955	0.10986
	2	5.4541	0.10009
	3	5.4785	0.14892
NaCl	1	5.4785	1.26953
	2	5.5371	1.24999
	3	5.50292	1.25976
$K_3Fe(CN_6)$	1	5.41014	0.87646
	2	5.45898	0.65185
	3	5.41014	0.86669

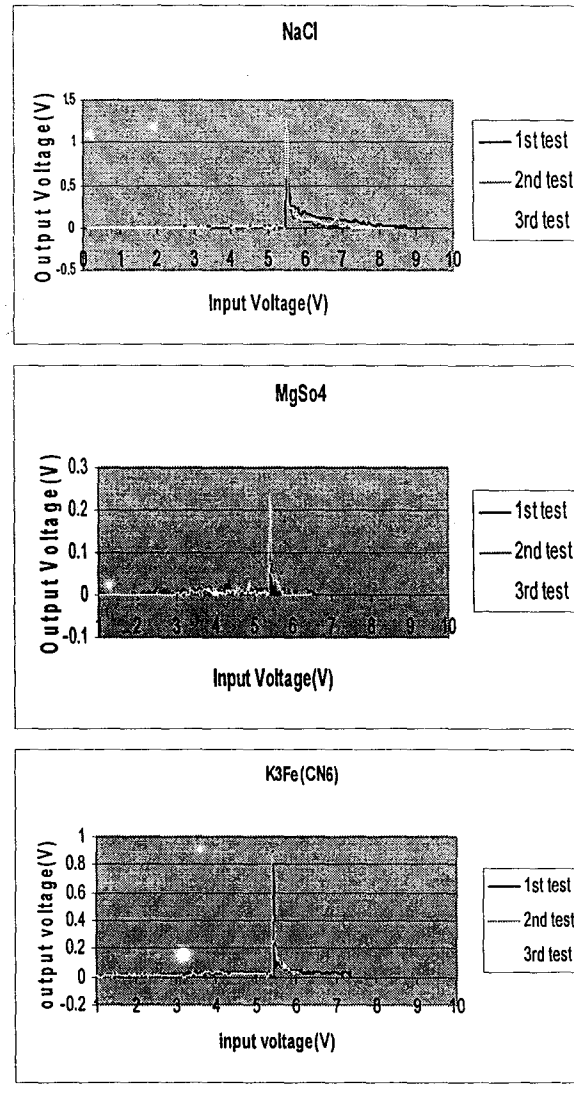


Figure 6: Output voltage for $MgSO_4$, NaCl and $K_3Fe(CN_6)$

ISBN: 978-983-42747-9-5

Data are tested three times for every sample to ensure that the circuit is stable. Results summary are shown in table 2. In $MgSO_4$, NaCl and $K_3Fe(CN_6)$, the redox activity occurs within the following input voltage; 5.3955V to 5.4785V, 5.4785V to 5.5371V and 5.41014V to 5.45898V respectively. It also shows that there is no significant difference in output voltage for all experiments. It indicates that the circuit is stable.

5. CONCLUSION

The potentiostat built in this work can detect an electron transfer in redox process. Results show that the circuit is stable since there is no significance difference in output after being tested three times continuously. This potentiostat is easy to build and inexpensive since it only requires a low cost components. The power requirement is minimal and can be supplied by a 5 volt battery, making this potentiostat field-portable.

References

[1]. A.V.Gopinath and D. Russell, "An Inexpensive Field Portable Pragrammable Potentiostat", Chem Educator, 2006. pp 23-28.

[2]. J.Wang,"Portable Electrochemical Systems", Trends in analytical chemistry, vol. 21, no. 4., 2002, pp 226-232.

[3]. S.Cho, n.Seong and J.J.Pak, "Development of a Cheap and Portabble Sensing System for Electrochemical Genotyping", 0-7803-7454-1/02 @2002 IEEE.

[4]. M.A.Mazidi," The 8051 Micrcontroller and Embedded System", Prentice Hall, 2000.

[5]: K.R.laker, " Design of Analog Integrated Circuits and Systems", McGraw-Hill, 1994.

4th International Colloquium on Signal Processing and its Applications, March 7-9, 2008, Kuala Lumpur, Malaysia.

ISBN: 978-983-42747-9-5

Universal Infrared Receiver For PC's Media Player

Z.Abdul Halim and Y.H.Ying
School of Electrical and Electronic Engineering, University Sains Malaysia,
Engineering Campus,
Seri Ampangan, 14300 Nibong Tebal, Seberang Perai Selatan,
Penang, Malaysia.
Tel: +604-5995999 ext 6061
E-mail: eezaini@eng.usm.my

Abstract:Infrared data transfer is becoming increasingly more important. Television remote control uses infrared light, so do PC mice, keyboard, printers and other peripherals. This is because the infrared devices offer lower production cost, wide operating range and good communication security. The PC infrared receiver presented in this project was a practical application that represents a continuation of the series of articles on PC interface and Visual Basic. Visual Basic 6 was used to decode the received infrared signal and perform PC's functions such as window media player. Serial port DB-9 (COM1) was used as the interface between computer and the hardware of this project. By end of this project, total three brands of remote controls were able to use to remote the PC such as Philips, Sony and JVC.

1. INTRODUCTION

If a PC in a living room was installed as a television or as a part of hi-fi setup, a remote control was practically indispensable. Who wants to trouble of going to the PC's keyboard to adjust the volume or change channels? Unfortunately, PC's do not offer built-in infrared receivers compatible with ordinary transmitter.

A low cost wireless PC's remote control has been designed in this project since it is convenience for the user. A normal TV or DVD remote control in our house can be used in this application. So the user can use the existing remote control to control PC's media player from up to 30 feet away from their PC or lap top [1], [2].

There were several steps involved in getting a PC to react to the pulses of infrared sent out by the transmitter. Hardware was required to receive the pulses and process them into a form suitable for decoding. The receiver was designed to work using COM1 port. This has many advantages: it keeps the hardware to a minimum (for example, no level shifters are required); power can be drawn from the PC's COM1 port and removing the need for an external power supply.

No software was needed to be installed on the computer, making the circuit completely independent of operating system, working equally well under Windows, Linux or DOS.

2. INFRARED REMOTE CONTROL'S PROTOCOL

Different manufacturer was using different IR coding. Three remote controls have been used in this project. They are Philips RC 0764/01B, LG 6710V00090A and Sony RM-S343. Philips and LG remote control has used RC5 code which is the most widely used coding method. One bit of data is represented by two half bits. A low/high combination of these bits indicates a data '1' whereas a high/low combination indicates a data '0'. The length of each bit is 1.778ms and a complete message is 24.889ms long.

