Microcontroller based Instrument for Tracking the change of Resonance Frequency with time

A Thesis submitted to the
Faculty of Engineering & Technology,
Jadavpur University
in the partial fulfilment of the requirements
for the degree of
Master of Technology in
Instrumentation and Electronics Engineering

Under the Guidance of
Dr. Bhaswati Goswami
Department of IEE, Jadavpur University

Submitted by
Somen Biswas
Examination Roll No: M4IEE1604

Department of Instrumentation and Electronics Engineering, Jadavpur University, Kolkata - 700 098 | May 2016

Microcontroller based Instrument for Tracking the change of Resonance Frequency with time

A Thesis submitted to the
Faculty of Engineering & Technology,
Jadavpur University
in the partial fulfillment of the requirements
for the degree of
Master of Technology in
Instrumentation and Electronics Engineering

Submitted by

Somen Biswas
Examination Roll No: M4IEE1604

Under the Guidance of
Prof. Bhaswati Goswami
Department of IEE, Jadavpur University

Department of Instrumentation and Electronics Engineering
Jadavpur University,
Kolkata–700098

May 2016

Department of Instrumentation and Electronics Engineering
Faculty of Engineering & Technology
Jadavpur University

Certificate of Recommendation

I hereby recommend the thesis entitled "Microcontroller based Instrument for Tracking the change of Resonance Frequency with time", submitted by Mr. Somen Biswas (Registration No 129478 of 14–15) and the work done under my guidance be accepted in partial fulfillment of the requirement for the degree of Master of Technology in Instrumentation and Electronics Engineering from the Department of Instrumentation and Electronics Engineering, Jadavpur University, Kolkata.

(Prof. Bhaswati Goswami)
Dept. of Instrumentation & Electronics Engineering
Jadavpur University
Kolkata 700098, India

Countersigned by -

(Prof. Rajanikanta Mudi)
Head of the Department
Dept. of Istrumentatation &
Electronics Engineering
Jadavpur University

(Prof. Sivaji Bandopadhyay)
Dean
Faculty of Engineering & Technology
Jadavpur University

Department of Instrumentation and Electronics Engineering
Faculty of Engineering & Technology
Jadavpur University
Kolkata–700098

Certificate of Approval

The foregoing thesis, entitled as "Microcontroller based Instrument for Tracking the change of Resonance Frequency with time" is hereby approved by the committee of final examination for evaluation of thesis as a creditable study of an engineering subject carried out and presented by Mr. Somen Biswas (Registration No 129478 of 14–15) in a manner satisfactory to warrant it's acceptance as a perquisite to the degree of Master of Technology in Instrumentation and Electronics Engineering. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only for the purpose for which it is submitted.

Committee of final examination for evaluation of thesis:

....
(Signature of External Examiner)

....
(Signature of Internal Examiner)

Declaration of originality and compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of my Master of Instrumentation & Electronics Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also, declare that as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name:

Roll No:

Thesis title:

Signature with date:

Acknowledgement

I offer my sincerest gratitude to my project guide, Prof. Bhaswati Goswami, who has supported me thoughout my project with her patience and knowledge while allowing me to work in my own way. I attribute the successful completion of my Masters degree as well as my project work to her encouragement and effort and without her this project would not have been completed or written.

I would like to extend my gratitude to Prof. Rajanikanta Mudi, Head of the Department of IEE, Jadavpur University and for providing valuable information regarding my thesis and Master's course work.

I would like to express my gratitude to Prof. Ratna Ghosh, Professor, Department of IEE, Jadavpur University for her motivation and advice for the timely completion of this thesis.

I would like to thank Prof. Kumardeb Banerjee, Professor, Department of IEE, Jadavpur University for his time to time advice on microcontroller.

I would like to thank Prof. Dr. Tamal Kanti Pal, Principle, Professor and Head, Department of Periodontist, Guru Nanak Institute of Dental Science & Research, Kolkata for his advice and help. I would also like to express my gratitude to all the staff members and patient of Dr.T.K.Pal's clinic for their cooperation.

I would also like to thank Mr. Atish Nandi, Power Engineering, Jadavpur University.

In my daily work, I have had the luxury of a warm and friendly group of seniors, Mrs. Aditi Bhattacharya, Mr. Nishant Pathak, Mrs. Paramita Banarjee and Mr. Arindam Sarkar. I would like to thank for their productive suggestions and motivation throughout the work.

A word of thanks is also due to all the teachers and the non-teaching staff of Department of IEE, Jadavpur University for providing required knowledge, information, equipments, devices and whatever help was needed during my coursework.

I thank my family members for supporting and encouraging me at each stage throughout my study at the University.

....

(Somen Biswas)
Instrumentation & Electronics Engineering
Jadavpur University

Abstract

The aim of the work is to design and develop a standalone instrument for checking the dental implant stability. The sensor used for this purpose has already been developed by previous researcher. A compatible input and its corresponding readout device has been developed in this work.

The novelty of the work lies in the fact that a microcontroller is used to generate a variable frequency sinusoidal signal whereas usually it is used to generate a sinusoidal signal of a particular frequency. The amplitude of the signal is set here at peak to peak 1volt with a bias of 0.5volt. Options are provided to introduce or remove the bias. The other novelty is detecting both positive and negative phase. To do so an innovative way of detection are used by two D flip-flops, one for positive and another for negative phase.

The instrument can be used to measure impedance phase response of an element. It has the facility to display it graphically. It is also capable to determine the resonance frequency from the phase response and display it. It can store the data in a PC. It is able to record and plot the shift of resonance frequency with time.

A rigourous experimental analysis has been done for calibrating the instrument. The calibration of the instrument has been performed using different combination of passive components as sensor and analysing the data statistically. The performance of the instrument has been checked by using in-vitro and in-vivo tests. In this case in-vivo readings are taken after implant reaches a stable condition. The recording shows repeatable nature.

Contents

Chapter 1

Introduction

1.1 Introduction

A standalone instrument usually designed with a aim to do some specific work. Determination of resonance frequency or specifically its shift with time using a standalone instrument has a broad application. The design of the sensor depends upon its application. One of the popular application of this type of instrument is for identifying the stability of dental implant. The stability can be measured using Resonance Frequency Analysis (RFA) technique in which shift in resonance frequency indicates the change in the bonding of the dental implant with the surrounding bone and tissue. With the improvement of the bonding over time the resonance frequency will reach a stable value, which is indicative of the stability of the implant. Osstell^{TM} transducer (Integration Diagnostics AB, Sweden)[1] is an available instrument based on this technique to clinically monitor the stability of a dental implant. The sensor system used consists of a sensor actuator piezo element, supported by a metal strip [2],[3]. It is screw tightened onto the implant or to its transmucosal component, the abutment, and measures the resonance frequency.

It has been reported in[4] that the phase change of a piezo element is better identifiable than its magnitude so, the impedance phase response may be used as a measurand. Hence, for this purpose there is a need to develop a phase response measuring instrument. A rigorous study on the design and development of a piezo based sensor has been done [4][5]. A preliminary work has also been done on the design and development of an instrument dedicated for this purpose [6],[7]. The lacuna lies in the fact that they are not standalone instrument. The input used is from a function generator and the range of phase that can be measured by the instrument are 0° to -90° only. But the range of phase of the sensor is ±90°. The available design does not have the facility to store the data for further analysis later.

1.2 Objective of the work

Objective of this work is to address the problems of the instrument reported in [6],[7] and to develop a standalone instrument with some added feature. To make the instrument self-sufficient there should be an in-built input supply for the sensor. This supply should be a sinusoidal voltage signal because the measurand is the impedance phase response of the sensor. The aim is to measure the phase within $\pm 90°$. With an objective of a graphical display of the response with a facility of storing of the data will be an added advantage.

1.3 Organisation of the thesis

The thesis is organised into two parts. The novelty of the work lies in the first part which are explained in Chapter 1 and Chapter 2. Chapter 1 explained in details the generation of in-built input supply voltage to the sensor. This has been done using a combination of software and hardware. Chapter 2 deals in details the design and development of a circuit for measuring the phase of $\pm 90°$ of the sensor. The second part consists of assembling all the objectives. A microcontroller is used to control the total instrument together with incorporating display and storing facility. Calibration and testing the performance of the instrument is explained in the same chapter. Last chapter consists of conclusion and future scope of the study.

Chapter 2

Sine Wave Generator

The specification of the input of this sensor, as mentioned, should be a varying frequency sine wave within a range of 1kHz to 20kHz and an amplitude of about ±0.5volt. A better resolution of frequency will provide better measurement as the measurand is the resonance frequency of the sensor. Options must be provided to increase or decrease the frequency gradually as and when required.

A sine wave can be obtained using analog circuits that is, passive electronic components where the output will be a continuous time signal or using digital circuits or microprocessor / microcontroller where the output will be discrete time signals. The advantages of analog circuit are i) there is no need to use Digital-to-Analog Converter which will increase the cost of the instrument, ii) high frequency noises due to the digitization of the signal will not appear in the input. The disadvantages are the output may change due to the change of the environment and is sensitive to the components. On the otherhand advantages and disadvantages of using digitised or discrete time signals are just the disadvantages and advantages of analog output. Other advantages of digital output are easy to store, easy to manipulate and multitasking. The read out device is also easier to develop and manipulate according to the requirement. So, considering advantages and disadvantages of both it seems that digital output will be a better option.

2.1 Wave generation technique

The steps used for generation of sine wave of a particular frequency are first to provide the frequency, f of the wave to be generated, second is to decide the number of samples, N or sampling frequency, f_s required, third is to compute the values of the sine wave with an interval to be decided by the number of samples or the sampling frequency. These values are in 8bit pattern. The next step is to output these 8bit and fed into a R-2R ladder DAC for converting it into its analog form to

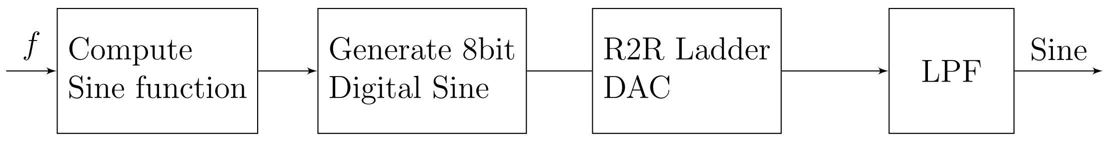


Figure 2.1: Block diagram for generation of sine wave of a particular frequency

be used as an input to the sensor. The block diagram of this is shown in Fig.2.1. At every frequency the number of samples required have to be calculated and corresponding look up table for that frequency have to be computed which needs a faster microcontroller with a suitable memory and suitable number of ports to send the data parallely to the DAC.

2.1.1 Choice of Microcontroller

A sine wave can be generated using a microcontroller by calculating sine function from tailor series. In microcontroller there are library functions where there is a function to calculate sine values by math.h header file of C30 compiler, this requires a significant time for execution. In this scenario all the values of pre-calculated sine values can be stored in a array just like a lookup table. Array must be a circular array to execute cycle of the sine wave. At the time of execution of the instruction to out the values from the look up table one by one a pointer is used which will be incremented by one after a defined fixed time interval. For the highest frequency in the range, that is, 20kHz sine wave if 50 samples/cycle is used then the sampling frequency, f_s will be 50 x 20kHz = 1MHz. To maintain 1MHz sampling frequency it needs 1 Million Instruction Per second (MIPs) and for sending the instruction of changing the pointer value after a specified interval needs at least 10 times instruction per second, that is 10MIPS. Also DAC operates in parallel execution so, execution speed should be more than 10MIPs. Hence, considering the thumb of rule it should be about 20 MIPs. dsPIC30f family has 30MIPs speed which will be suitable for this purpose. Among dsPIC30f[8] family the best chip dsPIC30f4013[9] chosen for the purpose.

2.1.2 Choice of number of sample

It is well known that discrete time sinusoid is represented as $x[n] = sin[2 \cdot \pi \cdot f \cdot n \cdot \delta t]$ where f is its frequency and δt is the sampling interval in s. Sinusoids will be periodic of period N if

$$N = 1/(f \cdot \delta t) = f_s/f \tag{2.1}$$

where N is number of samples in a cycle of sine wave and hence is an integer, f_s is the sampling frequency. That is, $x[n] = sin[\frac{2 \cdot \pi \cdot n}{N}]$. So, for a given f there are two

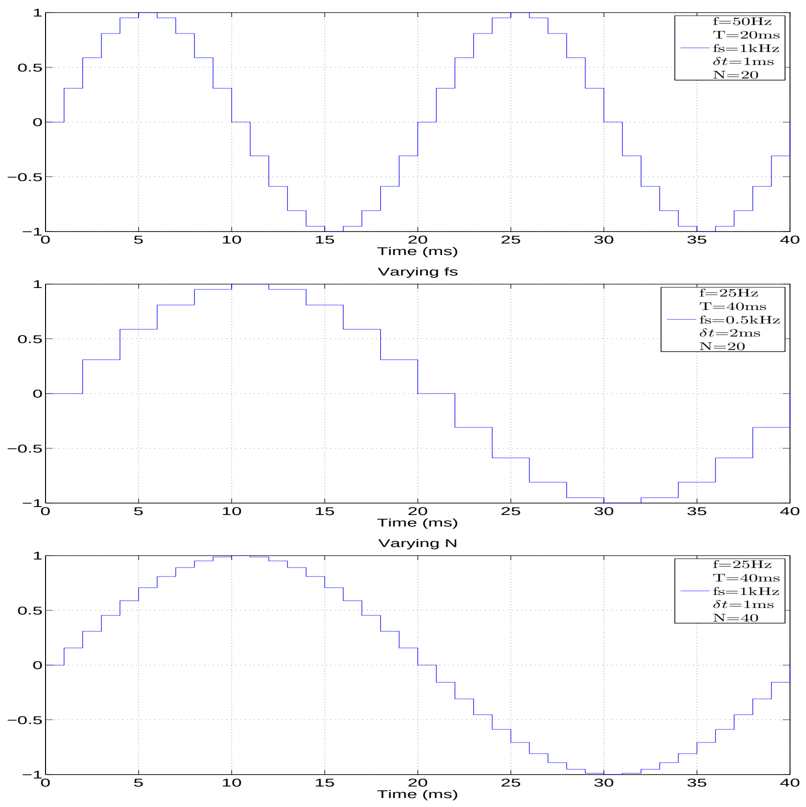


Figure 2.2: Plots of sine wave a) of frequency 50Hz with N=20 and δt=1ms b) of same N but different δt c) of same δt as (a) but different N

options: either choose N or δt and to vary the frequency of the sine wave change either of the two. The effect of change of δt or N is shown in Fig.2.2. From the figure it is clear that the frequency of the sine wave can be changed either by N or δt.

The output of the DAC will be a step type waveform, like that as shown in Fig.2.2, where the step value will depend upon the value of the sine wave at that instant and width of the step will be the sampling interval δt. This steps will introduce higher order odd harmonics and the values of the odd harmonics will change according to the step size. To eliminate these harmonics from the output and generate a smooth sine wave it is required to design a low pass filter whose cut off frequency f_c will change according to the presence of harmonics. It is difficult to implement this type of low pass filter with dynamically changing cut off frequency. To eliminate this problem the option is to keep δt fixed so that the step width remains same and a low pass filter with a fixed f_c can be used for the purpose.

The other option, as mentioned, is to vary N. In this case pre-calculated N number of data of the sine wave at a fixed interval of δt are stored into an array as unsigned char in the memory as a look up table. Every time the frequency changes N will change and the values in the look up table is refreshed with the new calculated values. Maximum size of array has to be calculated from the lowest frequency of the sine wave to be used so that, $N_{max} = f_s/f$, where the sampling frequency f_s is fixed for the total range of frequency of the sine wave.

The sampling frequency f_s = 1MHz as calculated in Section 2.1.1. The lowest frequency of the sine wave, in this instrument is 1kHz. So,

$$N_{max} = \frac{f_s}{f} = \frac{10^6}{10^3} = 1000$$

Therefore, maximum number of sample in the array will then be 1000.

2.1.3 Computation of sine values

The microcontroller dsPIC30f4013 of Microchip provides C30 [10] compiler to program sine function in math.h header file. This returns an array of floating point numbers which are the calculated values of sine function. The number of data to be calculated and to be stored in memory as look up table is specified by N. The amplitude of the generated sine wave is specified by the instrument for this purpose which is peak to peak 1volt that means the output of the microcontroller should be ± 0.5 volt. This 1volt peak to peak value is discretized into 255 steps considering data as a 8bit word. The problem of R-2R ladder DAC is that it does not have the feature of converting $-$ve values so, a DC shift is introduced to vary the amplitude within 0 to 1volt instead of ± 0.5. The flowchart to be used for generation of a sine wave of a particular frequency f to be supplied by the user is shown in Fig.2.3.

Next step is to send the data to DAC at a particular interval which can be set by a timer provided in the microcontroller chip. The output from DAC is passed through a low pass filter to get a smooth continuous time sine wave.

2.2 Generation of digital sine wave

Sine wave can be generated by the steps discussed in Section 2.1 which are implemented one by one as stated in the following sections.