Sony RM-S343 has used SIRCS or CNTRL S protocol that consists of 12 to 20 bits. A '1' is represented by a 1.2ms ON followed by a 0.6 ms OFF. A '0' is represented by a 0.6ms ON and 0.6 ms OFF. Logic '1' and logic '0' for every remote control is summarized in table 1. It shows that different manufacturer of remote control has used different time length of logic '1' and logic '0' [1]. Consequently these time length will be used in Infrared Data Decoder.

Table 1: Time length for remote control

Remote Control	Bit '1'	Bit '0'
Philips RC 0764/01B	0.889ms on follow by 0.889ms off	0.889ms on follow by 0.889ms off
LG 6710V00090A	0.889ms on follow by 0.889ms off	0.889ms on follow by 0.889ms off
Sony RM-S343	1.2ms on, 0.6ms off	0.6ms on and 0.6 ms off

3. HARDWARE OF IR RECEIVER

The schematic diagram of IR receiver is shown in figure 1. It consists of two main parts; power supply and IR receiver. The supply voltage was taken from the DTR and RTS output of the RS 232 interface, which were connected together via the isolating diodes D1 and D2.

A voltage around 10V can be activated using a program running on the PC. Since high pulse current is needed for transmitting, a relatively large electrolytic capacitor (C1) was used to smooth the input voltage of the voltage regulator.

Infrared receiver, Sharp ISIU60 was used in this project. When the IR receiver received and infrared signal, it produced an output signal on its middle pin with an active low level. These output pulses were connected directly to the CTS lead, where they must be decoded using software. A supplementary pull up resistor was necessary here, since CTS lead has a relatively low input resistance.

ISBN: 978-983-42747-9-5

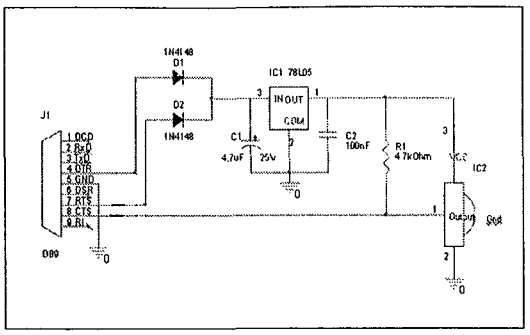


Figure 1: Schematic of the IR receiver

Besides of the IR receiver a regulator 78L05 was needed to regulate the supply voltage from the DTR and RTS output of the RS232 interface. It was because only 5v supply voltage was needed to turn on the IR receiver while the voltage from the serial interface was around 10V and this voltage may damage the IR receiver. So the regulator was used to regulate the voltage supply from 10V to 5V to protect the IR receiver [7][8][9].

Apart from the two main components, another two capacitors (4.7μF, 100nF), one resistor (4.7kΩ) and two diodes (IN4148) were used to build the hardware. The 4.7μF capacitor was used to smooth the input voltage of the voltage regulator while the two isolating diodes were used to protect the whole receiver circuit [5][6].

Hardware of the IR receiver is shown in figure 2. Remote controls of Philips RC0764/01B, LG 6710V00090A and Sony RM-S343 were used to test the functionality of the hardware.

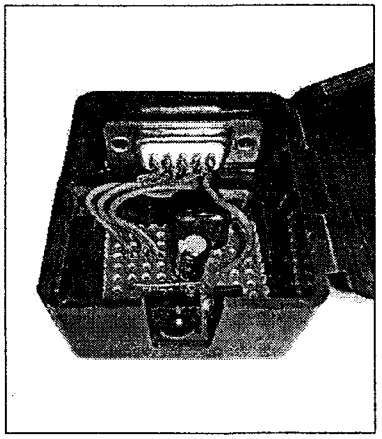


Figure 2: Hardware of the IR receiver

4. SOFTWARE DEVELOPMENT

Hardware was required to receive the infrared pulses and process them into a suitable form to read by computer. But to use the receive signals from the remote control to remote the media player, a program were written to decode and recognize which button has been pressed on the remote control. Three brands of remote control were used in this project, so three different routines of program were used to figure out the brand of the received infrared signal first. After decoding the received signal, few more programs were developed to let the user control their favorite computer programs.

The data output of the receiver IC was connected directly to the CTS lead. Signals from a remote control unit that have been demodulated by the receiver IC thus appear on this lead. A program for decoding the signal only has to evaluate the incoming pulse in order to recognize which button has been pressed on the remote control.

This task is relatively time critical and requires the use of REALTIME = TRUE in Visual Basic program [3],[4]. This routine initially waits for a low level signal which acts as a start pulse. In order to prevent the PC loops from hanging in an infinite loop if no signal is present, a timeout condition is built in. If no signal has been received after 200ms, the program terminates with an error message. After the start sequence, the individual bits are read.

If a '0' level is read at the start of the routine, this should represent a 0 bit. Next the routine waits for the signal level. After half the pulse width pulse (444μs for Philip's remote control), a new query is made to see whether the same level is present. If this is not the case, an error is detected and all data read up to this point are overwritten with the value -1.

If a '1' level is read at the start of the routine, this should represent a 1 bit. Next the routine waits for the signal level to change. After half the pulse width pulse, a new query is made to see whether the same level is still present. If this is not the case, an error is detected and all data read up to this point are overwritten with the value -1. After slightly more than 444μs, the routine returns the value of the bit that was read. Signal reception is controlled by a timer in the visual basic program and all received data are displayed in a text window. Same routine can be used to decode other brand of remote controls, such as Sony and JVC but the time delay is change from 444μs to other values depends on the perspective protocols.

Flow chart of program initialization is shown in figure 3. Once the program was start, it will go to the Setting.ini file to load back the last saved remote control brand and data.

After that it will load back the previous saved song list from songlist.txt and songname.txt file. Main form has divided into three tabs, which were setting, player and power point. If the exit button was pressed, then whole program will be closed. Print screen of the universal infrared receiver form is shown in figure 4.

In setting tab, there were another three features, which were brand selection, teaching and real time signal decode. When a new brand is selected, the new data will overwrite the old data and saved into the Setting.ini file. The user can start to teach the program when the teaching button is pressed. Once the button of active IR is clicked, the user can check the functionality of the remote control and the command code of the button pressed. The flow chart is shown in figure 5.

ISBN: 978-983-42747-9-5

The purpose of the remote teaching is to make this project to be more users friendly. The user can assign whatever button on the remote control to replace the PLAY button to perform the function of play. In teaching mode, the entire assigned keys were stored in an array. After the user press the button which they would like to assign for function of play, the data of the received signal will be decode before save in the array. The message of "INVALID button Assign, setting No Saved, Please Re-teach Again", will be pop out and the user need to re-teach again if the user still haven't teach the button after 5 second. After all keys have been teach, the program will check through every data in the array to make sure that there is no duplicate key has been touched. If it is found the same button was assigned for two different functions, then a message box of "DUPLICATE Button Assign, setting NO SAVED. Please Re-teach Again" will be pop out and the user needs to re-teach again for the duplicate key. This routine helps to avoid confliction inside the program.

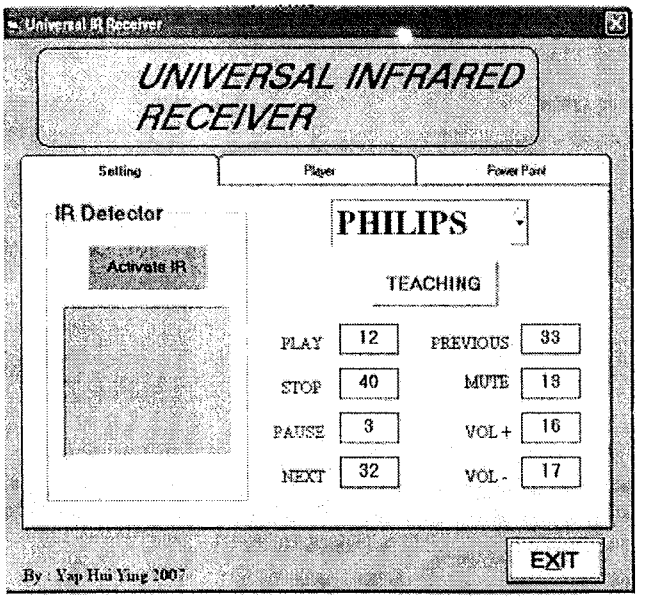


Figure 4: Print Screen of the Infrared Receiver Form

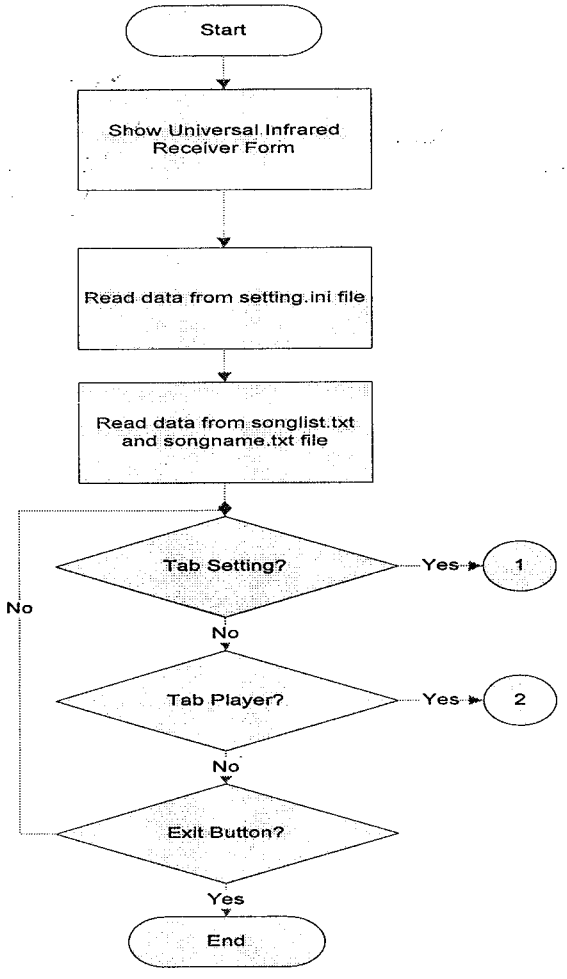


Figure 3: Flow chart of program initialization

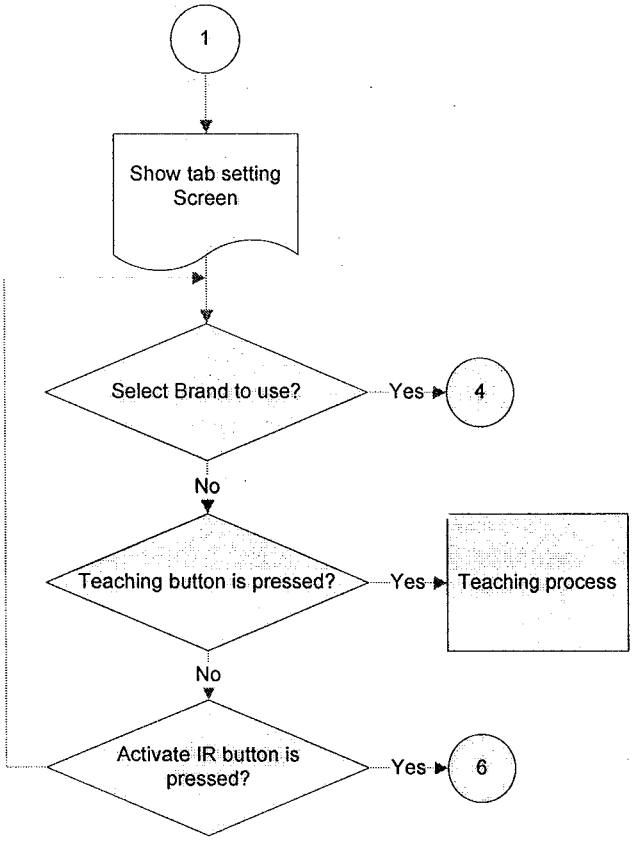
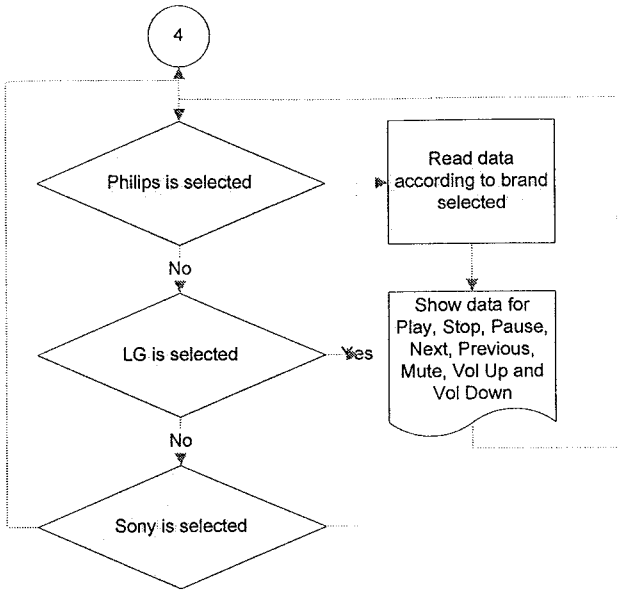