2.2.1 Setting sampling frequency f_s

In this case the sampling frequency f_s or interval δt is fixed and can be generated using timer. There are five timer options in dsPIC30f4013 chip. The value of the

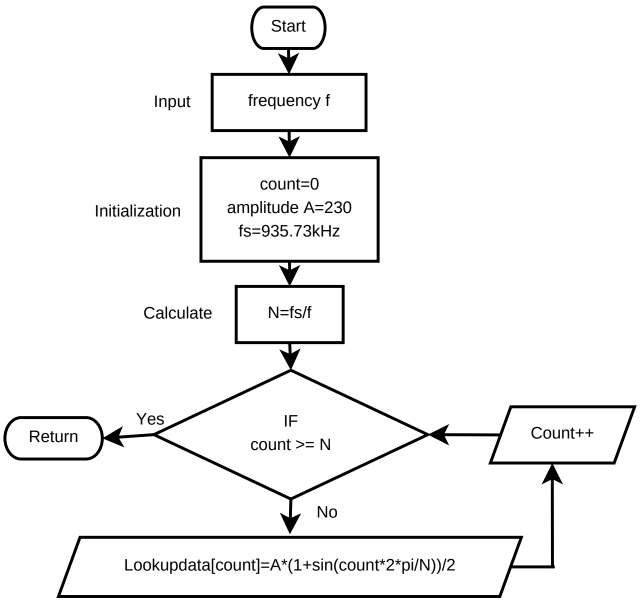
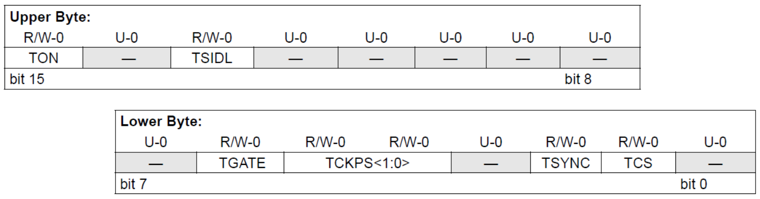


Figure 2.3: Flowchart to calculate sine function



Upper Byte:

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:

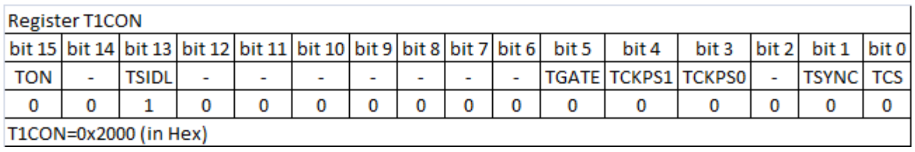
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

Figure 2.4: Bit configuration of Control Register of Timer 1

interval is critical so it is required to use the timer which has maximum number of bit resolution and among them timer 1 is a 16 bit timer and will be able to provide minimum available interval. Timer 1 is a A-type timer as specified in dsPIC30F family Reference Manual (DS70046). Steps that have been followed to set δt are given below.

1 Set Timer1 control register TICON:

The bit configuration of T1CON is shown in Fig 2.4. The change in the bit pattern will control the interval and interrupt of Timer 1. The bits as shown in Fig.2.4 are
bit 15 TON is Timer On Control bit:*Initially =0, at the time of generation =1*
bit 13 TSIDL is Stop in Idle Mode bit :*=1 to save power in idle mode*
bit 6 TGATE is Timer Gated Time Accumulation Enable bit :*=0 as default*
bit 5-4 TCKPS are Timer Input Clock Prescale Select bits :*=00 to generate*


Register T1CON

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
TON	-	TSIDL	-	-	-	-	-	-	-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

T1CON=0x2000 (in Hex)

Figure 2.5: Bit pattern of Control Register of Timer 1

high frequency interrupt

11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 2 TSYNC is Timer External Clock Input Synchronization Select bit: *=0 as default*

bit 1 TCS is Timer Clock Source Select bit : *=0 as default to run using internal clock cycle.*

Other bits are unimplemented bits. The bit pattern for Timer 1 is shown Fig.2.5

2 Setting Interrupt control register IEC0

Interrupt control register, IEC0 is defined to enable interrupt of timer. For Timer 1, bit 3 of IEC0 should be defined and =1.

3 Setting Period Register PR1

To get a desired sampling frequency f_s it is required to select a suitable value of Period Resister (PR1). The formula given in dsPIC30f Reference Manual is

$$PR1 = \frac{Fcy}{f_s \cdot Prescaler} - 1 \tag{2.2}$$

where Fcy is MIPs. Fcy is calculated from the frequency of the crystal oscillator. The frequency of the crystal used in dsPIC30f4013 is 7.3728MHz which can achieve maximum performance using Phase Locked Loop (PLL). C30 compiler provides the code XT_16PLL to configure PLL and external oscillator which will increase the crystal frequency 16 times that of its original frequency. Using the code, clock speed achieved is, $Fosc = 7.3728 \times 16 =$ 117.9648 MHz. And for one instruction 4 clock cycle is required, MIPS achieved is then $Fcy = Fosc/4 = 29.4912$ though in dsPIC30f4013 data sheet specification it is 30MIPS.

So, with $f_s = 1MHz$, $Fcy = 29.4912$ MIP and $Prescaler = 1$ as bit 5 and

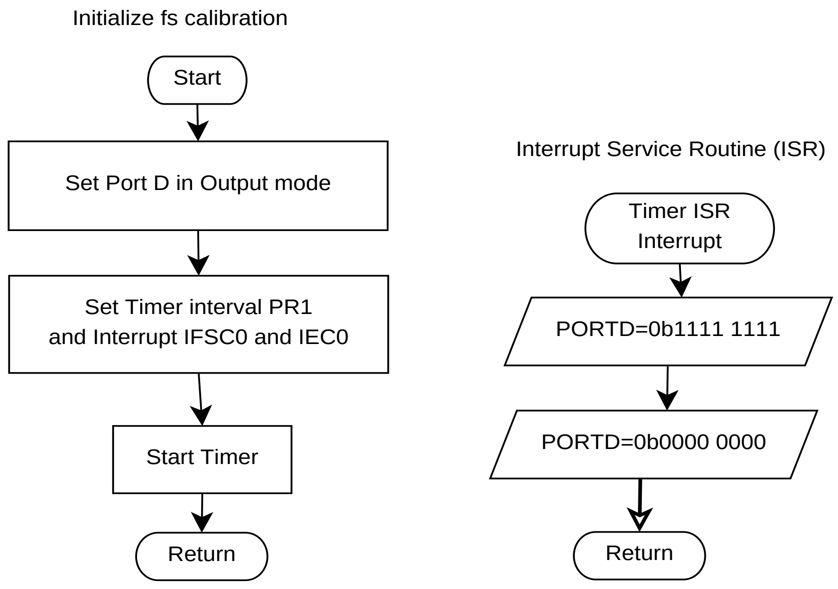

Initialize fs calibration
Start
Set Port D in Output mode
Set Timer interval PR1
and Interrupt IFSC0 and IEC0
Start Timer
Return
Interrupt Service Routine (ISR)
Timer ISR
Interrupt
PORTD=0b1111 1111
PORTD=0b0000 0000
Return

Figure 2.6: Flowchart for measuring the sampling frequency with DSO

4 already has been set as 0 0

$$PR1 = \frac{29.4912}{1 \cdot 1} - 1 = 28$$

Some parameters like watchdog timer, power up timers and code protections should be kept off while external reset should be kept on, C30 also provides codes to configure these parameters.

2.2.1.1 Calibration

The output of the signal through the port D of the microcontroller is measured by a digital oscilloscope (DSO) using different PR1 to find out the value of actually generated sampling frequency. First T1CON register is set with the bit pattern shown in Fig.2.5 and then the flowchart shown in Fig.2.6 is used for sending data to the oscilloscope for measurement.

Frequency calculated using equation (2.2), measured frequency obtained from port D using oscilloscope, its standard deviation (SD) with different PR1 are tabulated in Table 2.1. From the table it is observed that calculated sampling frequency does not match with the measured one so, the standard deviation is determined using k number of cycles. It is further observed that the measured frequency is exactly 1MHz when PR1=28 with SD of 6.19Hz. So, to keep the sampling frequency within 1MHz it is better to choose PR1=30 where f_s =935.73, slightly less than 1MHz of SD 5.63.

Table 2.1: Calculated and measured sampling frequency with different PR1

PR1	Calculated f_s in kHz	Measured f_s in kHz	SD in Hz	reading(k)
25	1134.276923	1171	7.9	10
26	1092.266667	1071	7.9111	50
27	1053.257143	1034	7	10
28	1016.937931	1000	6.19	14
29	983.04	967.61	6.1097	10
30	951.3290323	935.73	5.6358	30
31	921.6	907.9	6.079	10
32	893.6727273	876.71	5.89	10

2.2.2 Calculation of Number of sample, N

The values of f_s and PR1 are selected according to the criteria stated above and are used for the total range of frequency of the sine wave, 1kHz to 20kHz. The frequency of sine wave is then changed by changing the value of N as discussed in Section 2.1.2. and the relation is given in equation (2.1). The calculated value may not be an integer so, only integer part of the number is considered as the value of N discarding the fractional part. This approximation of the value of N will introduce a deviation from the desired frequency. So, to eliminate the error between the desired and generated frequency it is recalculated with the used value of N and considered as the frequency of the sine wave.

2.2.3 Sending data at the output port

The data has been computed using the C30 complier which corresponds to the N number of samples of sine wave amplitude, of a particular frequency and stored as look up table as discussed in Section 2.1.3. The maximum amplitude of sine wave calculation will be $\frac{255}{2} = 127$. A reduced value is selected which is 115. The look up table is refreshed with the change of the frequency. These are 8bit word and to be send parallely to the R-2R ladder DAC. So, an 8bit port is required to send the data as an output. The pin configuration of dsPIC30f4013 is shown in Fig. 2.7. Pin diagram of the device as observed from the Fig. 2.7 shows that only Port B has 8bit data bus. Thus, LATB is used to send discrete sine value and next data is selected from the look up table after an interval of δt. Flow chart to send data from look up table whose values are computed from the sine wave function, through Port B, is given in Fig.2.8. Block 1-4 of Fig.2.8 are used for initialisation of the timer and Port B and Block 5-9 for sending the data at the output Port so as to be used as an input to the R-2R ladder DAC.

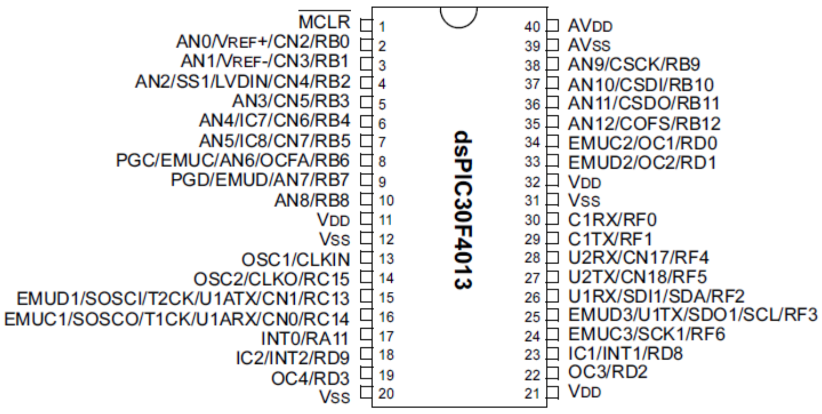


Figure 2.7: dsPIC30f4013 Pin diagram

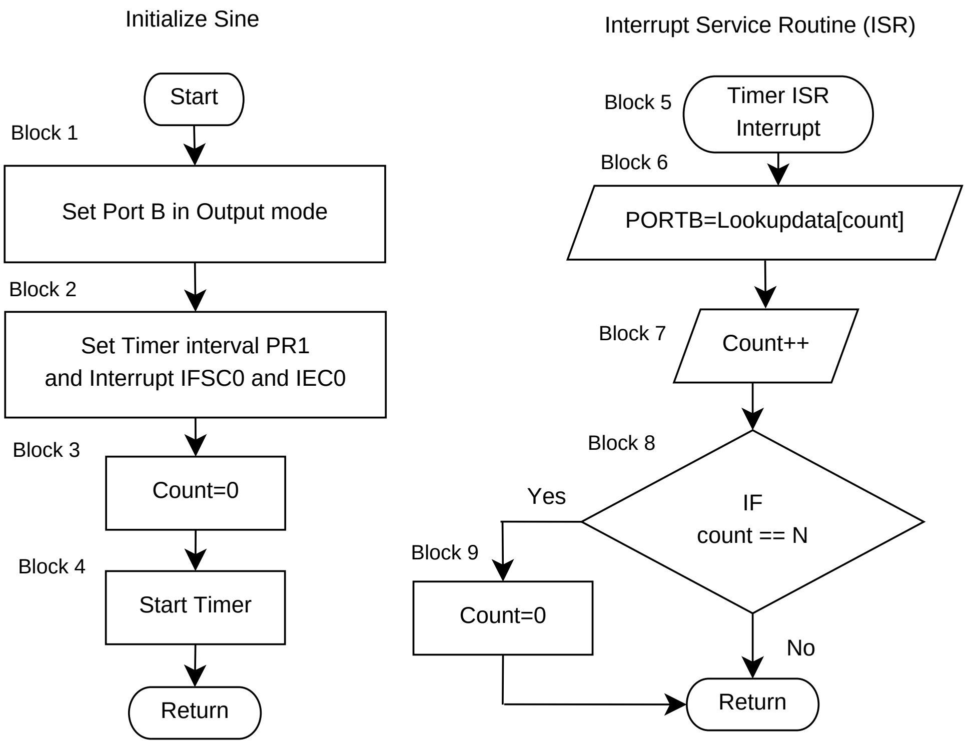


Figure 2.8: Flowchart to generate discrete sine function

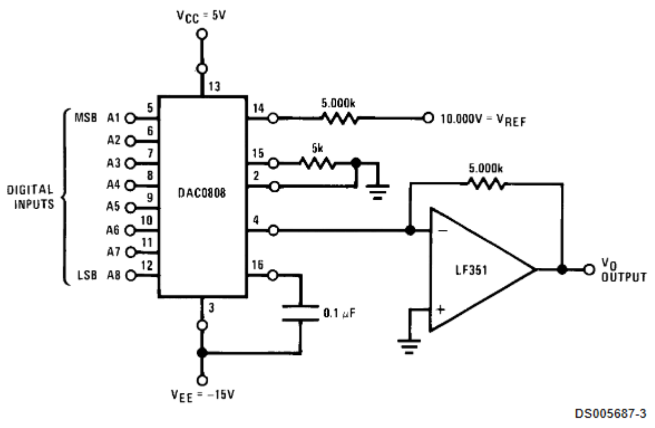


Figure 2.9: DAC0808A recommended circuit

2.3 Generation of continuous time sine wave

Sine function in its digital form is first converted into its analog version using a DAC then it is smoothed using low pass filter. An optional DC shift circuit is also included to get a continuous time voltage sine wave.

2.3.1 Conversion of data from digital to analog form

Data obtained at the output port, Port B of dsPIC30f4013 are 8bit word. It has been converted to its analog form using DAC0808 [11] chip. Recommended typical circuit diagram of the chip is shown in Fig.2.9. A few changes have been done in the circuit those are
i) all 5kΩ resistances are replaced by 4.7kΩ.
ii) 10 volt V_{REF} is changed to 5 volt to simplify the power supply design.
iii) Opamp is changed to LF411 instead of recommended LF351 to cater the frequency range of 1kHz to 20kHz.
iv) Ignoring the guideline power supply is reduced to -5V instead of -15V.
v) The reference current at pin number 14 is now $5V/4.7k\Omega = 1.064mA$ so, the maximum output current from pin number 4 will be 1.06mA.

2.3.2 Filtering circuit

The specification as supplied by the user the output should be a sine wave voltage signal of amplitude ± 0.5 volt and frequency range 1kHz to 20kHz. The output from pin 4 of DAC 0808 is a current signal. The shape of the signal is changing step where the value of the steps are that of sine function at that instant like that as shown in Fig.2.2 . Maximum amplitude of the signal is 1.064mA and its minimum value is 0. So, to get a smooth voltage signal it is passed through an Opamp filter circuit. The modification of the signal are: i) current signal converted to voltege signal. ii) high frequency noises removed, iii) an optional DC shift.

2.3.2.1 Current to Voltage conversion

The wave shape of the generated sine from pin 4 of DAC0808 is in the form of square step and it is a current signal. First it has to be converted into a voltage signal. An Opamp based circuit as shown in Fig.2.10 is used for this purpose. In the specification of the instrument it is given that the amplitude of the output sine wave should lie almost within 1 volt. So, to convert maximum current amplitude of 1.064mA into a maximum voltage amplitude almost within 1 volt the value of the resistance has been calculated as

$$R_f = \frac{1V}{1.064mA} = 0.94k\Omega$$

Nearest standard available value of resistance is $1k\Omega \pm 1\%$. Therefore, V_{somax} of the circuit shown in Fig.2.10 will be

$$V_{somax} = 1.064mA \cdot 1k\Omega = 1.064V.$$

2.3.2.2 Signal Filtering

The aim is to generate a smooth sine wave. Hence, it is required to design a low pass filter (LPF) whose cut off frequency will cut off the sampling frequency. An Opamp based active low pass filter is shown in Fig.2.10. The calculated sampling frequency is $951kHz$ as obtained from Section 2.2.1.1. The cut off frequency of LPF[12] then may be kept in between 100kHz and 200kHz which is well above the operating frequency range and hence there will not be any effect on its amplitudes. The design of the filter is obtained using following equations. The value of $R_f = 1k\Omega$ has already been determined as given in Section 2.3.2.1. If the cut off frequency is $f_c = 200,000Hz$ then

$$C_f = \frac{1}{2 \cdot \pi \cdot 200,000 \cdot 1,000} = 0.796nF$$

If the cut off frequency is $f_c = 100,000Hz$

$$C_f = \frac{1}{2 \cdot \pi \cdot 100,000 \cdot 1,000} = 1.592nF$$

So, to keep the cut off frequency between 100kHz and 200kHz then let the value of the capacitor be $1nF$. Then the actual cut off frequency will be

$$f_c = \frac{1}{2 \cdot \pi \cdot 1 \cdot 10^{-9} \cdot 1,000} = 159.2kHz$$

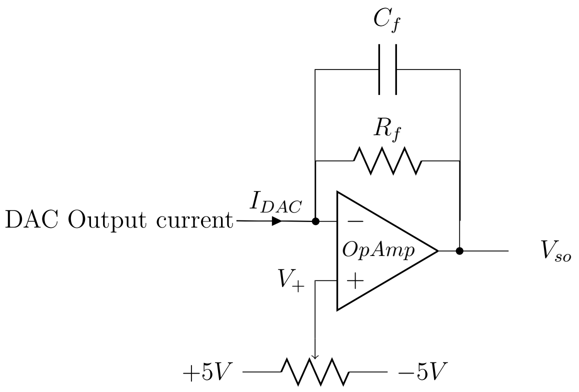


Figure 2.10: Multi-purpose Opamp Circuit used for converting current to voltage signal, filtering and a DC shift

2.3.2.3 Circuit for DC Shift

Generated sine wave has a DC shift which oscillates between 0 and 1.064V. As mentioned, this varying frequency (between 1kHz to 20kHz) sine wave will be used as an input to a sensor for recording its impedance phase response. So, this DC shift above zero will not affect the measurand. A circuit has been included to introduce any DC shift[13] as and when required. For that purpose, the non inverting terminal voltage of the circuit is varied manually with the help of a potentiometer to introduce a negative offset. It is shown Fig.2.10.

Equation used is

$$V_{so} = V_+ - R \cdot I_{DAC}$$

where varying V_+ the DC offset can be changed manually as and when required.

The total circuit diagram of DAC 0808 for a continuous time sine wave generation is shown in Fig.2.11. The component values used are tabulated in Table 2.2.

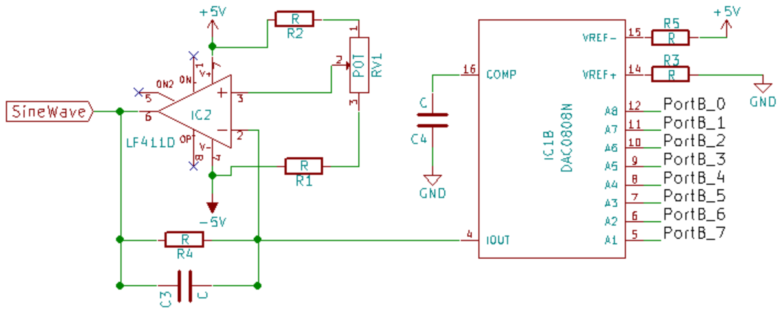


Figure 2.11: Modified DAC with external circuit

Name	Value
R1	$10k\Omega$
R2	$10k\Omega$
R3	$4.7k\Omega$
R4	$1k\Omega$
R5	$4.7k\Omega$
RV1	$10k\Omega$
C3	$10nF$
C4	$0.01\mu F$

Table 2.2: Component

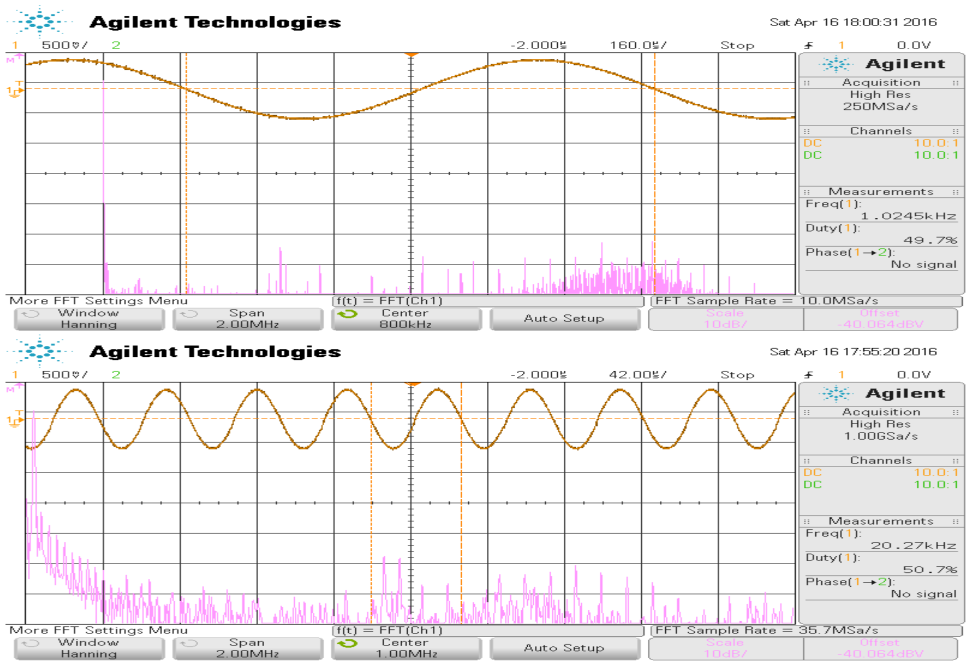


Figure 2.12: Plots of sine wave and its spectrum at 1kHz and 20Khz

2.4 Testing of purity of sine wave

A continuous time sine wave of frequency 1kHz and 20kHz as obtained from the output of DAC 0808 are shown in Figs.2.12. But, purity of sine wave is an important factor for measuring the resonance frequency from impedance response. One of the way to check it is to determine its Total Harmonic Distortion (THD)[14] which is

$$THD = \sqrt{\frac{V_1^2 + V_2^2 + V_3^2 + ... + V_n^2}{V_0^2}}$$

where V_0 represents amplitude at fundamental frequency and V_n represents that at n^{th} harmonic. For calculating THD at every frequency of the generated sine wave all local maxima and one global maxima of the amplitude are determined from the frequency spectrum of the output in MATLAB. Log plots of frequency spectrum is shown in Fig.2.13. Considering global maxima as fundamental frequency and all local maxima as its harmonics calculated values of THD of some selected frequency of sine wave is tabulated in the table 2.3. From the table it is observed that THD is less than 1% for frequencies less than 6kHz and increases as the frequency increases and is about 3.7% at 20kHz.

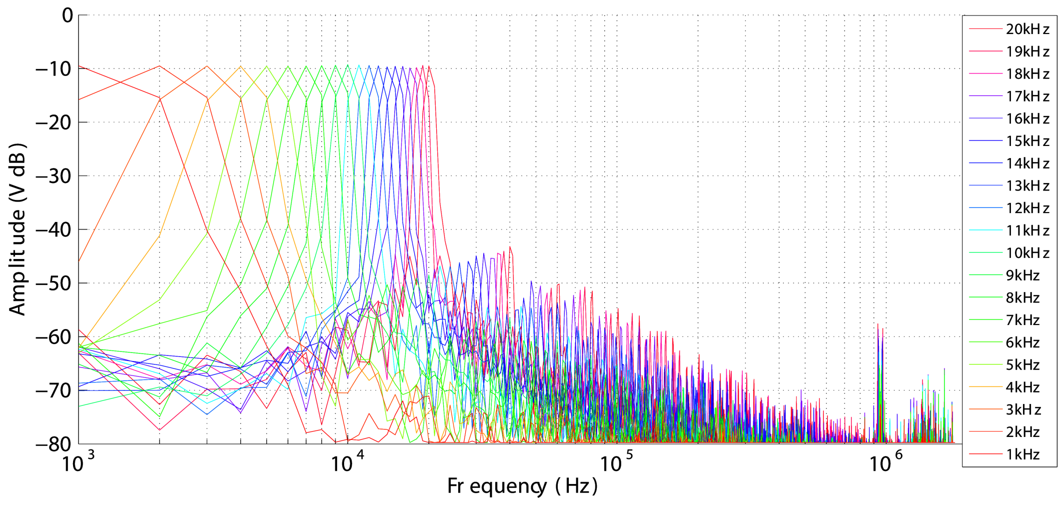


Figure 2.13: Frequency spectrum of each generated sine wave from 1kHz to 20kHz

Table 2.3: THD of Sine wave

Frequency(Hz)	1000	5000	10000	15000	20000
THD (%)	0.254	0.714	1.474	2.482	3.63

2.5 Error in frequency of generated sine wave

Error in generation of the sine wave have been checked by measuring the output using Agilent MSO-X 3032A digital oscilloscope, considering it as a standard instrument. Absolute frequency error is calculated by subtracting the frequency as read by the microcontroller, called as $InstrumentFrequency$, and the measured frequency, named as $MSOXFrequency$ and taken its absolute value as shown in the following equation.

$$AbsoluteError = |MSOXFrequency - InstrumentFrequency|$$

Relative error is also calculated using

$$RelativeError = \frac{AbsoluteError * 100}{CorrespondingFrequency}\%$$

Absolute error at every frequency for 6 acquired data is plotted in the Fig.2.14 and relative error in Fig.2.15. Mean of absolute errors and relative errors and their standard deviation (SD) of some selected frequencies are given in the table 2.4. From the figures and the tables it seems that both the errors are more for high frequencies.

	Mean of Error		SD of Error	
Frequency (Hz)	Absolute (Hz)	Relative (%)	Absolute (Hz)	Relative (%)
1509	0.667	0.044	0.816	0.054
5000	7.000	0.140	8.000	0.160
10300	15.700	0.156	0.817	0.156
14880	21.700	0.146	16.300	0.110

Table 2.4: Statistics of frequency error

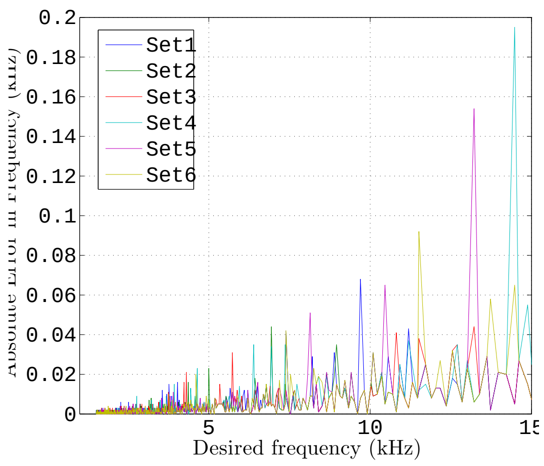


Figure 2.14: Absolute Error

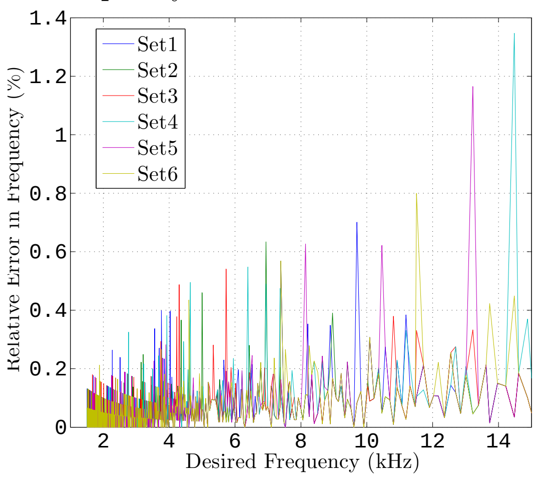


Figure 2.15: Relative Error

2.6 Conclusion

The purity and repeatability of generated sine wave has been tested by determining total harmonic distortion (THD) and the errors of generation of a particular frequency has also been checked for the total operating range. Both THD and error increases with the increase of frequency though they are between a tolerable limit. So, the generated sine wave within 1kHz to 20kHz can be used for the purpose of it is generated.

Chapter 3

Phase detection

This chapter deals with the design of the circuit to measure the impedance phase response. The concept is to use a sine wave signal of varying frequency as an input to two path: one consist of resistive elements and considered as the reference path ref and the other contains the sensor, named as sig. So, a measure of the phase of the sensor can be obtained from the phase of the voltage output of the sensor s' with respect to the reference path r'. The block diagram of the scheme is shown in Fig.3.1. The signal obtained from both sensor path and reference path is converted into a square wave s'' and r'' respectively to determine the phase between two.

3.1 Signal Sensing circuit

The supplied voltage as mentioned is a sine wave peak to peak 1volt. To measure the impedance of the sensor it is required to measure the current passing through it in terms of voltage. An opamp based current to voltage converter circuit is shown in the upper part of Fig.3.2. Similarly, in the reference path a unity gain opamp circuit is used as shown in lower part of Fig.3.2. A phase shift of 180° will be introduced in both outputs from reference and sensor path, hence the phase between them will remain same.

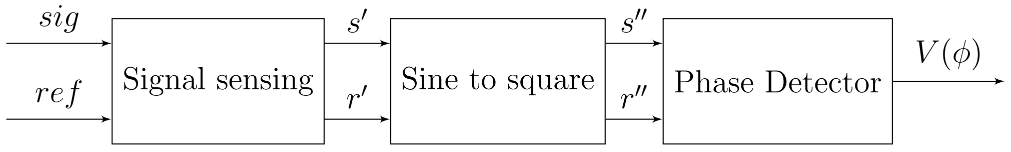


Figure 3.1: Block diagram of phase detection circuit

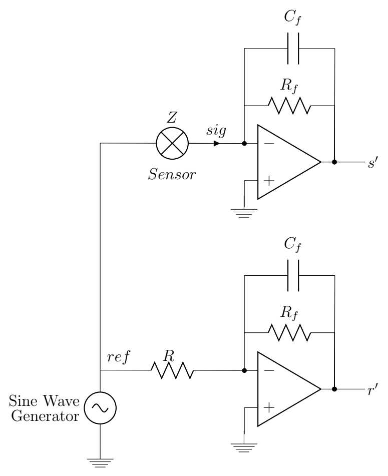


Figure 3.2: Signal Sensing Circuit

3.1.1 Selection of an OpAmp

To make the design reliable and repeatable it is important to chose an appropriate opamp for the circuit considering some of its parameters.These are discussed below.

3.1.1.1 Bandwidth

Bandwidth of the opamp is very important as the circuit will work with varying frequency in the range of 1kHz to 20kHz and also in this case with varying impedance. To find out the desired gain within the operating frequency range measured impedance response of a piezo based sensor as shown in Fig.3.3 has been considered. Average impedance magnitude of the sensor as calculated from the data of the Fig.3.3 is $2.263k\Omega$. A better performance will be obtained if this average impedance magnitude is used as feedback resistance R_f in the circuit shown in Fig.3.2 so as to keep voltage gain nearer to unity. Hence, considering nearest available resistance, R_f is $2.2k\Omega$. As a thumb of rule the gain of the open loop opamp should be 10 times the calculated one that is 20dB. Bode plot of two general purpose opamp LF411 and LM741 are shown in Fig.3.4a and 3.4b for comparison. Figure shows that LM741 has gain 10dB at 20kHz while LF411 have a larger bandwidth and phase is almost constant upto $1MHz$. So, from the bandwidth point of view LF411 will be a better choice.

3.1.1.2 Input bias current

For better performance the opamp should have a high input impedance and low input bias current. Input impedance for LM741 and LF411 is $2^6\Omega$ and $10^{12}\Omega$

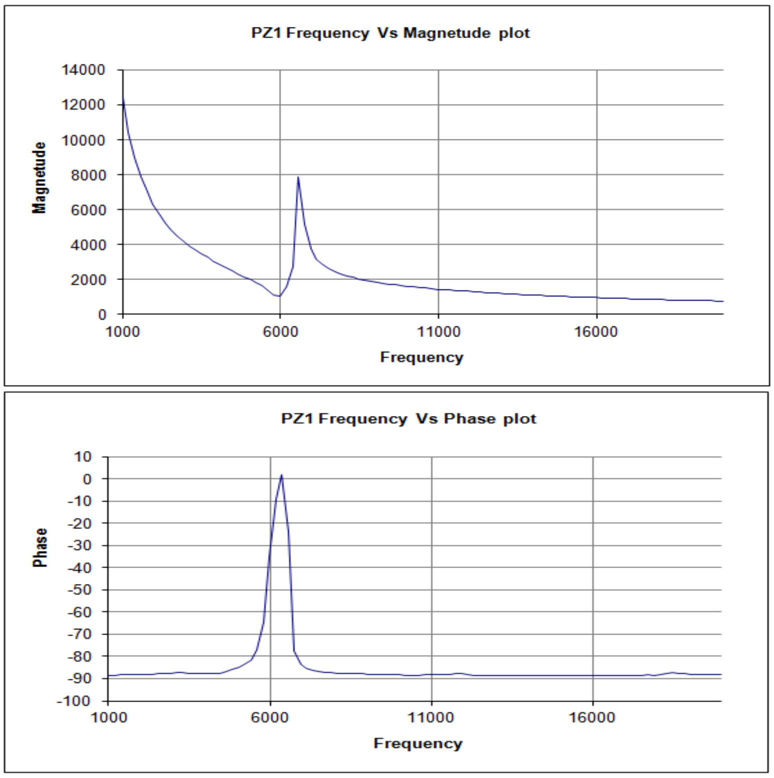
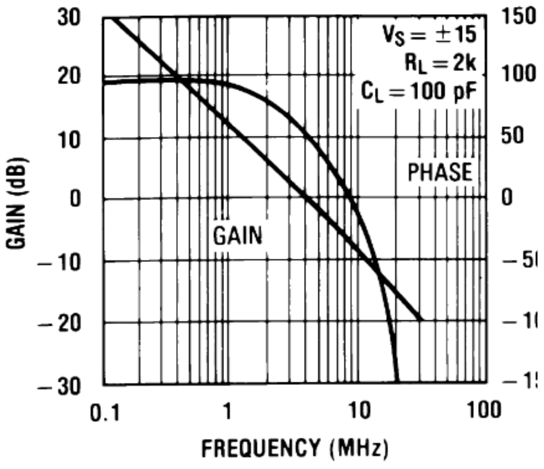
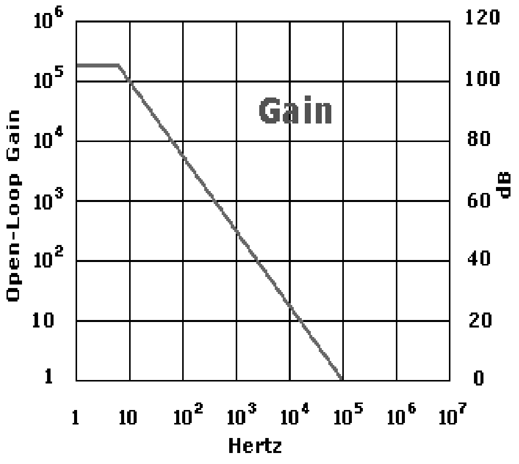


Figure 3.3: Impedance magnitude and phase of a piezoelectric sensor



(a) LF411



(b) LM471

Figure 3.4: Bode plot of open loop LM471 and LF411

respectively and input bias current for LM741 and LF411 is $80nA$ and $50pA$ respectively at 25^{o}C. So, LF411 is again a better choice if this criteria is considered.

3.1.1.3 Slew rate

Another important criteria is to consider the slew rate of the opamp within the operating frequency range. Maximum allowable frequency of the sine wave is 20kHz. So, the voltage output of the opamp at this frequency will be

$$V_o = A_v \cdot 0.5 \cdot sin(2\pi \cdot 20 \cdot 10^3 \cdot t)volt \tag{3.1}$$

where A_v is the gain of the amplifier, which is selected as 10 and 0.5 is the specified amplitude of the sine wave. Then, rate of change of voltage is

$$\frac{dv}{dt} = A_v \cdot 0.5 \cdot 2\pi \cdot 20 \cdot 10^3 \cdot cos(2\pi \cdot 20 \cdot 10^3 \cdot t)volt/sec \tag{3.2}$$

Thus required maximum slew rate is $10 * 0.5 * 2\pi * 20k$ or $0.629V/\mu sec$. Slew rate for LM741 and LF411 is $0.5V/\mu sec$ and $15V/\mu sec$ respectively which indicates to use LF411.

3.1.2 Choice of feedback resistance

As mentioned in Section 3.1.1.1 a better choice of the value of the feedback resistance is $2.2k\Omega$. This has been tested using the sensor for which it is designed. Some other resistance values are also used to check it. Result of the test is shown in Fig.3.5.