Figure 5: Flow chart of the setting tab



igure 6: Flow chart of the brand selected

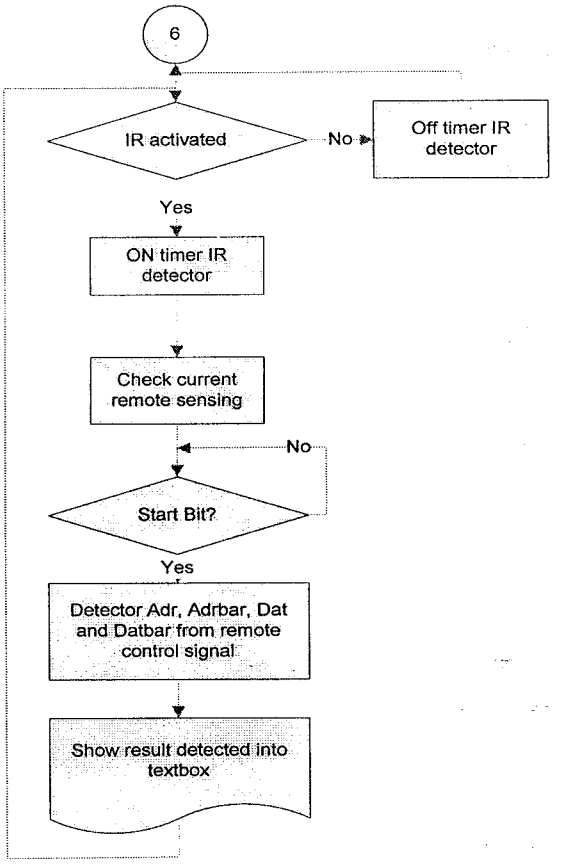


Figure 7: Flow chart of real time signal code

5. MEDIA PLAYER APPLICATION

Flow charts for media player application are shown in figure 5, figure 6, figure 7 and figure 8. This application was not only able to play song in three different file types, which were mp3, wav and wma but it also can be used to play a video clip. Its functions were almost the same with the Winamp which was very popular and widely used nowadays. The user can either control this application with PC mice or remote control.

In this application, six main buttons were used. PLAY button was used to play the song while STOP button was used to stop playing the song. If NEXT button is pressed, it will go to the next song in the song list while PREVIOUS button will play back the previous song in the song list. PAUSE button was used to temporarily stop playing the song, it continue to play the song from where it stop just when the PLAY button was pressed. The user can mute the sound by pressing the MUTE button. Besides, the user can control the volume by scrolling the volume bar or using button VOL+ and VOL- on the remote control. The flow chart is shown in figure 8.

Component media player must be added into the program so that this application manages to play song and video clip. To make this player can be played songs one by one automatically; a play list was created in this player to add in all the songs. Another sub application needs to be created so that the song files which added into play list can be be chosen. There are two song lists inside the player, named Lst1 and List2. List 1 was used to record the location of the song while List2 was used to show the song name to the user. All the information of the song will be added into List1 and List2.

All files will be filter first before added into the song list. Only files with format *.mp3, *.wav, *.avi, *.wmv and *.wma can be added into the song list. Once the song was added, user can play the song either by double clicking on the name in the song list or by clicking the PLAY button. After click on the Active IR button, users can remote the player with the remote control. PLAY button on the remote control was used to play the songs. Not only PLAY button, user can assign any button to play the song by using teaching mode program. Figure 9 shows the print screen for application of playing mp3 and figure 10 shows the print screen for application of playing a video clip.

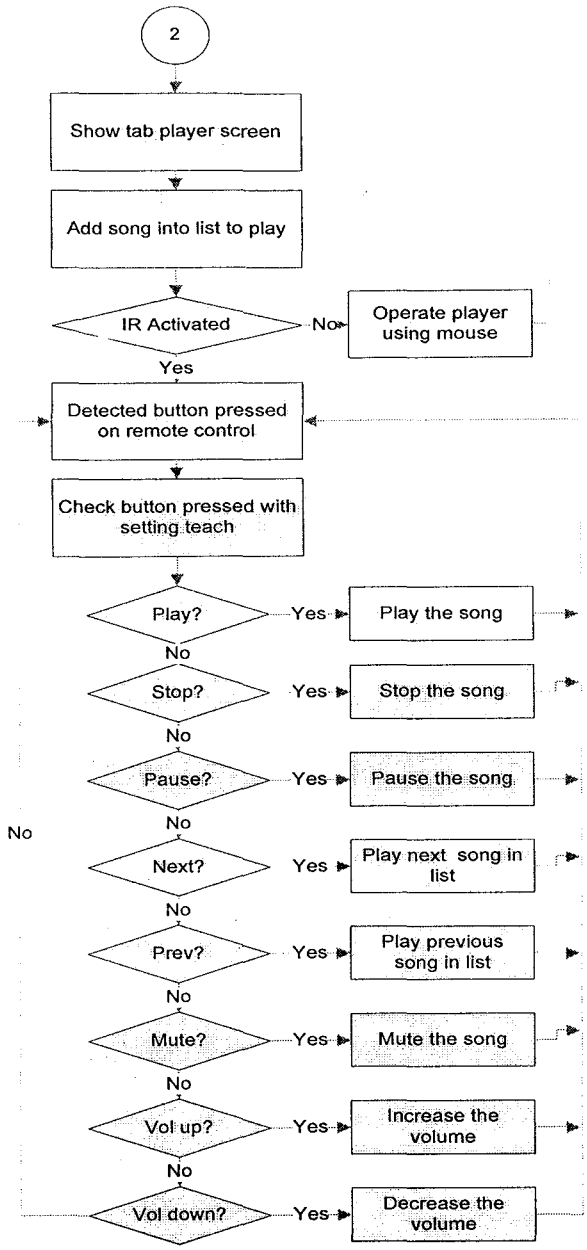


Figure 8: Flow chart of the media player application.