Comparing the impedance phase response shown in Fig.3.3 and phase response in terms of voltage in Fig.3.5 visually it may be said that the response is better with $4.7k\Omega$ than $2.2k\Omega$. So, a middle value, that is, $3k\Omega$ is used as feedback resistance.

3.1.3 Noise at the output

The phase between the *sig* and *ref* will be measured by the time delay between them identifying delay of the crossing of zero between two. A noisy signal mostly appears in *sig* path. This will detect many zero crossing position near to actual zero crossing and also generate an erroneous square wave as shown in the Fig.3.6a.

3.1.3.1 Filter circuit to remove noise

Two steps have been used to reduce the noise in the *sig* path. The first is to use coaxial stereo audio cable so as to reduce noises due to external electrical and

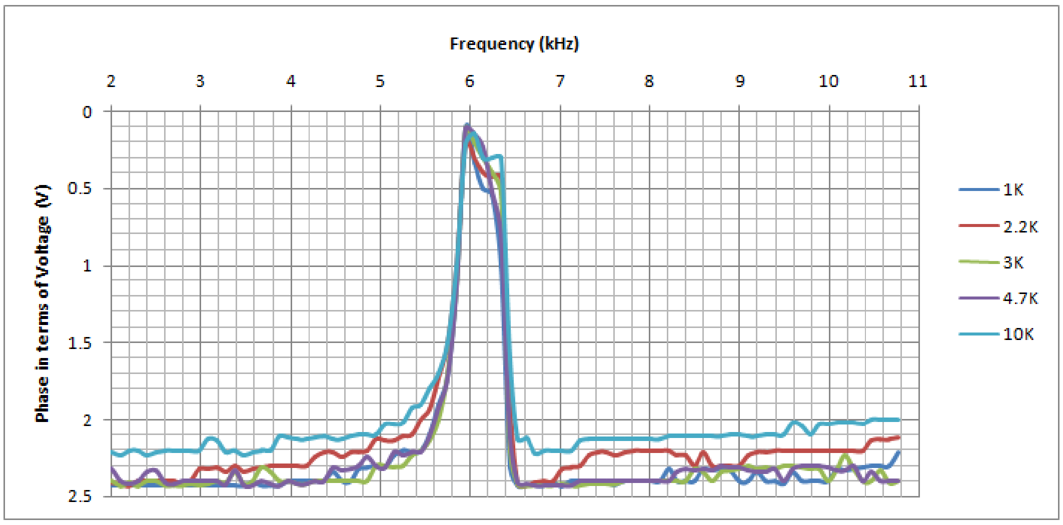


Figure 3.5: Phase response of a piezoelectric sensor using different feedback resistance

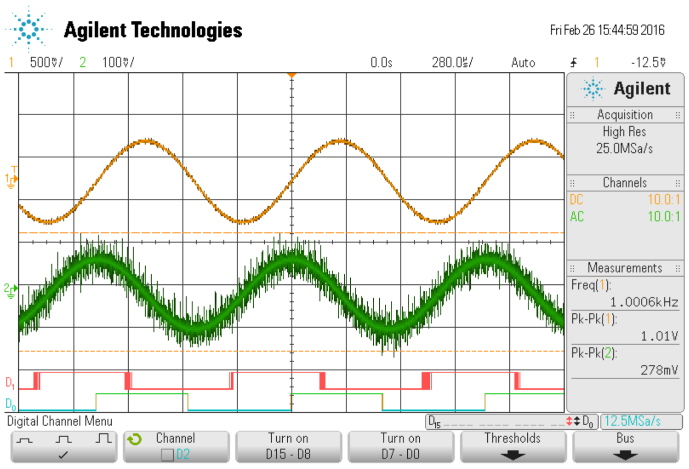
magnetic induction. The shielding of the cable is connected with ground and two signal wires are connected across sensor. The other step is to include a capacitor in the signal sensing circuit so that it will take the added load of acting as low pass filter. So, a capacitor is added in the feedback path of the opamp in both sig and ref path as shown in Fig.3.2. Filtered output is shown in Fig.3.6b using C_f=10nF.

3.2 Sine to square

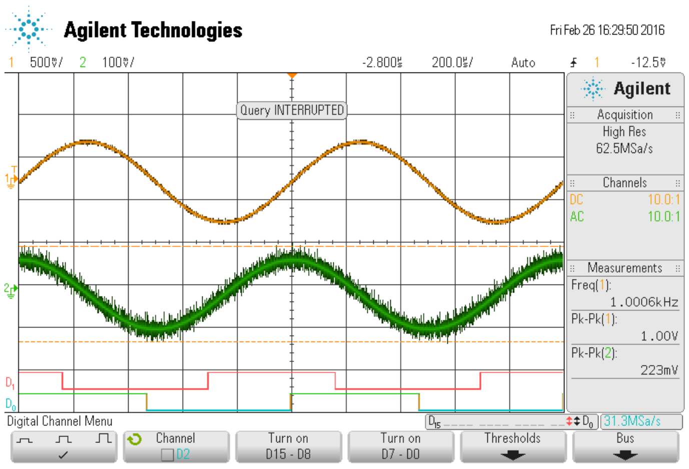
The output $s'$and r' from signal sensing block are sinusoidal voltage signal. The next step is to convert both the signal into corresponding square wave to measure the phase between the two. For this purpose, the block Sine to Square is used in both sig and ref path. The circuit diagram [15] of the the chip used is shown in Fig.3.7.

3.2.1 Statistical Analysis of Sine to Square block

A statistical analysis has been done on the conversion of sine to square wave. A sine wave signal from a function generator of 20kHz frequency is used as an input to the CD4069 chip. Data are recorded for 1 hour and is repeated 7001 times. Mean and standard deviation (SD) of the duty cycle of the square wave are tabulated in Table 3.1. The phase between input and output are also recorded. The mean value of the phase difference and its SD are also included in the table.



(a) Noisy *sig* signal corrupt the square wave



(b) Filtered signal output

Figure 3.6: Noisy and filtered signal output from both path: Channel 1 is ref and Channel 2 is sig

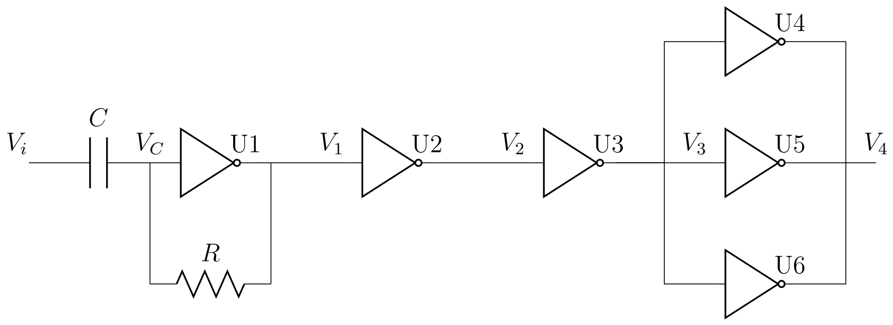


Figure 3.7: Sine to square using CD4069

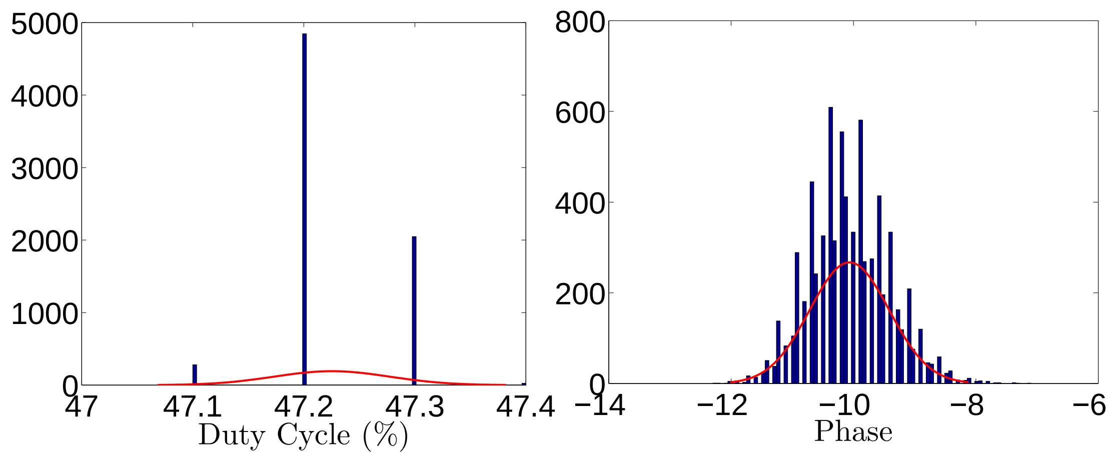


Figure 3.8: Distribution of Duty Cycle and Phase

Measure	Mean	Minimum	Maximum	SD	% of deviation
Duty cycle (%)	47.226	47.0	47.4	0.0523	0.1108%
Phase (in - out)°	-10.0665	-14.5	-3.4	0.6501	6.4576%

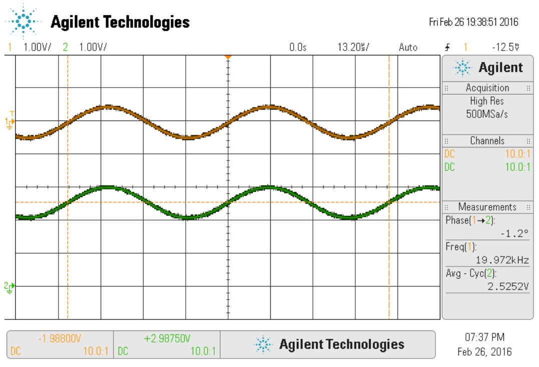
Table 3.1: Statistical analysis of sine to square

A histogram plot of duty cycle and the phase of the chip are shown in Fig.3.8. From figure and table it seems that the chip has a phase of $-10.0665°$. This phase of the chip will appear both in the sig and ref channel and mean offset will be nullified. But, the maximum deviation of phase in differential condition will be $2 \times (2 \times 0.6501° = 1.3°) = 2.6°$.

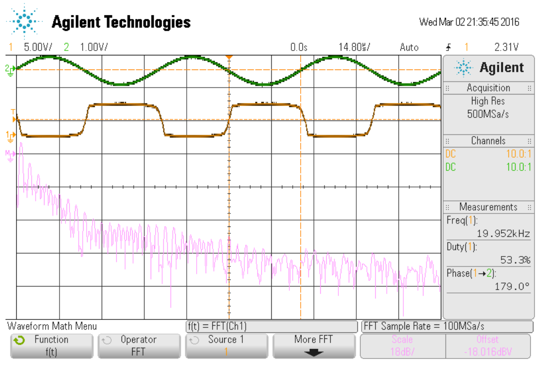
A thorough study of the chip has been done by observing the output with respect to the input at every step using Agilent MSO . Input used is a sine wave signal V_i from a signal generator of frequency 20kHz. Output voltage V_C from the capacitor with respect to the input is shown in Fig.3.9a. It can clearly observed that the signal V_C has $2.5252Volt$ offset. The output from $U1$ shows a square wave signal but does not have sharp edges as observed from Fig.3.9b. Similarly, output at every step are observed and shown in Figs. 3.9c, 3.9d and 3.9e. Nature of the output has been improved by using inverters. Three inverters are used in parallel to introduce sharp change at the transition of the square wave.

3.3 Phase detection circuit

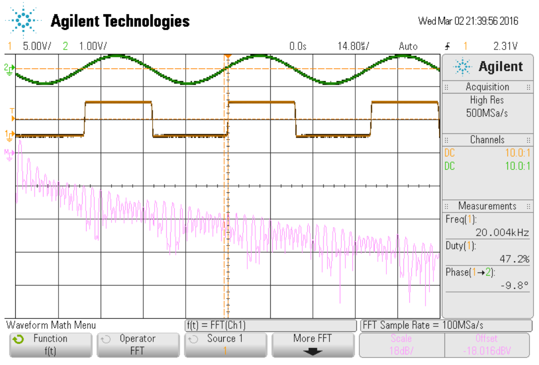
The next is the phase detection block whose input will come from “Sine to Square” block. The output of this block is a square wave voltage signal whose width is



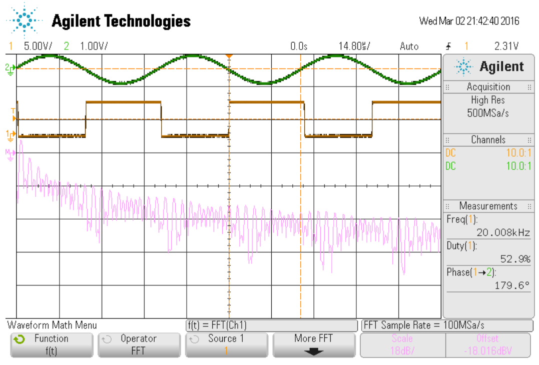
(a) Channel 1 is V_i and Channel 2 is V_C



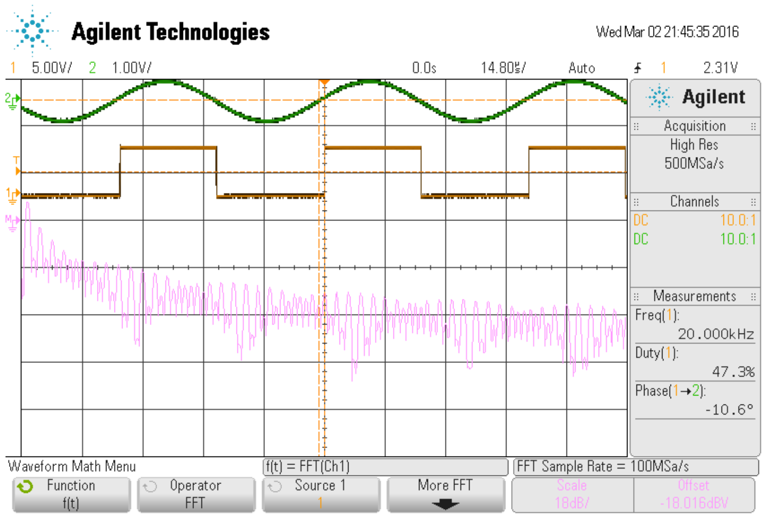
(b) Channel 1 is V_1 and Channel 2 is V_i



(c) Channel 1 is V_2 and Channel 2 is V_i



(d) Channel 1 is V_3 and Channel 2 is V_i



(e) Channel 1 is V_4 and Channel 2 is V_i

Figure 3.9: Intermediate output of sine to square

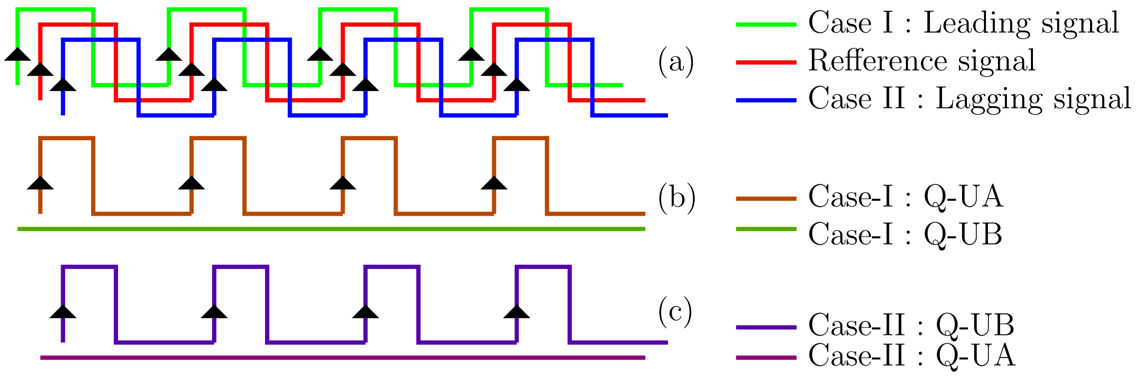


Figure 3.10: Logic of phase detection with respect to input and its corresponding output wave form

proportional to the phase difference of sensor signal, referred as s'', and reference signal, referred as r''. The phase difference between the two may be a leading or lagging one, as shown in the Fig.3.10(a). Case I shows a leading or negative phase where r'' becomes HIGH (or 1) after a delay with respect to HIGH (or 1) s''. On the otherhand, Case II shows lagging or positive phase where the condition becomes reversed. Here s'' is a delayed signal with respect to r''. The negative and positive phase detection has been done using two D flip-flops, named as UA and UB respectively. The output of the Q-UA will be HIGH (1) and Q-UB will be LOW (0) when the phase is negative and for positive phase output of Q-UB will be HIGH (1) and that of Q-UA will be LOW.

The logic for identifying the negative phase is shown in the Fig.3.10(b). The output of Q-UA will be HIGH when both s'' and r'' are HIGH. The width of the pulse is a measure of leading phase. In this condition D-flipflop UA will be activated if it gets HIGH s'' signal first then HIGH r''. On the otherhand, the logic of positive phase identification is shown in Fig.3.10(c). The output of Q-UB will be HIGH when again both s'' and r'' is HIGH and in this case also the width is the measure of phase. But, in this condition UB will be activated if it gets HIGH r'' first and then HIGH s''.

The circuit diagram of the two connected flip-flops is shown in Fig.3.11. Property of the connection of the two flip-flops, UA and UB, is such that once one flip-flop is activated the other will be deactivated. This will be decided by the fact that which one is getting HIGH $\overline{CLR}$ first. If the sensor signal s'' is leading then it will be high before the reference signal r'' so, it is required to identify of the two positive going edge which one appears first and have to activate or deactivate the flip-flops accordingly. For this case because first $\overline{CLR}$ of UA is HIGH while that of UB is LOW then UA will be activated while UB remain deactivated. So,

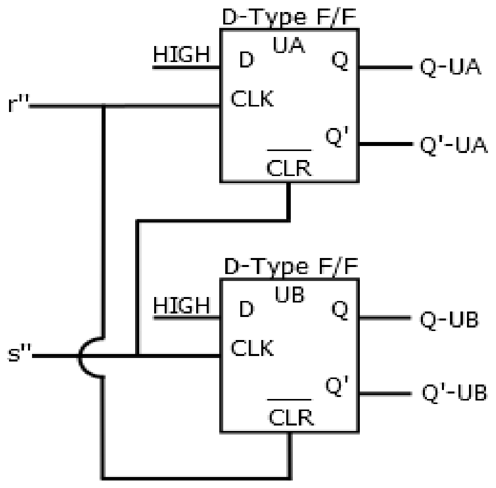


Figure 3.11: Phase detection circuit using two D flip-flops

Table 3.2: Signal Leading

		UA			UB		
Reference r''	Signal s''	$\overline{CLR}$	CLK	**Q-UA**	$\overline{CLR}$	CLK	**Q-UB**
0	0	0	0	**0**	0	0	**0**
0	1	1	0	**0**	0	1	**0**
1	1	1	1	**1**	1	1	**0**
1	0	0	1	**0**	1	0	**0**

the output of UB will be LOW during this period. When both r'' and s'' are HIGH then $\overline{CLR}$ of UA flip-flop being HIGH and CLK also becomes HIGH the output will be HIGH. It will remain HIGH till s'' becomes LOW. The truth table of the flip-flop of both UA and UB are given in Table.3.2 and 3.3 for leading and lagging phases respectively. Above logic originated from and changed into single chip device.

3.3.1 Simulation of the logic

The logic developed has been tested by simulating the output as shown in Fig.3.12. Input waves, referred as reference and signal, as connected to the flip-flops are shown on the screen of the scope. The plots of UA-Q, UA-Q' and UB-Q, UB-Q' are also shown on the scope. Characteristic of all the four outputs $phase_N$, $phase'_N$, $phase_P$, $phase'_P$ are obtained by passing UA-Q, UA-Q', UB-Q and UB-Q' outputs through a simulated low pass filter. The plots of all four outputs is shown

Table 3.3: Signal Lagging

		UA			UB		
Reference r''	Signal s''	$\overline{CLR}$	CLK	**Q-UA**	$\overline{CLR}$	CLK	**Q-UB**
0	0	0	0	**0**	0	0	**0**
1	0	0	1	**0**	1	0	**0**
1	1	1	1	**0**	1	1	**1**
0	1	1	0	**0**	0	1	**0**

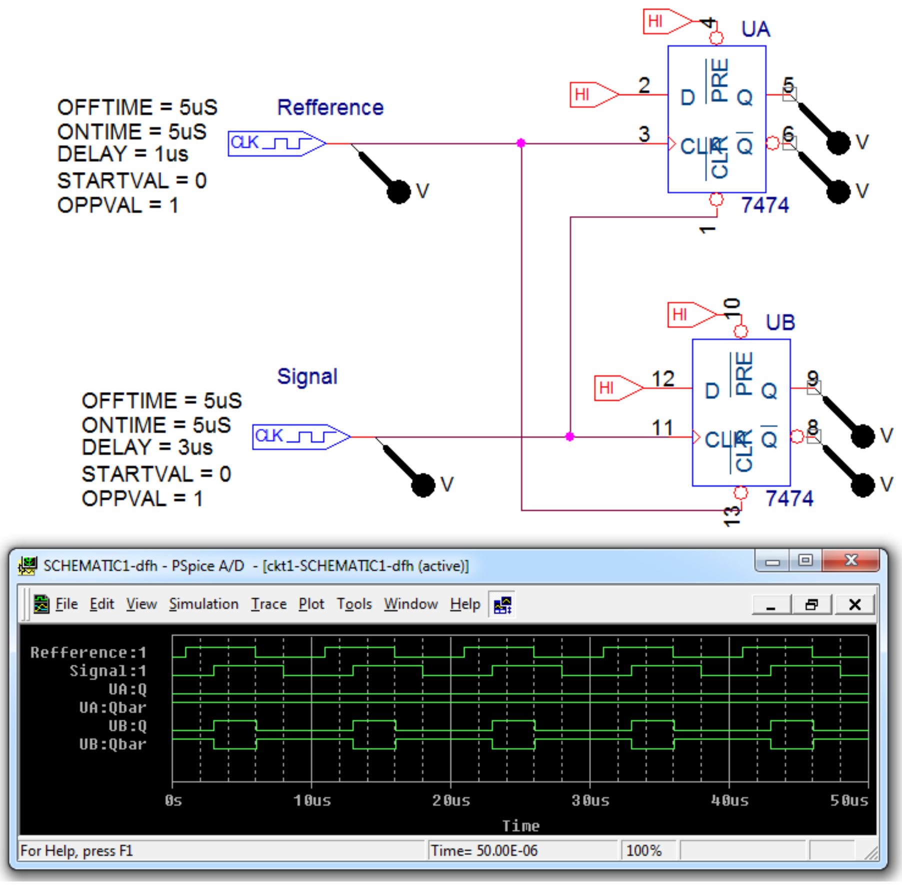


Figure 3.12: Simulation of phase detection logic

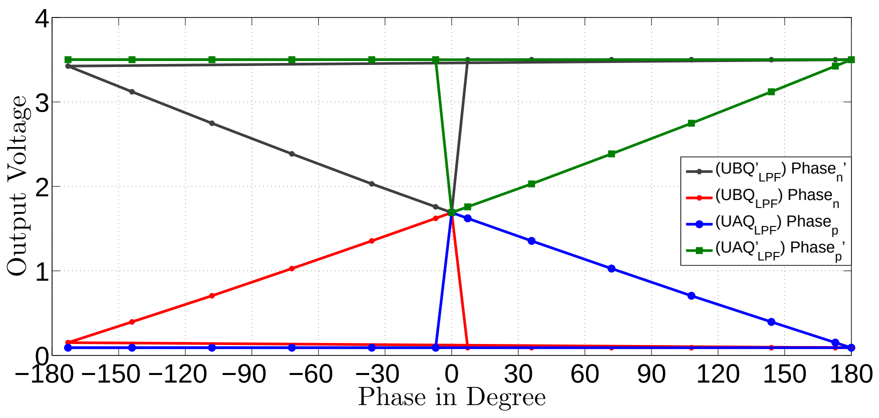


Figure 3.13: Characteristics of all four outputs in terms of voltage with phase $\pm 180°$

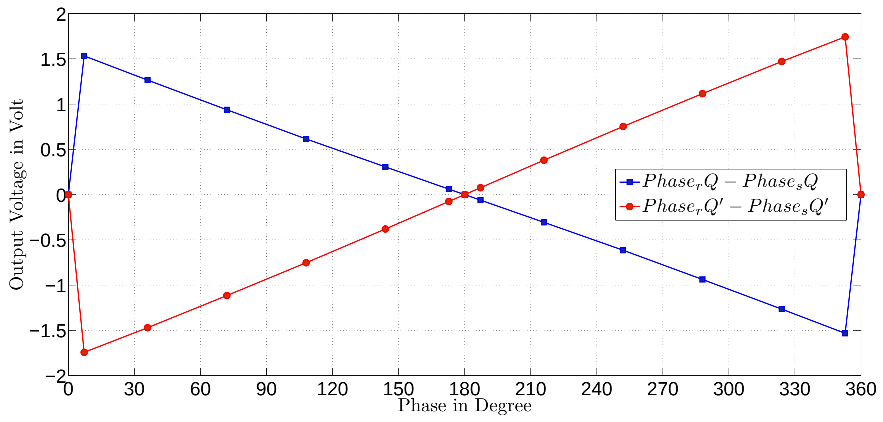
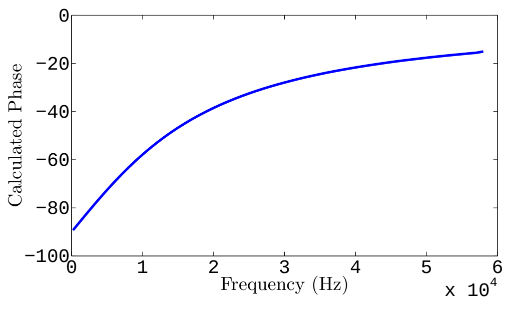


Figure 3.14: Characteristics of outputs in terms of voltage with phase

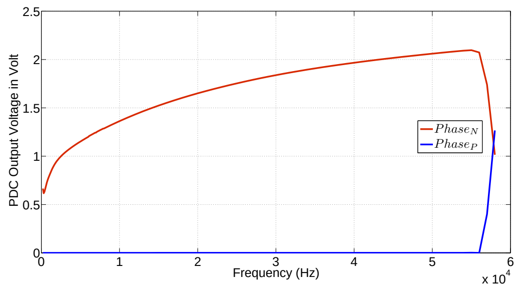
in Fig.3.13. From Fig.3.12 sensitivity will be obtained as $0.0094V/^o$. Now, to get a linear relationship between phase and voltage, voltage difference between Q and Q' of both the filtered output of UA and UB are plotted as shown in the Fig.3.14. This shows a linear relation but a sharp change near 0° and +360°.

3.4 Calibration and testing

The overall circuit with all the blocks as shown in Fig.3.1 is calibrated using a varying frequency sine wave as an input and connecting a RC circuit at the position marked as sensor as shown in Fig.3.2. The values of R and C are so chosen that the phase shows a linear relationship within this frequency range. The problem is



(a) Theoretical impedance Phase response



(b) PDC output with RC

Figure 3.15: Impedance Phase response of Series RC

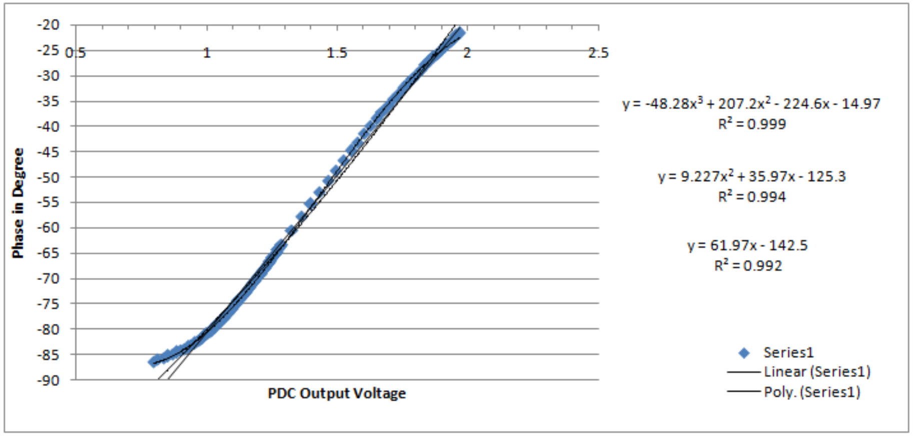


Figure 3.16: Calibration Curve

that impedance phase of RC is only positive.

3.4.1 Calibration with RC

To calibrate the circuit a series RC is used where the value of R and C are selected as $10k\Omega$ and $10nF$ respectively. Phase response for the frequency range of $200Hz$ to $58kHz$ is calculated and plotted in Fig.3.15a. For the same frequency range the phase values are detected using overall Phase Detection Circuit. The recorded values are plotted in Fig.3.15b. The output has a DC shift and shows near about linear change with frequency within 200Hz to 55kHz which is well beyond the operating frequency range and it fails above 55kHz.

By comparing Fig. 3.15a and 3.15b it may be said that it is following nature of the impedance phase of series RC. For fair comparison a calibration curve is drawn, shown in Fig.3.16. Some of the value are shown in the table 3.4. Curve fit equation are given in the Fig.3.16 where it can be observed that the goodness of

Table 3.4: Statistical analysis of sine to square

Frequency	$Phase_N$	Calculated phase
2000	0.9517	-82.84
6000	1.1999	-69.34
10000	1.363	-57.86
14000	1.4947	-48.66
18000	1.6032	-41.48
22000	1.6942	-35.88
30000	1.8385	-27.95
55000	2.0967	-16.14

fit value, R^2 is highest for third order equation. It is always better to use a linear relationship than high order polynomials and also after checking R^2 value which is also high enough for 1st order equation so, this one is selected as calibration curve. The value of x is nothing but the voltage so replaced by V.

$$Phase = 61.97V - 142.5; R^2 = 0.992$$

$$Phase = 9.227V^2 + 35.97V - 125.3; R^2 = 0.994$$

$$Phase = -48.28V^3 + 207.2V^2 - 224.6V - 14.97; R^2 = 0.999$$

3.4.2 Testing of different Impedance phase

The overall circuit is then tested with different passive elements. It is known that theoretically the impedance phase response of resistance is 0°, of capacitance it is (-)90°, for series RC it is increasing and for series RL it is decreasing. A choke coil is connected in series with resistance and different resistance values are used to get different R and L combination. The circuit is tested with all these elements. The frequency range used is 4kHz to 8kHz. The result is shown in Fig.3.17. It may be said that the circuit almost follows the nature of responses.

3.5 Practical implementation

The circuit then has been tested for total operating frequency range so as to check the output for the next step of operation. The output of this circuit will now be connected to the microcontroller for recording and display of the phase response of the sensor. For this purpose, the readings are recorded at every stage using a 10nF capacitor as a sensor, shown in Fig.3.18. The error in the output of signal sensing circuit is eliminated by the next block, "Sine to Square". In the lower frequency,

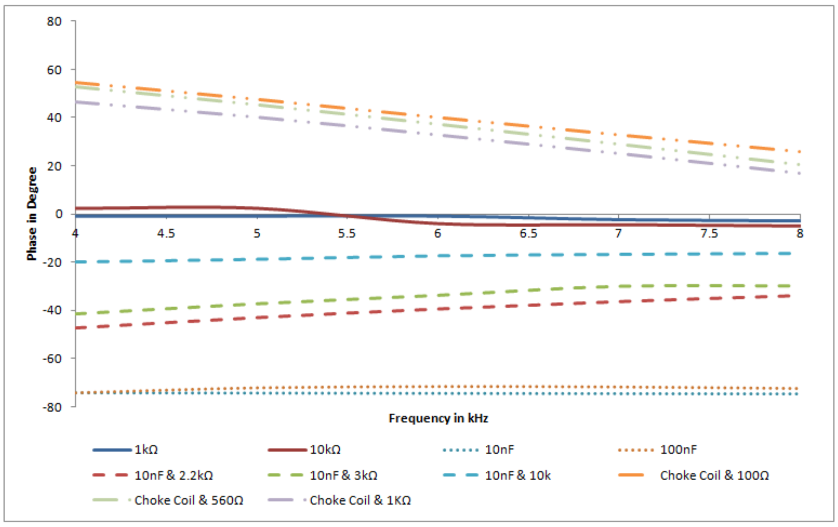


Figure 3.17: Impedance phase reading of R, C, Series RC, Series RL

error is high but a fixed shift in the value is observed in the higher frequency while the output of the overall circuit represented by 'Measured phase' shows a slow increasing trend hence not able to generate good result. It may be due to the noisy sensor signal which have several zero crossing near the actual zero crossing of the wave, as mentioned in Section 3.1.3. This may affect the conversion to square wave which on turn to the flip flop output because the flip-flops getting positive edge several times. To find out the problem the outputs are shown in the Fig.3.19. Upper most sine wave trace (yellow trace) is r', next lower trace (green trace) is s', next one is $D11$ (red) is of s'', $D10$ (bluish green) of r'', $D9$ of UA-Q' and $D8$ shows UB-Q. This problem may be solved if phase difference is used as shown in Fig.3.14. In that case a subtracter has to be used which will make the circuit more complicated and addition of another error due to the inclusion of the circuit. If both the output of UA and UB are fed into two channels of microconctroller through its Analog to Digital Converter (ADC) then it will take care of the noise problem. It will ignore the value if input from one channel is below a threshold value.

3.6 Conclusion

The phase detection block has been implemented and calibrated. It has also been tested with passive elements of known phase impedance. The problems are also encountered and solved using microcontrollers. The output of this block then used for the next step for recording and display of the data.

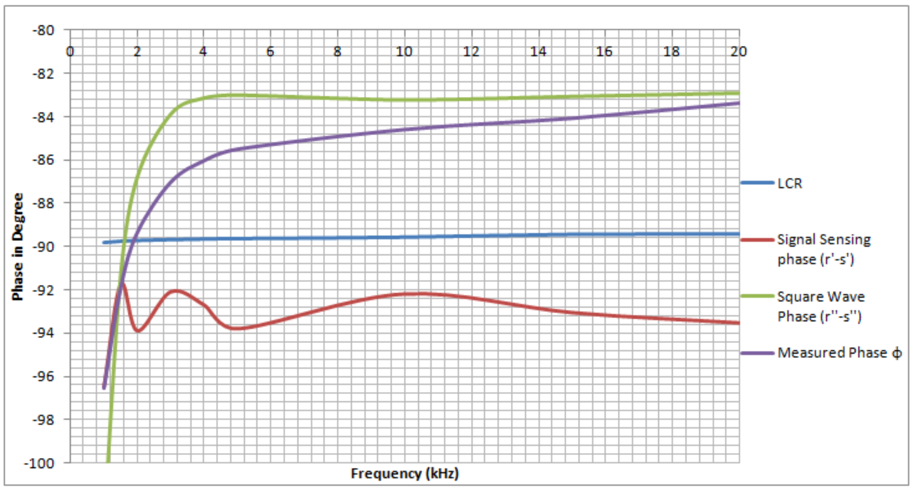


Figure 3.18: Signal Sensing, PDC output with C

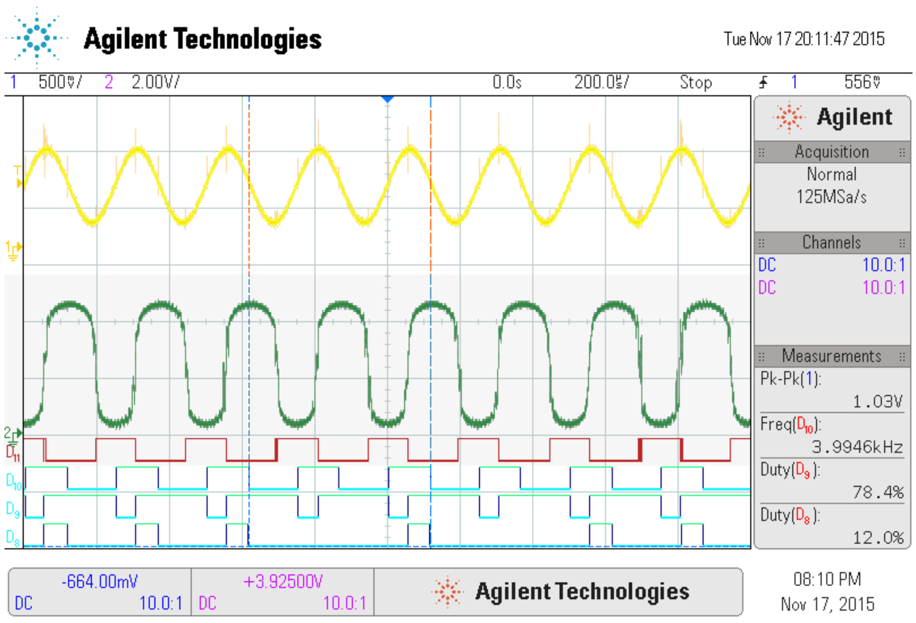


Figure 3.19: Square wave noise effect on phase detector

Chapter 4

Instrumentation system

This chapter deals with the implementation and description of overall instrumentation system for measuring impedance phase response of a piezo based sensor in the frequency range of 1kHz to 20kHz. It may be used to measure the dental implant stability by tracking shift of the resonance frequency with time from the recorded phase response for a specified time interval. The instrument is designed and developed to display the phase response graphically. It can also save the data in a Laptop or Personal Computer if required. The circuit developed and reported in Chapter 2 and 3 are connected and assembled to perform the overall task. The additional segments for the implementation are explained in details. MCU code development has been started initialy with dsPICDEM2[16] board and its example software.