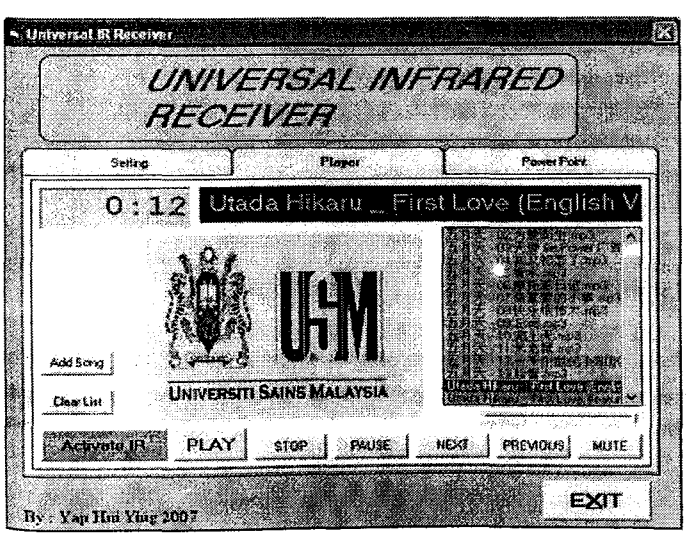


Figure 9: Print screen for application of playing mp3.

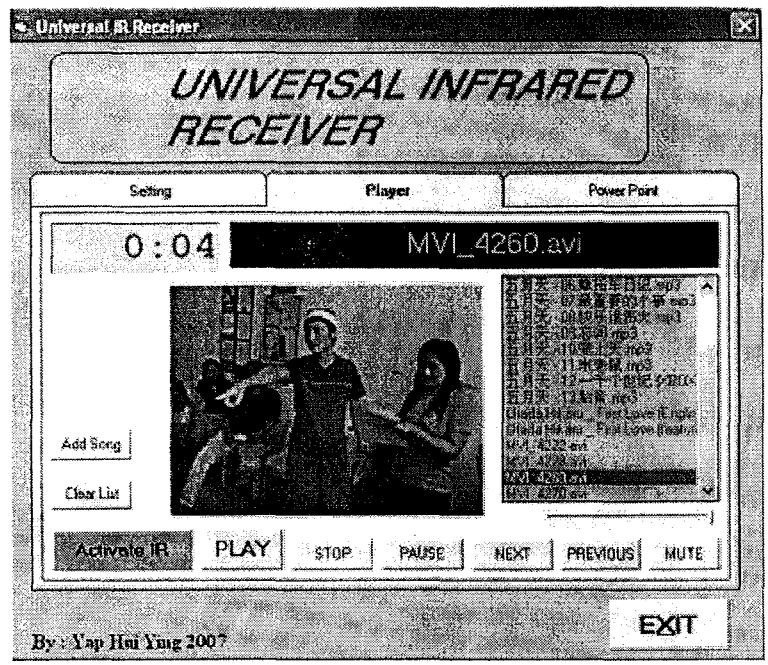


Figure 10: Print screen for application of playing a video clip.

6. CONCLUSION

The project of universal infrared receiver for PC media player has been presented. The system can be implemented without any external power supply, so it makes the circuit completely independent of operating system, working equally well under Windows, Linux or DOS. Besides it able to have some functions with other product in the market nowadays but with a much lower cost.

References

[1]. Other AN., (2001 March), "IR Remote Control Codes (1) format, protocols and (in)compatibility", *Elektor Electronics*, pg 50-54.

[2]. Bainka B. and Hans-Joachim Berndt, (2000), PC Interfaces under Windows, Measurement and control using standard ports, Prentice Hall.

[3]. Chua Chooi See, (2001), A Step-By-Step Guide Visual Basic 6, Federal Publications.

[4]. Michael Ekedahl and Wiliam Newman, (2000), Programming with Visual Basic 6, Course Technology.

[5]. Kainka B., (2001 January), "Infrared Transceiver for the PC (1)", *Elektor Electronics*

[6]. Kainka B., (2001 Febuary), "Infrared Transceiver for the PC (2): RC5 transmitter and data link", *Elektor Electronics*

[7]. Kainka B., (2000 September), "PC Serial Peripheral Design (1)", *Elektor Electronic*, pg 12-15

[8]. Kainka B., (2000 October), "PC Serial Peripheral Design (2)", *Elektor Electronic*, pg 52-55

[9]. Kainka B., (2000 November) "PC Serial Peripheral Design (3)", *Elektor Electronic*, pg 62-65

FPGA Implementation on Temperature Data Logger Using SHT75 Temperature Sensor

Z.Abdul Halim[1],L.C.Chaw[1]

[1] *School of Electrical&Electronic Engineering Universiti Sains Malaysia*
Universiti Sains Malaysia, Engineering Campus
14300 Nibong Tebal, Seberang Perai Selatan, Pulau Pinang, Malaysia
Tel: +604-5995999 ext. 6000, Fax: +604-5941023, E-mail: eezaini@eng.usm.my

Abstract

The temperature data logger is used extensively in industries such as food processing, manufacturing, printing and metallurgy where critical process variables (temperature, pressure, etc.) may adversely affect the results of the process. In this paper, a temperature data logger using FPGA technology has been designed. The system has been implemented on FPGA since FPGA is a programmable device and the code can be changed depending on the application without changing the hardware part. The steps of design using state machine and the implementation of a temperature data logger using VHDL code has been discussed in this paper. The system is fully controlled by a FPGA processor (model XC4010PC84) with the use of Xilinx Foundation Series 2.1i software. In the first stage, the temperature is measured by SHT75 sensor. The data from the sensor is then processed by the FPGA and the temperature value until one decimal point will be displayed on LCD screen.

Keywords:

FPGA,Temperature data logger, Temperature sensor SHT75, State Machine.

Introduction

Almost every event and process in our daily life is influenced by the temperature. For example 100°C is needed to boil up drinking water while 0°C is needed to freeze the water. In the chemical field, it is observed that some chemical reactions happen faster under certain temperature condition. In the study of physics, it is found that velocity of electrons inside a volume increase with temperature. Thus it comes to an idea of designing a temperature control and optimization system. This temperature control and optimize system have been developed and commercialized in market intensively such as temperature chamber, temperature monitoring system and so on.