The block diagram of the total instrumentation system is shown in Fig.4.1. The block consists of a microcontroller unit represented as MCU, a Digital to Analog Converter represnted as DAC, a phase detector circuit represented as PDC, a Graphical Liquid Crystal Display unit represented as GLCD and an extra optional peripheral like Laptop or PC to store the data for further analysis. The sensor position is shown in the figure.

4.1 Working Principle

The steps used for measuring phase response are listed below:

1 Initialization of the MCU includes specification of the range of frequency for which it will display or record the data and other necessary commands to run the MCU.

2 MCU will then calculate an array of values of sinusoidal function of one cycle of a particular frequency, save it as look up table and generate a digital sine

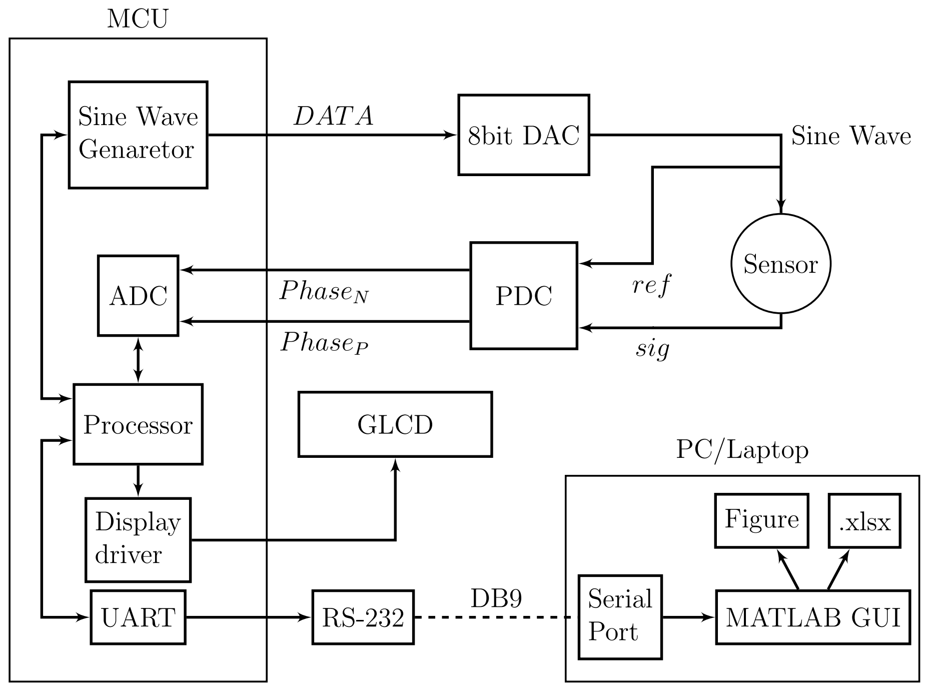


Figure 4.1: Block diagram of a Standalone instrument for measurement of phase response

wave as reported in details in Chapter 2.

3 This digital output is fed to a DAC.

4 The analog output is then fed to Phase Detector Circuit (PDC). PDC will generate a voltage with respect to a reference signal which will be proportional to the phase (positive or negative) of the sensor.

5 Two signals from PDC is fed back to MCU through its in built ADC. The MCU processor will calculate phase from two signals and also identify whether it is positive or negative.

6 The calculated phase value together with its sign is parallely converted into RS232 signal through UART for storing and send to GLCD for display.

7 The loop rotates starting afresh from step 2 with new frequency and continues until it reaches last value of the specified range.

8 At the end when frequency value is the last value of the range MCU calculate resonance frequency, call UART write and send data for display.

9 MCU then will go to the idle mode for power saving.

From the steps stated it is evident that this is a combination of software that are implemented by the MCU and hardware for which circuits have been developed. So, the two are to be connected to develop the total instrument. For this, there should be a handshaking between the two which is also controlled by MCU.

4.2 Complete hardware

The components as shown in Fig.4.1 are connected in a single board to make the instrument standalone. In addition some more components are required to complete the hardware of the instrument.

4.2.1 MCU pin arrangement

Pin diagram of dsPIC30f4013 microcontroller unit (MCU) is shown in the Fig.4.2. The number of pins are 40 as shown as CN1 and CN5. A jumper P1 is connected to provide a provision to disconnect pins RB6 and RB7 if and when required.
Pin RB0 to RB7 are used for sending the computed sine data to DAC and also to GLCD as these two are not working simultaneously. For this jumper P1 is used to disconnect DAC and GLCD during dumping of code to MCU.
Pin RB9, RB10, RC13, RC14, RD0, RD3 are connected to R/W, RS CS1, CS2, E and control pin RST of GLCD respectively.
Pin RB11 and RB12 are connected to the output of PDC named as Phase-s and Phase-r respectively.
Pin UartRx and UartTx are connected to RS232 for communicating information to outer world.
Over and above the MCU has 4 external interrupt pin, those are MCLR (pin 1), INT0 (pin 17), INT2 (pin 18) and INT1 (pin 23) and these pins may be connected to push button for user interrupt. Among them pin MCLR is the reset. The user input connection is shown in Fig.4.3.

RF1, RF3, RF6 , RF2 are kept open to connect SD card to store data in future. It is not included here but space and circuit of card connection is shown in Fig.4.4 for future development.

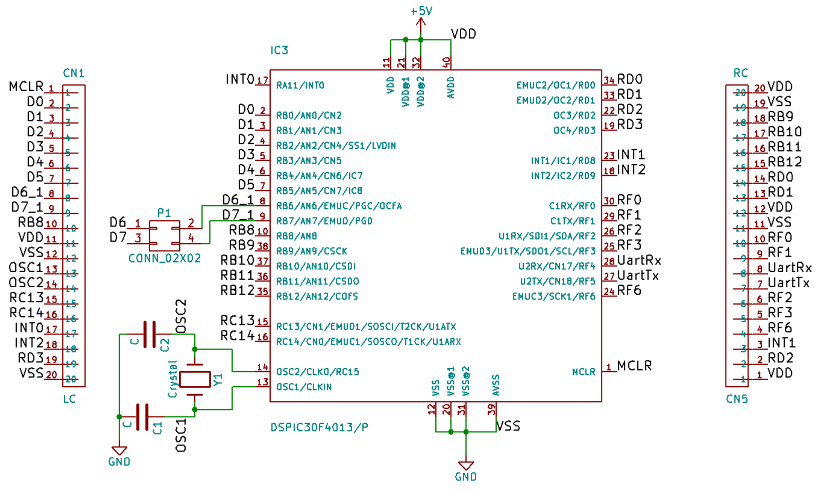


Figure 4.2: MCU pin diagram

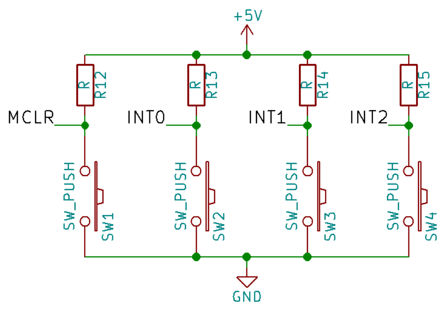


Figure 4.3: User input

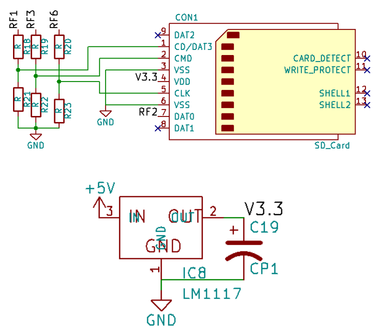


Figure 4.4: SD card arrangement

4.2.1.1 Oscillator

A 7.3728MHz crystal oscillator with the specified components as 22pF capacitor is connected to MCU according to the specification supplied by the manufacturer as shown in figure 4.2. Pin number 13 and 14 are used for connecting the oscillator.

4.2.2 Generation of sinusoidal voltage signal

The data generated from the computed sine function by MCU are saved as look up table as explained in Chapter 2. To out these data, Pin numbers RB0 to RB7 of MCU are connected to the A1 to A8 of DAC. The DAC output is then connected to signal processing blocks as explained in Chapter 2 and also shown in the Schematic diagram.

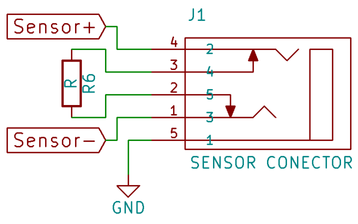


Figure 4.5: Sensor Connector

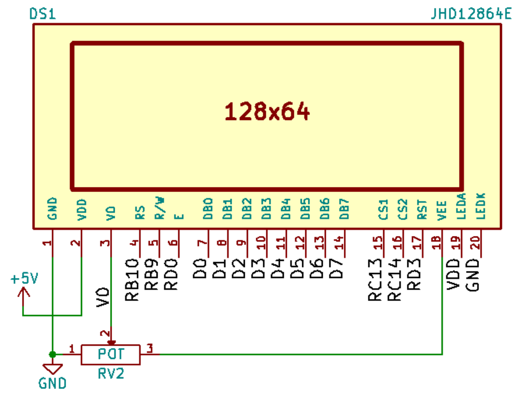


Figure 4.6: GLCD Connection

4.2.3 Circuit for measurement of Impedance Phase Response

The generated sine wave is used as an input to the phase measurement circuit which is again explained in details in Chapter 3. Here a jumper CN2 is used to disconnect this if and when required. In the circuit other four jumpers CN4, CN6, CN8 and CN9 are used for the same purpose. The sensor is connected in this block, connection is shown in the Fig.4.5. A 3.5mm stereo jack connector is used to reduce noise and plug and play feature of the sensor. Shielding is connected to ground to remove noise and two core wires are connected with the sensor. When jack is not connected with the socket then the device is connected across a resistance R6 as shown in the figure and the output will show zero phase.

4.2.4 Graphical Liquid Crystal Display (GLCD)

A 128×64 dots graphical Liquid Crystal Display unit, JHD12864e GLCD[17] is connected to MCU. The total display consists of two chips both are of 64×64 dots and are placed side by side so as to form left part and right part. Each chip has 8 pages. x-position is selected by selecting first the chip, left or right, then the exact position while y-position is selected by selecting the chip again, exact position and the page number to be used. Diagram is given in Appendix A.1.

This GLCD will display the measured impedance phase response graphically. It is a 20 pin chip whose 8 bit data bus and six control pins are connected to MCU pins mentioned in Section 4.2.1. Of the six control pins CS1 and CS2 are horizntal chip select pin, RS is data / instruction pin, R/W is Read/Write pin , E is enable pin, and RST is reset pin however RST pin does not follow conventional rule. RST zero means it will remove every thing and one means it is in activated mode

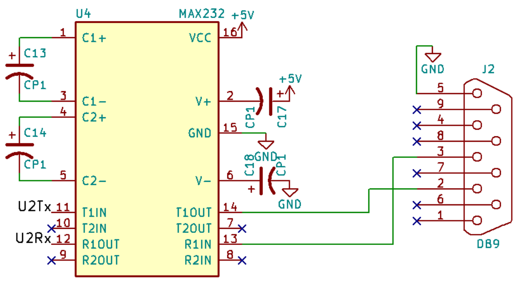


Figure 4.7: UART to DB9 Connector

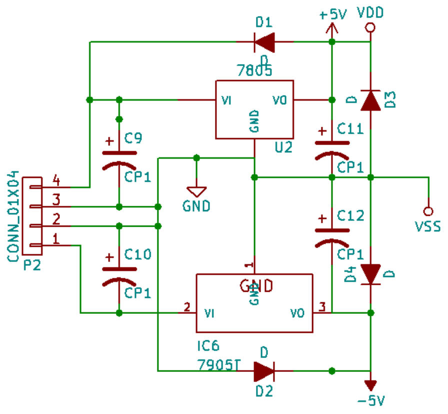


Figure 4.8: Power supply

though not documented in datasheet. Pin 1 and 20 are connected to ground and pin number 2 is connected to VCC. V0 and VEE are connected to a potentiometer and then to ground to vary the contrast of the GLCD. The connection of GLCD is shown in Fig.4.6.

4.2.5 PC or Laptop Interface

A large data saving feature is not included in the MCU device. This facility if added into the instrument then it will help for further analysis because calibration and application both require post data analysis. Considering the requirement, UART provision of MCU is utilized through RS232 cable or serial port. It will communicate with PC or Laptop. For the purpose, it needs a transmitter from the MCU device side and a receiver from PC or Laptop side.

4.2.5.1 MCU Device side for interface

dsPIC30f4013 has UART module for communication. But, UART signal is not suitable for direct communication with Laptop or PC. UART pins as mentioned in Section 4.2.1 is connected to RS232. Its output is connected to a serial port DB9 as shown in Fig.4.7.

4.2.5.2 PC side for interface

The data from DB9 connector may be connected directly to the serial port of the PC. In fact no extra hardware is required for this communication only a Graphical

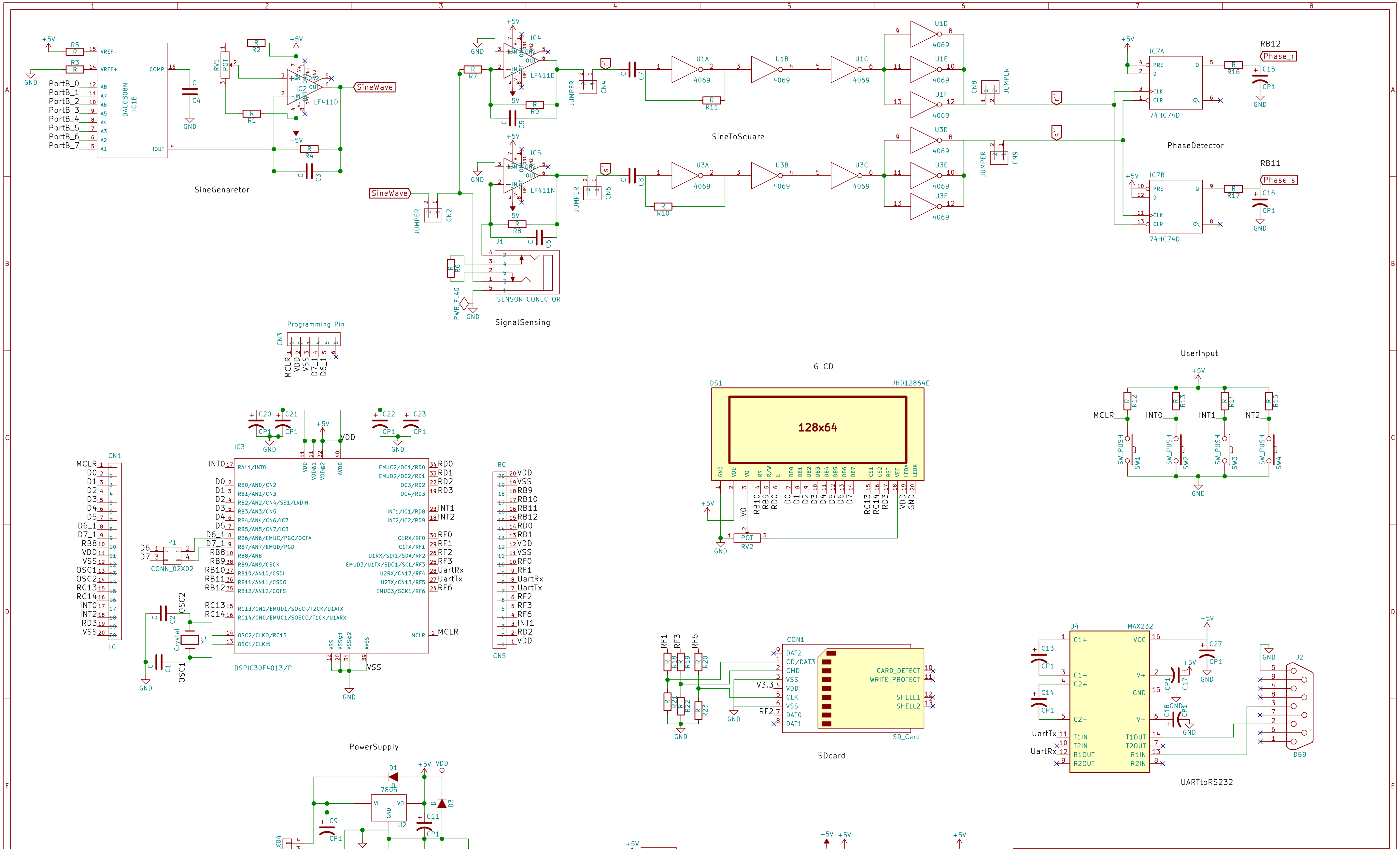
User Interface (GUI) software will receive the signal from the port. MATLAB software has the provision of designing GUI.

4.2.6 Isolation from power line

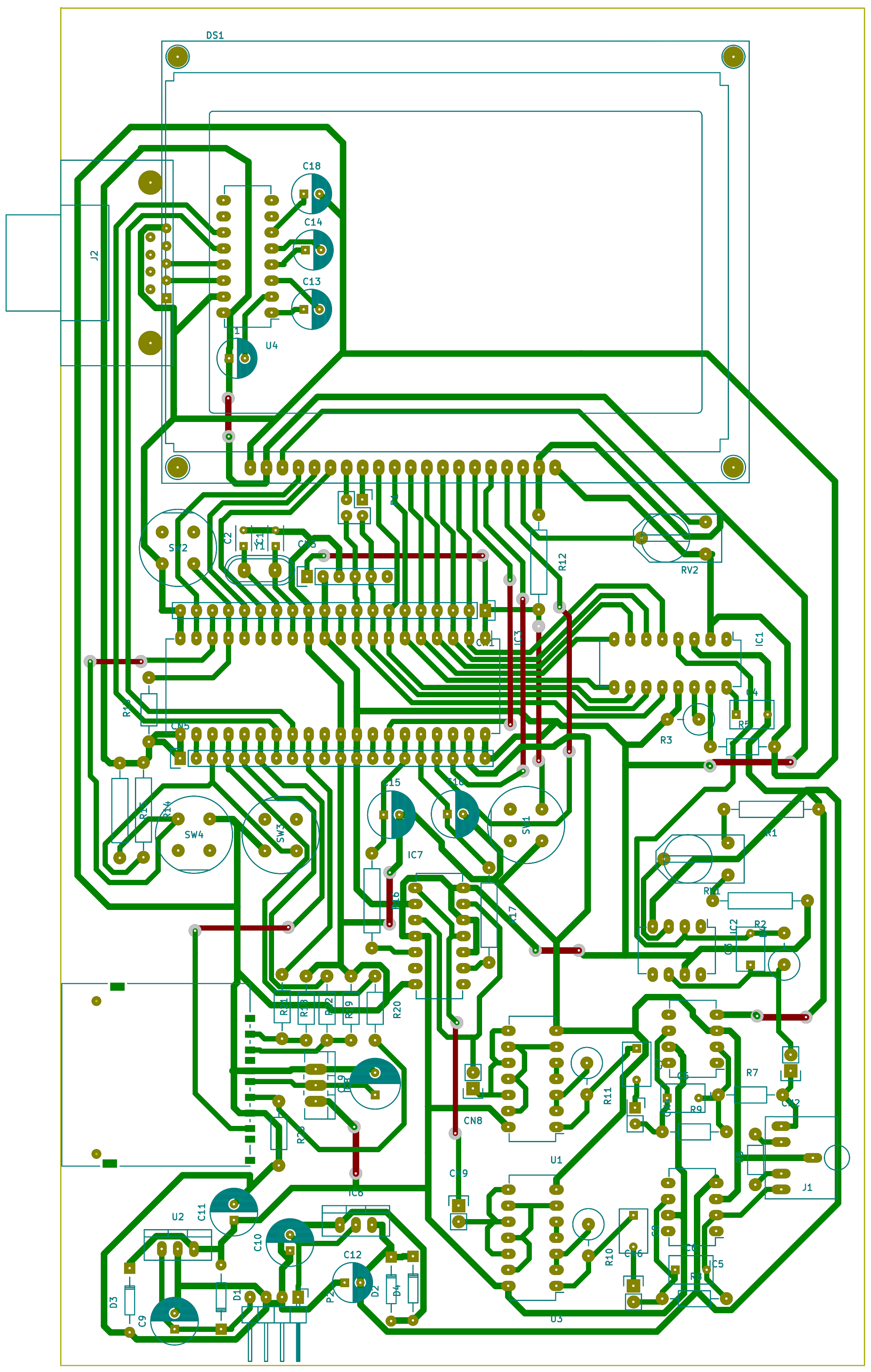
The device should have isolated power supply when it is in contact with a human body. But the circuits needs supply of ± 5 volt which has been provided using 9volt battery. Circuit diagram for power supply is shown in the Fig.4.8. And for data storage instead of connecting DB9 to PC it may be connected to a Laptop.

4.3 PCB Design and Fabrication

PCB has been designed using Ki-Cad software and fabricated in house by using negative photo resist and $FeCl_3$ etching. The schematic diagram and PCB layout are shown the figures.



Schematic
Somen Biswas (M.tech IEE)
Under the guidance of Dr. Bhaswati Goswami
Jadavpur University
Sheet: /
File: pdc.sch
Title: Standalone impidance phase
Size: A3 Date: 2015-12-16 Rev: A
KiCad E.D.A. kicad 4.0.1-stable Id: 1/1



PCB Design
Design and developed by Somen Biswas
Under the guidance of Dr. Bhaswati Goswami

IEE, Jadavpur University

Sheet:
File: pdc2.kicad_pcb

Title: Standalone instrument for determination of Resonance Frequency

Size: A4	Date: 2015-12-26	**Rev:**
KiCad E.D.A. pcbnew 4.0.1-stable		Id: 1/1

4.4 Software implementation

The flowchart of the implementation of the software to be incorporated into the MCU is shown in the Fig.4.9. From the figure it is clear that the total process of measurement will be controlled by MCU. Sometime it will be directly in action and sometime it will control peripheral device to act accordingly. Now, to implement all the work assigned to MCU it is required to configure and initialize all the devices to be used by the MCU.

4.4.1 Configure and initialise ADC

ADC used here is an in-built one and its reference voltages may be connected to pins AVCC and AVSS. In this case the two are connected to 5 Volt and ground respectively. The flowchart to read phase value from PDC is shown in Fig.4.10. Reading phase voltage can be divided into 3 segments:
1) Configure and initialize ADC,
2) Start Sampling data : this instruction will come from MCU,
3) Convert and Store data : after sampling it will convert and send the data automatically to MCU.

4.4.1.1 Configure ADC

Following steps are used to configure ADC.

•Select input channel on ADCSSL register
  -Make port AN11 (Pin 36) input ADCSSLbits.CSSL11=1
  -Make port AN12 (Pin 35) input ADCSSLbits.CSSL12=1

• Select A/D conversion clock
  -Set up Acquisition time (Tacq) for 31 TAD periods using ADCON3bits.SAMC=31.
  -A/D Conversion clock select bits selected as 7(000111_2) using
  -ADCON3bits.ADCS=7

• Select ADC mode
  - Make 111 to conversion starts automatically after sampling using ADCON3bits.SSRC=7(111_2)

• Configure A/D interrupt
  -Clear ADIF bit IFS0bits.ADIF = 0.

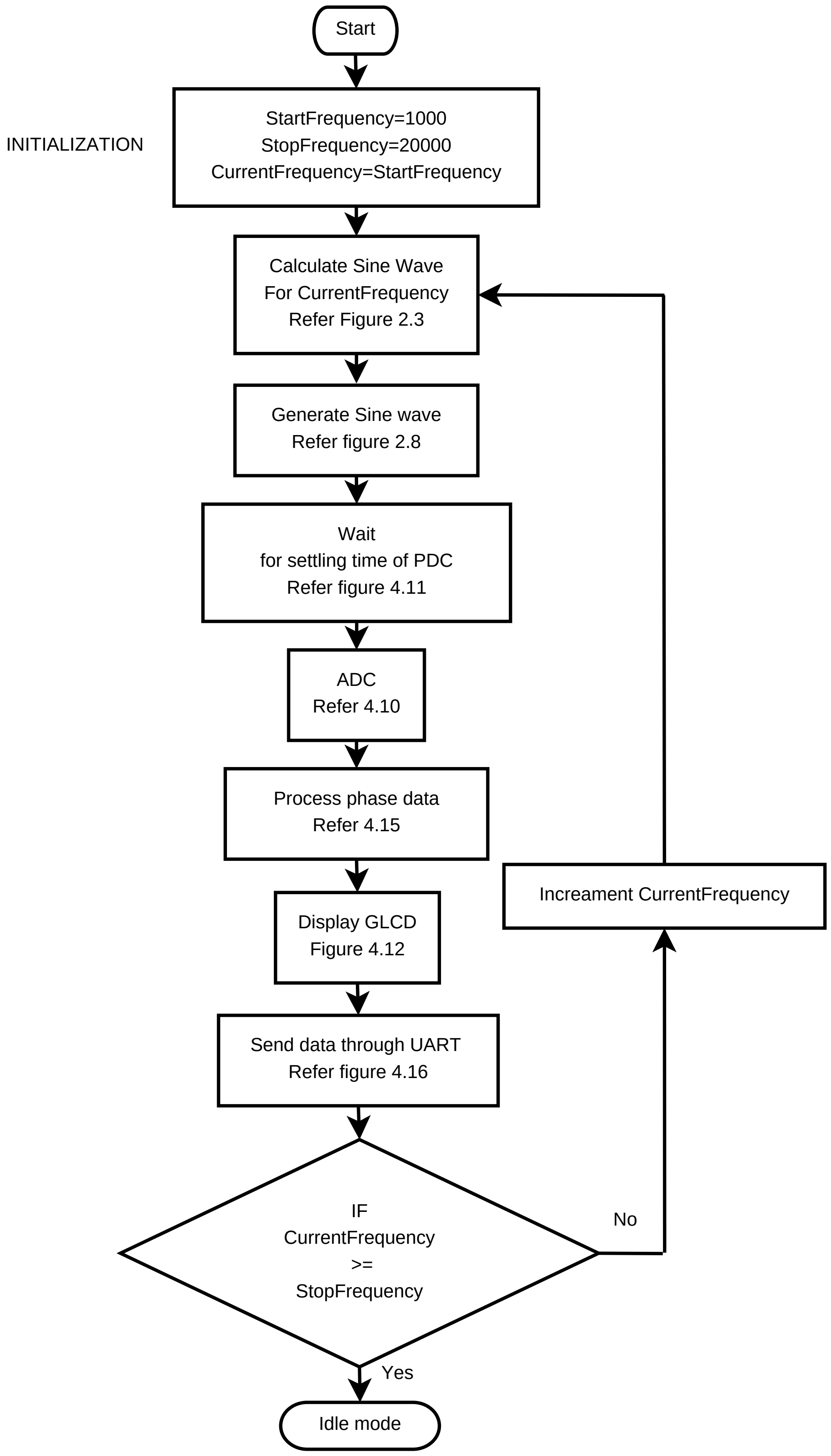


Figure 4.9: Algorithm for measurement of Phase response of a piezo based sensor

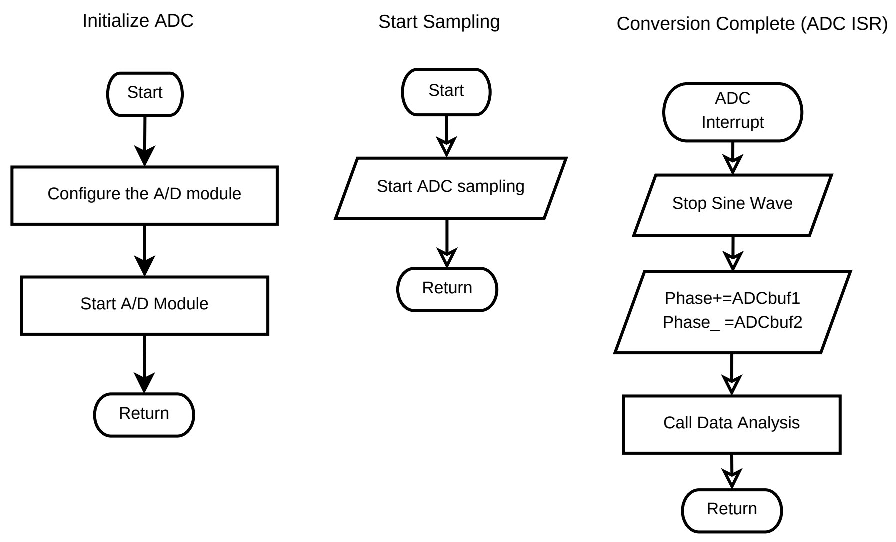


Figure 4.10: Flow chart of action of ADC

- Set ADIE bit (for ISR processing) IEC0bits.ADIE = 1.
Now ADC is ready to receive instruction from MCU.

4.4.1.2 Instruction to start sampling

To instruct ADC to start sampling the flowchart used is shown in Fig.4.11. This is implemented using timer 2 module of MCU. Following steps are used

1. Initialize Timer by setting T2CONbits.TON=1;
2. Setting PR2
3. Setting Interrupt Service Routine (ISR) function by which MCU will send instruction to ADC to start sampling both channels using the command ADCON1bits.SAMP=1.

4.4.1.3 Conversion of data

After completion of sampling ADC module starts conversion automatically. After conversion completed ISR function will instruct to stop to send the data of sine function by making zero of the TON of T1CON control register. ADC peripheral will store the result in ADC buffer register ADCBUF0 and ADCBUF1. Now the data needs to be stored in specified registers of MCU, named as phaseP and PhaseN. Another important task is to clear the interrupt flag register otherwise the ISR function will continue in a loop. The resolution attained is $\frac{5V}{4095} = 1.221mV/bit$.

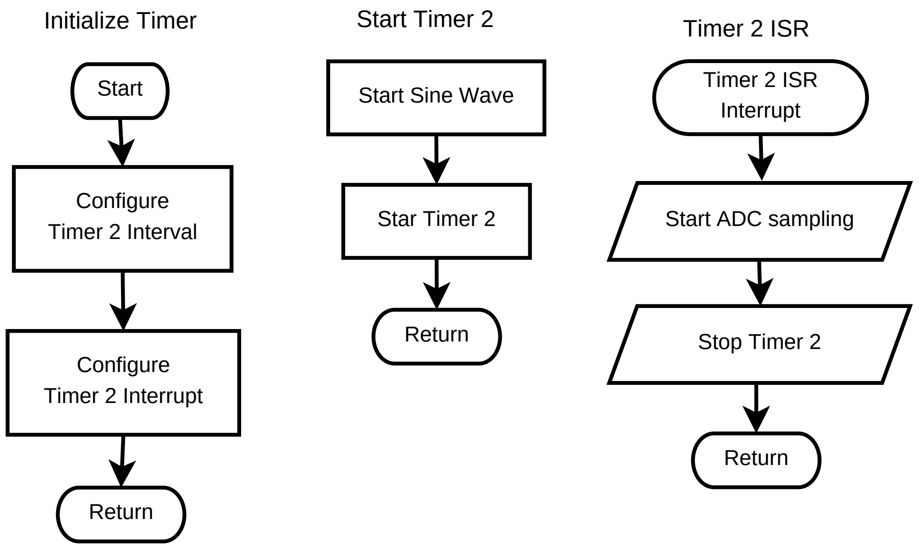


Figure 4.11: Timer setting for instructing ADC to start of data sampling

4.4.2 Configure and initialise GLCD

To configure and initialize GLCD following tasks are to be performed. And after every task the function will be enabled using 'Enable Function'.
1) Set six control pin in output mode.
2) Switch on GLCD according to the instruction sheet using GLCDon function
3) Clear GLCD

4.4.2.1 Enable Function

To write any instruction or data in GLCD there is a prescribed format which is given in AppendixA.2. After that it is required to set Enable pin to receive data from control pin and data bus. In timing diagram, given in AppendixA.3, it is stated that after Enable pin becomes high data write time T_{WH} should remain high for atleast 450ns. So a delay function is introduced in MCU of the order of μs.

4.4.2.2 Switch on GLCD

GLCD is first switched on using the instruction given in datasheet. To do so, data 0011 1111 is written through data port. Both RS and RW are set as 0 and followed by an enable function.

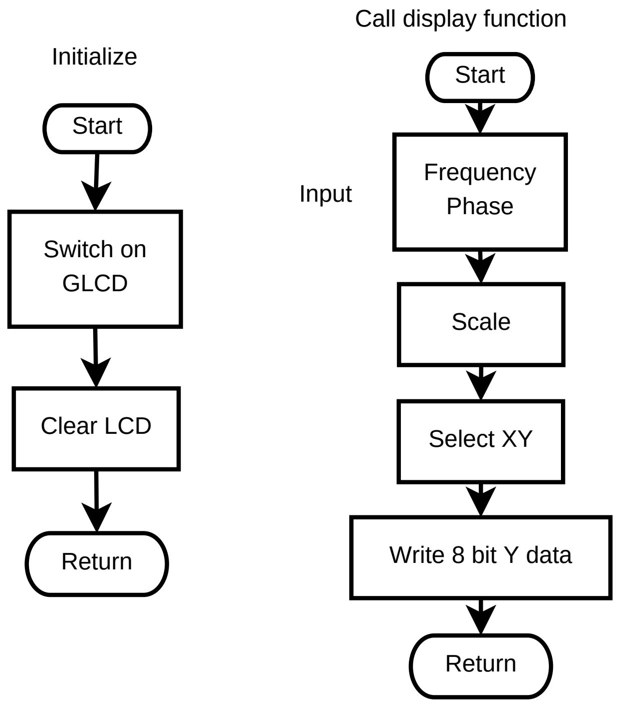


Figure 4.12: Flow chart for display by GLCD

4.4.2.3 Clear GLCD

Clearing GLCD can be done by making zero (dark spot) to all the dots of GLCD. To do so, it is required to select X value and Y pages of GLCD and put the data as 0. The used steps to clear it is given below.

- Select both the chip CS1 and CS2 by making these port 1
- Select x=0
- Select y page one after another upto 8 pages
- Write zero through data bus
- Make RS one and RW zero to instruct GLCD for data write mode
- Call enable function

4.4.3 Configure and initialise UART and RS232

UART2 module is selected for the purpose and baud rate used is 9600 bit/sec. Flow chart to initialize UART is given in the Fig.4.13.

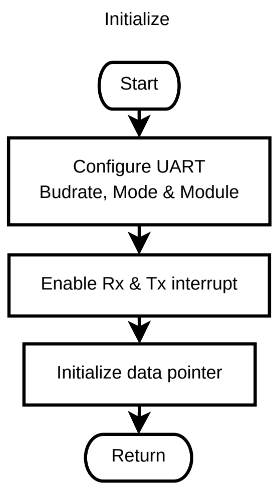


Figure 4.13: Initialize UART

4.4.4 Main program as controlled by MCU

The total measurement process will be controlled by MCU as mentioned above. The steps that are to be followed are listed in details below.

1) Initialization of necessary set up of MCU
2) Initialization of MCU for generation of sine wave
3) Initialization of MCU Timer for Settling time
4) Initialization of ADC by MCU
5) Initialization of Display by MCU
6) Initialization of UART by MCU
7) MCU Start calculating sine
8) MCU send data to DAC
9) MCU send instruction to ADC to start sampling and then auto conversion of data by ADC
10) MCU will receive data from ADC
11) MCU will process data
12) MCU will send data to GLCD and UART

13) Go to infinite loop

Initialization of steps 1 to 6 are discussed in above sections. 7,8 and 9 are also discussed in details. 11 and 12 will be discussed in the later part of this section.

4.4.5 Time of assigned task of MCU

To execute the above said process systematically by MCU, some of the block needs to work simultaneously and some of them can work sequentially. It is required to arrange those work properly in the time line. The timing diagram is shown in the Fig.4.14. The total time for executing a loop will be divided into execution of a single or multi-tasks. These are listed below.