Undeniable, this system could be realized and developed by using different technologies and software. For past few decades, microprocessor and microcontroller are the leading technology in the embedded system design [1][2]. However, this technology has its limitation in terms of cost, capacity and size of the device. Recently, FPGA technology has become popular and prominent in the field of ASIC and embedded system design. FPGA is chosen because it provides a high density ASIC design that allows user to re-configurable the hardware without changing its physical structure [3]. It also allow the implementation of system to do the operation for the millions logic gate. Furthermore, this design style provides a fast prototyping and also for cost-effective chip design, especially for low-volume applications [4].

There are three mains parts in this project, which are temperature sensor, 2 x16 line LCD display and a Xilinx FPGA board. Before the SHT75 sensor starts function, a start sequence and a measurement command will be issued to it. After that it will continuously sense the temperature data in 16 bit form until the system is reset. The frequency clock that used in this project is 259khz. However the value of frequency will not reduce or increase the speed of the design significantly because the time for each temperature measurement is around 55 ms. This measurement time is far longer than the minimum time needed for LCD to display the temperature value. Thus it is estimated that this system could measure the temperature for 17 times in one second.

After each measurement, the 16 bit data will be processed by the FPGA system. In this system, the arithmetic operation for binary number is required because the processing of the temperature data involves multiplication of constant in floating point. Besides, binary to decimal and decimal to ASCII code converting are also performed in this stage.

The final stage is LCD display implementation. There are two topology of implementing the LCD in VHDL code. The first topology is by using the delay methods where a sufficient delay is provided for each send or write data to be latched in the LCD. The second topology is by monitoring the busy flag which is bit D7 in LCD for each read or write command. The bit D7 will be in logic 1 if it is busy while it will be logic 0 when it is in the rest state. The rest state means that it is ready for new data or command to be read or wrote. In this project, the first topology is used because it is easy to implement just by referring to the minimum latch in time for ach command or data to be performed. The system has been implemented on FPGA board, using XC4010PC84 device [5].

Design Methodology

The design can be divided into three main parts and each part

represents a module that describes each sub-system. These three modules will be interfaced to form a main entity that represents the whole system. The first module is the design of sensor interface, the second one is the design of data processing for sensor and the last module is the design of LCD interface.

Design of SHT75 Sensor Interface

The SHT75 sensor has 4 pins. Pins 2 and 3 are connected to the power supply and ground respectively. A high level schematic of the SHT75 sensor interface is shown in figure 1. The sensor needs a voltage supply between 2.4Volt and 5.5 Volt. Pin 1 is connected to serial clock (SCK), a clock signal generated by the FPGA to synchronize the communication between FPGA and the sensor. There is no minimum frequency of SCK as the interfacing is completely consists of static logic. Pin 4 is connected to the FPGA through the serial bi-directional data line. This serial data bus is used to transfer data in and out of the sensor.

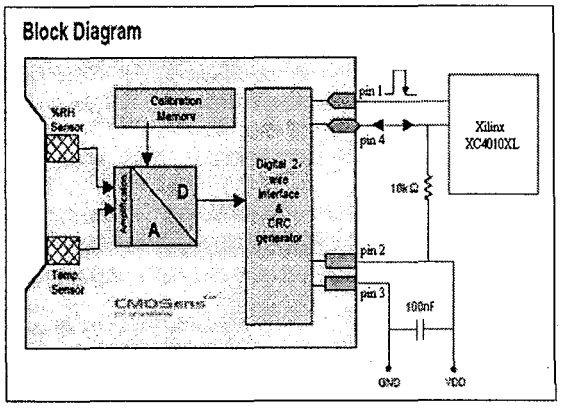


Figure 1: High level schematic shows SHT75 sensor interface.

As shown in figure 1, a pull up resistor is connected from the data bus to the power supply voltage Vdd. The data changes after the falling edge of SCK and is valid on the rising edge of SCK. The data must remain stable while SCK is high. The data signal must not be driven high by the FPGA. The external pull-up resistor pulls the value on the data line high when required. When a value 0 is written to the sensor, the FPGA pulls down the value to 0. The value of the pull up resistor is determined from the DC characteristics of the sensor.

The entire communication between the sensor and the FPGA is done using DATA and SCK line. In order to read data from the sensor, firstly, the transmission start command is given to the sensor by the FPGA processor. To initiate a transmission, the FPGA will send a logic 0 while SCK is high as shown in figure 2.

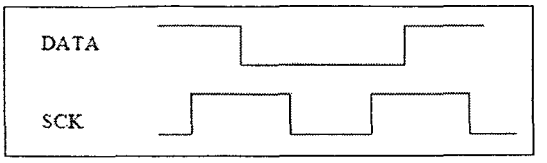


Figure 2: Transmission Start sequence of SHT75 sensor

After issuing the transmission start sequence the FPGA will send a command to measure temperature. The command that should be sent to the data line is "00000011". The data changes at the falling edge of the clock and is valid at the rising edge of the clock. After 8 SCK pulses, the FPGA should not drive the data line. At the ninth SCK pulses, the sensor will lower down the data line to zero. If the sensor does not bring the data line to zero, there is an error in the transmission and the transmission need to restart again by sending the command for measurement.

After sending the command for measurement to the sensor, the FPGA need to wait until the measurement is complete. The time needed for measurement depends on the type of measurement used. The SHT75 sensor supports 12 bit and 14 bit measurement. 14 bit measurement is used for more accurate measurements. In this design, 14 bit measurement is used and the time taken for wait state is approximately 210 ms. The SCK signal sent by the FPGA must be at logic low in wait state and the DATA line must be high in the wait state which means that the DATA line should not be driven in wait state. The sensor will end the wait state by pulling down the DATA signal to zero after 210 ms.

In measure state, the FPGA need to send the serial clock signal SCK to the sensor for two bytes of data measurement. The DATA line is pulled low by the FPGA, indicating to the sensor that one byte of data has been received. All the values received on the serial DATA line are MSB first.

Design of Data Processing Module

The purpose of this module is to convert the temperature data obtained from SHT75 sensor in binary form to the decimal form which is then interface with the LCD module. In this module, the input data is in 16 bit binary form. Some of arithmetic operations in floating point such as addition, subtraction and multiplications involve in data processing module.

Addition/Subtraction

In addition or subtraction, the exponents of the two numbers need to be compared. The significant of the number with the lesser exponent is shifted to the right until both the exponents are equal. A one bit right shift of the significant adds the value +1 to the exponent. Once both exponents are equal, the significant are added. The results of the addition/subtraction are normalized by incrementing or decrementing the exponent field of the result. The normalized form of a floating point number can be represented as 1.xxxxxX 2yyy, where x represents the significant field and y represents the exponent field.

Multiplication

In multiplication, the two biased exponents are added and the two significant are multiplied. These are generally done using binary adder and binary multiplier. Finally the product is normalized if needed, by shifting the exponent either right or left.

Design of LCD Module

Each time the temperature value has completely displayed on LCD screen, it will be cleared entirely and another new value of temperature will be displayed. Since the time used for clear screen is very fast, it is unable to see the clear process. In LCD module, there are signals which are called START (1 bit input signal) and STOP (1 bit output signal). These two signals are connected to RECEIVE (1 bit input signal) and SEND (1 bit out put signal) of data processing module as shown in figure 3. The SEND signal is connected to START signal of LCD module and the STOP signal is connected to RECEIVE signal of the data processing module. If the temperature data have been processed and ready for send, then SEND signal is logic '1'. If START signal is logic '1', then STOP signal is logic '0'. STOP signal is triggered to 1 after temperature value has been completely displayed. The data processing module will continue processing new temperature data if RECEIVE ='1'.

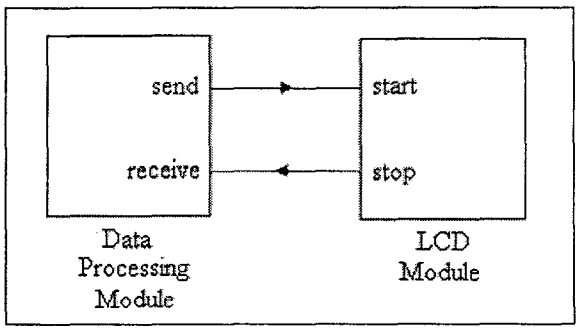


Figure 3: Interfacing between Data Processing Module and LCD Module.

Design Implementation

There are three modules to implement the whole system. The modules are HT_sensor, Arithmetic and LCD. Vhdl code is used to describe and implement the module based on the theory and methodology that has been discussed.

HT_Sesor Module

Block diagram of HT_Sensor Module is shown in figure 4. The tri state buffer is used for bidirectional communication between FPGA and the sensor. The output data will only send out data when the enable pin for the tri state buffer is in high condition. The state diagram for the design implementation is shown in figure 5.

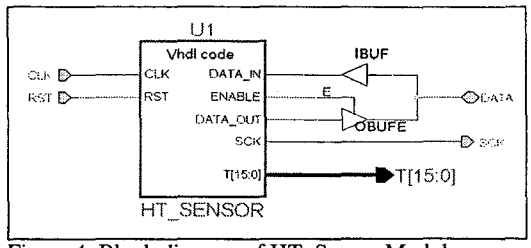


Figure 4: Block diagram of HT_Sensor Module

As shown in the state diagram in figure 5, state machine concept is used to implement the sub system. There are 10 states, which are boot, idle, start1, start2, start3, start4, start5, start6, write, waiting, wait and result.

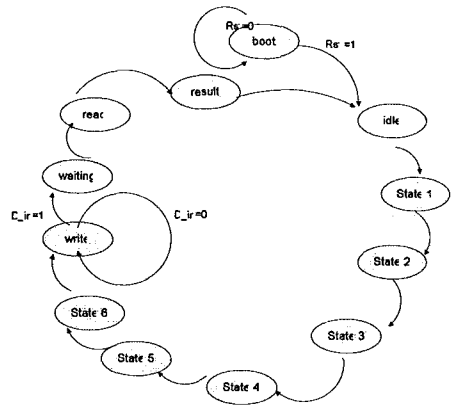


Figure 5: State Diagram For HT_Sensor Module

The initial state for the system is boot state. The sensor needs 11 ms to reach its "sleep" state after power up. No commands should be sent before that time. Thus VHDL code is written to create a 12 ms delay for the purpose. After the boot state, is the idle state, where command code for temperature measurement is loaded to P signal. Transmission Start sequence comes after idle state. The transmission start consists of 6 states from start1 to start6. To send data to the sensor through data_out, the enable signal is set to logic high.

After the transmission start is the write state, which is required to send the measurement command to the sensor which has been priory stored in the signal P. The next state is the wait state, which is used to wait for the complete measurement from the sensor. After SHT75 has complete the measurement, it will pull down the data line and enters idle mode. At this time, enable should be pulled down because data are now being received. The system will stay in the wait state if the signal from data_in is in high, else it will enter the next state which is the read state.

In the read state, the 14 bit temperature data are stored into a temporary register. After the the MSB data has been read, a low signal is sent by FPGA as an acknowledgement. This required the enable output to be high and data line is pulled down through data_out. After that enable signal output is

pulled down again so that the LSB data could be read through data_in. After reading process is completed, the enable signal output is pulled high again.
Result state is the state where the temporary signals are loaded into the output bus to interface with the arithmetic module . After the result state, the system will go to the idle state again and this loop will be continued until the system is reset.

Arithmetic Module

Block diagram of Arithmetic module is shown in figure 5.

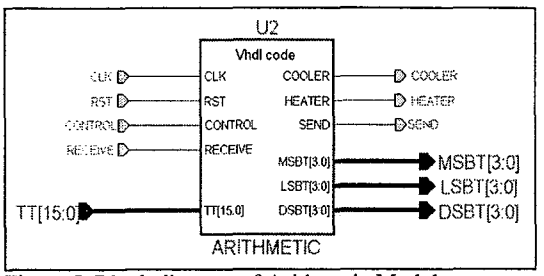


Figure 5: Block diagram of Arithmetic Module

State machine concept is also used to implement the arithmetic module [6]. There are 6 states in this design which consist of idle, start1, start2, start3, start4 and start5. The state machine of arithmetic module is shown in figure 6.

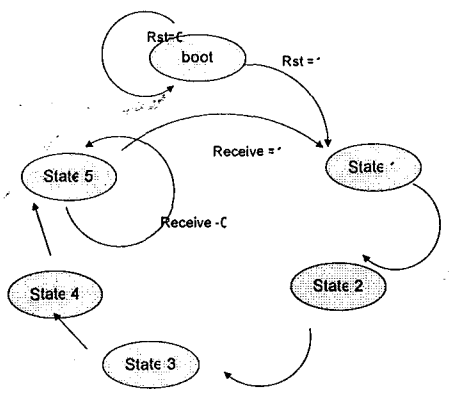


Figure 6: State diagram of Arithmetic Module

The initial state after reset is idle state. In this state, counter is used for each specific process. Count =1, temperature data from sensor is stored in TTT bus, For count from 2 to 4, these temperature data are process through the arithmetic operation. After processed, the data are stored in YY signal while the first decimal point is stored in 4 bit L bus. After the idle state, it comes to the start1 state. The binary data is converted to decimal number prior to the ACSII code conversion. After that it comes to start2 which MSB of the temperature data is determined. In state 3, the first decimal point will be determined. In start4 state, the physical temperature value in decimal form is loaded into the output bus and interfaced with the LCD module. After the physical temperature value is being sent, it will enter into start5 state. In this state, RECEIVE =1 if the LCD module has completely displayed the current temperature data, else it will always stay in the start5 state. When RECEIVE=1, the system will go back to the idle state during the next clock cycle.

LCD Module

Block diagram of LCD module is shown in figure 7. Sequential and concurrent signal approach is used to implement the sub-system. The state diagram is shown in figure 8.

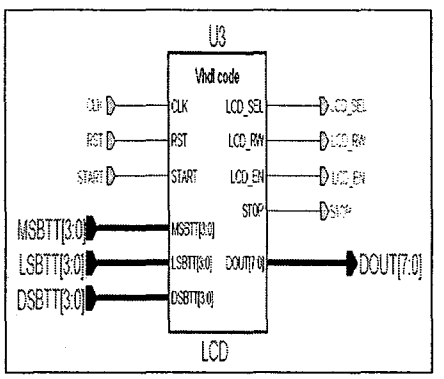


Figure 7: Block diagram of LCD Module

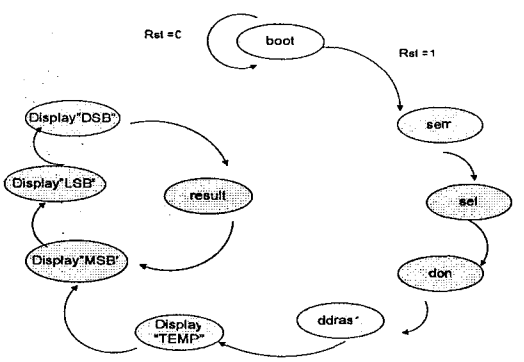


Figure 8: State diagram of LCD Module

The initial state is the boot state. The LCD needs 30ms to reach its sleep state after power up. No commands should be sent before that time. Thus, VHDL code is written to create a 30ms delay for the boot time. After the boot state, this sub-system starts sending the command code and data code to the LCD. The commands are used to initialize, turn on, set direction cursor flow and also set the first DDRAM address as 80H in the LCD. After that, the data code "TEMP:" is sent to the LCD.

After display the character "TEMP:", the system starts display the physical temperature which are 3 decimal number. The fist is the most significant bit of the temperature

value, the second is the least significant bit of the temperature while the last number is the first decimal point of the temperature value. After display the physical temperature value, the system set the DDRAM address to 86H so that the next data code will be displayed in the LCD starts from address of 86H. In the meantime, when count1=2, STOP=0, means that the Arithmetic Module is allowed to process new temperature data after the system has completely display the current temperature data. Hardware implementation for the system is shown in figure 9.

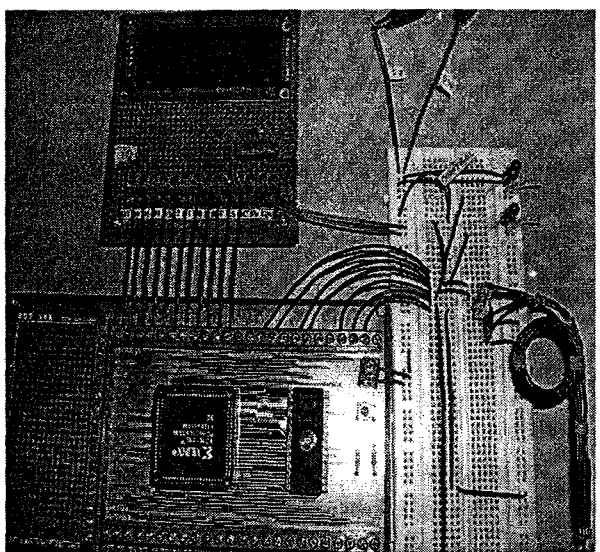


Figure 9: Hardware implementation of the temperature sensor

Analysis Of The System

Based on timing analysis of the system, it is found that the time needed for Transmission start sequence + Temperature Measurement sequence + SHT75 acknowledgement is 0.108ms. Time needed for temperature measurement by SHT75 is 55ms and time needed for reading data + FPGA processor acknowledgement + connection reset sequence is 0.24ms. Thus the time needed for each temperature data measurement is 55.348ms. The time needed for the temperature data to be processed is 56μs and the time needed for LCD to display the data is 2.64ms. Thus the total time needed to measure, process and display the temperature value is 58.014ms. From the information above, it is estimated that this system could measure, process and display the temperature around 17 times in one second.

The whole system requires 86% of configuration logic block (CLB) which is 346 out of 400 CLBs. According to the timing report, the highest frequency that can be achieved is 38.759MHz while the maximum net delay is 13.773ns. Thus the design is not violet the timing specification since the clock frequency used in the design is 250 KHz.

Conclusion

In this paper, it has been discussed the implementation of FPGA as a temperature data logger using SHT75 temperature sensor. State machine design methodology is used in the design of the system because a sequence of command code for the sensor SHT75 and LCD initialization is sent one after another. The system requires 346CLBs to be implemented on XC4010PC84 Xilinx device with the highest frequency that can be achieved is 38.759MHz.

References

[1] Mazidi M. and Mazidi J.G. (2000). The 8051 Microcontroller and Embedded Systems, Prentice Hall.

[2] Ayala. 2005 The 8051 Microcontroller. Thompson.

[3] S.Yalamancili. 1998. Introduction to VHDL:From Simulation To Synthesis, Georgia Institue of Technology. Prentice Hall.

[4] F.Scarpino. 1998. VHDL and AHDL Digital System Implementation. Prentice Hall.

[5] Xilinx. 1998. The Practical Xilinx Designer Lab Book. Prentice Hall.

[6] Floyd. 2003. Digital Fundamental With VHDL. Prentice Hall.