- t_{cs} is the time required by MCU for computation of an array which will contain a cycle of sine values at a particular frequency. This computed values are stored in a look up table. The duration t_{cs} depends on the number of samples, N in a cycle. This value of N is used to get different frequencies as mentioned in Chapter 2.

- t_s is the time during which MCU will perform the sole function of sending 8bit data parallely to DAC from the look up table one by one with a constant time interval δt as reported in Chapter 2. The data should be sent circularly to generate cycles of sine wave. To do so, after sending the last data of the array the process has to be repeated until the instruction to stop.

- t_{adc} is the time during which MCU will perform multi-tasks. It will continue to send data from the look up table and also instruct ADC to start data sampling and then automatically convert data to get phase dependent output voltage in-terms of integer from both channels. After completion of conversion of data by ADC, then MCU will stop sending data to DAC.

- t_{pro} is duration of processing of data where MCU will process the binary data and compute numeric phase value according to calibration curve.

- t_{com} is the duration for conversion of data to send it for communication to external world. Here also it is doing multi-task.

So, after t_{cs} MCU will start sending data to DAC, then after t_s it will instruct ADC to start sampling the data from PDC, after t_{adc} it will get some voltage signal and start processing it, after t_{pro} MCU is ready to send data for display or storage and after t_{com} it will start afresh with new value of frequency.

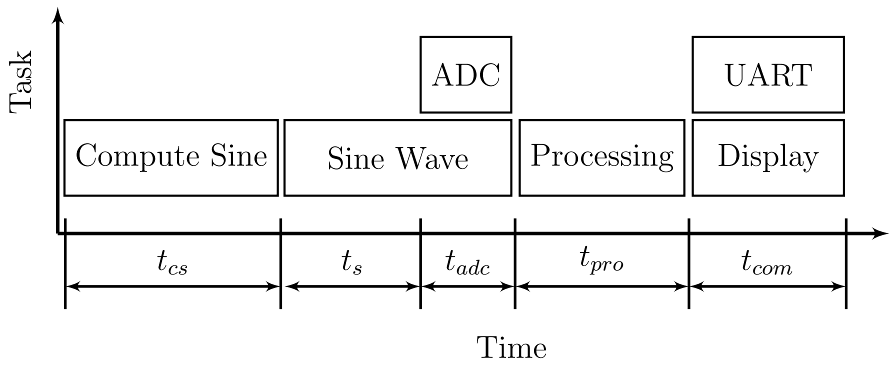


Figure 4.14: Process time line

4.4.5.1 Setting time t_s

MCU will send data of look up table to DAC in cyclic order from its initial value to final value and it will wait t_s s and then send instruction to ADC to start sampling of data. This is required because the DAC will take some time to settle, generate analog sine wave and send that to PDC. The output of PDC will take some time to reach a steady value which depends on the time constant of the circuit.The conversion of data of ADC is set in AUTO mode. The steps to be followed are discussed in Section 4.4.1 only it is required to set the value of PR2 which depends on t_s. t_s is set as 20ms. The value is chosen after statistically testing the time constant of the low pass filter in the PDC block. To get 20ms time gap between the start of generation of sine wave and start of sampling of data by ADC the PR2 register has to be set accordingly. PRESCALER bits is set to binary 11 (i.e., decimal 3) for 1/64 times of clock pulse. Therefore,

$$PR2 = \frac{F_{cy} \cdot T_{int}}{Prescaler} - 1 = \frac{29.4912 \times 10^6 \times 20 \times 10^{-3}}{64} - 1 = 9215 = 23FF$$

4.4.6 Data Processing

The output of 12 bit ADC is in integer form and maximum value that can be attained is $2^{12} - 1 = 4095$. The measured value is the phase. So, instead of converting integer data to voltage and then from voltage to phase it is better to convert directly from integer value to phase by the calibration equation. The calibration equation has been used by calibrating the instrument which is discussed in detail in Section 4.6. The equation used for negative phase is

$$phase = 0.0936 \times phaseN - 163.31$$

and for positive phase it is

$$phase = 0.1048 \times phaseN + 186.78$$

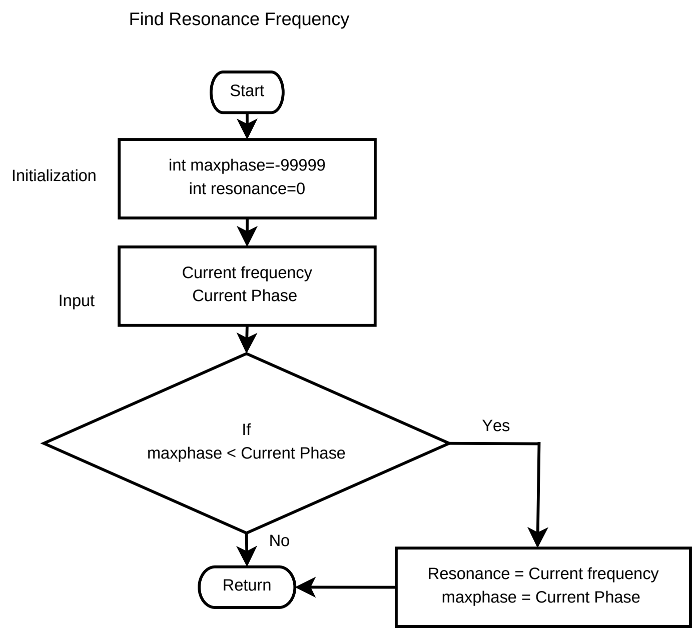


Figure 4.15: Flowchart for finding Resonance Frequency

Final output is multiplied with 100 to avoid floating point data and is stored into an integer variable called phase type.

Resonance frequency of the total range 1kHz to 20kHz is also determined by MCU. The resonance frequency and its corresponding phase value are also stored in a register of MCU. The flowchart is shown in Fig.4.15.

4.4.7 Display

To display the processed data graphically some instruction has to be followed that too are controlled by MCU.

4.4.7.1 Write to GLCD

To write to GLCD the function must have input argument for x and y data as well as the information of brightness of the spot. The function will first call the x and y select function it will select a page by using y/8. Then according to spot bit it will place 1 or 0 at y%8 to write only one bit of the data bus. The steps to be followed are listed below.

- Select x
- Select y

- Write only y modulus of 8 bit through data bus
- Make RS one and RW zero to instruct GLCD for data write mode
- Call enable function

4.4.7.2 Select x and y

The steps that are followed for selection of x and y are given below.

X-axis selection

- Select chip according to the left or right side is required to print.
- Select x axis among 64 column of dots.

Y-axis selection

- Select pages to write among the 8 pages.
- Each page containing 8 dots, can be written directly from data bus

4.4.7.3 Write one column

To write in one column first work is to clear that column. The function will first select x axis and then sequentially select pages and writing data as 0. The steps for performing the function is given below.

- Select x
- Select y page one after one upto 8 pages
- Write zero through data bus
- Make RS one and RW zero to instruct GLCD for data write mode
- Call enable function

4.4.7.4 Display function

The flowchart to display phase response graphically is shown in the Fig.4.12. Input argument for the function is frequency and phase. It is required to scale into GLCD resolution 128×64. Frequency will be scaled into 128 as frequency will be plotted along x axis. Equation for the scaling X-data will be

$$Xdata = \frac{(Currentfrequency - Minfrequency) \cdot 128}{Maxfrequency - Minfrequency}$$

On the otherhand phase will be plotted along y-axis and and equation of scaling of Y data will be

$$Ydata = \frac{(Currentphase - Minfrequency) \cdot 64}{Maxphase - Minphase}$$

So, MCU will calculate the scaled values and call GLCD data write function.

4.5 Data Storing

Data may be stored in a PC using UART as mentioned above. The process requires two function first is to start transmission and next to receive and store data into the register. To do so it is required to put data one by one from UartDATA array to 8bit register (U2TXREG). A pointer is assigned to track current character that is to be transmitted. UART transmitter has four U2TXREG from which data will be sequentially loaded to 'transmit shift register'. It will send the data serially and will generate Tx interrupt after getting 'register empty' signal. Then next four character will be written in U2TXREG. The loop will terminate immediately if it gets any null character '\0' and will start afresh. Flow chart is given in the Fig.4.16.

UART port is connected to ICL232 chip for voltage level enhancement only. The chip contain two module of UART to RS232 converter. Among them Module 1 is selected for the purpose. The output of ICL232 is connected to DB9 connector which in turn connected to serial port of PC. Reception of data to PC is controlled by PC only. As soon as the MCU device receive '#' from UART RX port it will go to RESET mode, start running the loop afresh and then start sending the data to PC. It will continue in an infinite loop till it will receive next '#' from PC.

4.5.1 GUI data interface for PC/Laptop

The received data can be displayed in graphical or text format in PC using MATLAB GUI software. The software has two provisions one is User Interface and other to perform main task. User Interface will collect user option for the task and on the basis of these instructions main program will collect data and store into specified location.

4.5.1.1 GUI Design

Matlab GUI provides different option as User Input and every input button will have a call back function. Call back function will execute the task of that particular input selected.

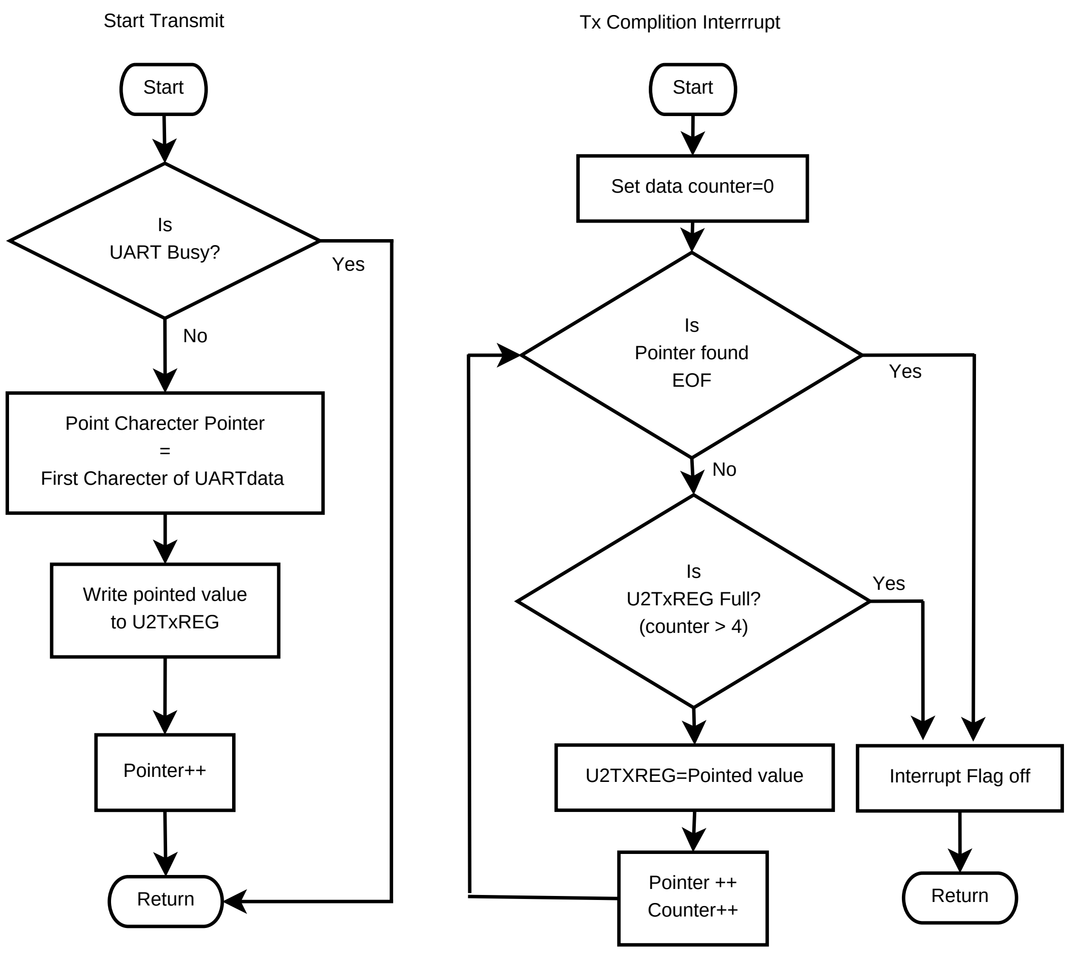


Figure 4.16: Flow chart to Transmit data through UART

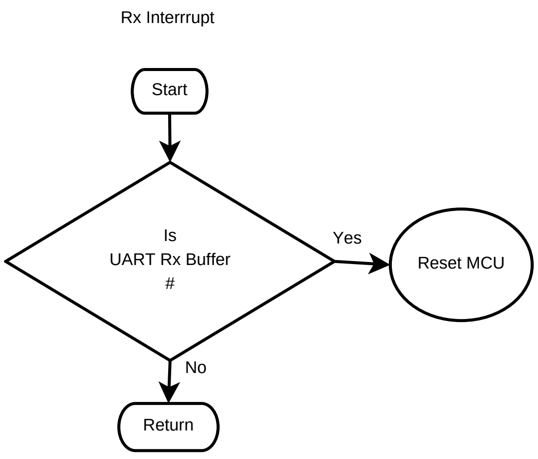
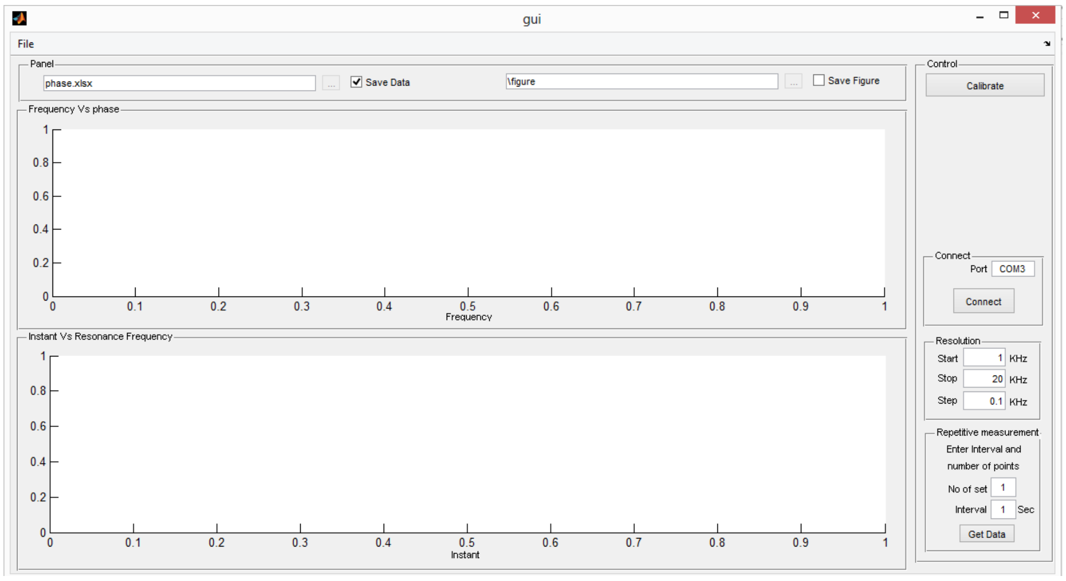


Figure 4.17: Flow chart to Transmit



phase.xlsx

[✓] Save Data

\figure

[ ] Save Figure

Port COM3

Start 1 KHz

Stop 20 KHz

Step 0.1 KHz

No of set 1

Interval 1 Sec

Figure 4.18: GUI Design

For this instrument it is designed to show two plots simultaneously. One is used to show frequency vs phase graph named Plot1. Another graph will show resonance frequency vs instant named Plot2. Screen shot of the GUI software is shown in the Fig.4.18.

•At the top of the panel there is text editor to put the path and the file name to assign the location to save data. The callback function will be executed to perform the task of selecting this option.

•At the right bottom of the panel shows two edit text box. One is to take user choice on number of data set and the other interval between each data set. MATLAB will execute its call back function.

•At the middle right of the panel there is connect button. Connect button call back function will create and open serial port and create a object ‘s’ to control the port. Button text will automatically toggle to disconnect if ‘s’ create successfully. If disconnect button is pressed then it will delete and close the object ‘s’ to disconnect the serial port as per MATLAB norms.

•At top right a × (close) button is provided. A call back function for close will be executed when this is pressed. This function will close and delete the ‘s’ object.

•At bottom right there is Get Data button. This button’s call back function will call the main get data function and data will be collected from the serial port.

Call back function will be executed for a particular option whenever there is any change in any option. Axes are controlled from the main get data function. Before plotting data ‘Plot’ is to be selected as there is more than one Plot.

4.5.1.2 Get Data option

For GUI software the main function is the Get Data option and need some special programming. The flowchart is given in the figure 4.19.

4.6 Calibration

The display of the measurement should be phase in degrees though it has been done in terms of voltage that too in integer. So, a calibration equation is needed to convert voltage to phase reading. And also representation of both positive and negative phase has to be introduced. As mentioned in Chapter 3 that the output of phase measurement circuit has two channels. At the time when phase is positive there will be a non zero voltage output in one channel while the voltage at other channel is zero. And during negative phase opposite output will be obtained. So, calibration of the channels should be done separately.

4.6.1 Repeatability and Bias of the instrument

The impedance phase of a resistance of magnitude $3k\Omega$ has been recorded 5 times in terms of integer to check the repeatability of the instrument. It is well known that the phase of a resistance is zero and constant throughout the range.The result is plotted in the Fig.4.20. From the figure it may be said that it is repeatable but it shows a non-linear curve. It signifies that the system has different offset at different frequencies.

In the next step to recheck the response of the instrument first the phase response of $10nF$ capacitor is recorded using a standard Agilent Impedance analyser, as shown in Fig.4.21a. The reading of same capacitor is recorded 10 times in terms of integer using the instrument, shown in Fig.4.21b . The result if compared shows that the recording of phase with the instrument is repeatable and also the nature of change with respect to Impedance analyser reading is quite similar. If resolution is considered then In analyser the change is about 0.5 degree while the instrument shows about 150 integer values throughout the operating range.

Finally, the phase response also recorded using a piezoelectric sensor again the result shows that the readings are repeatable but have some error in the lower frequency as shown in the Fig.4.22.

From the three test it may be said that when the phase of an element is close to $0°$ the instrument shows a positive bias in the low frequency region and when it is close to $-90°$ it has a negative bias in the same frequency range.

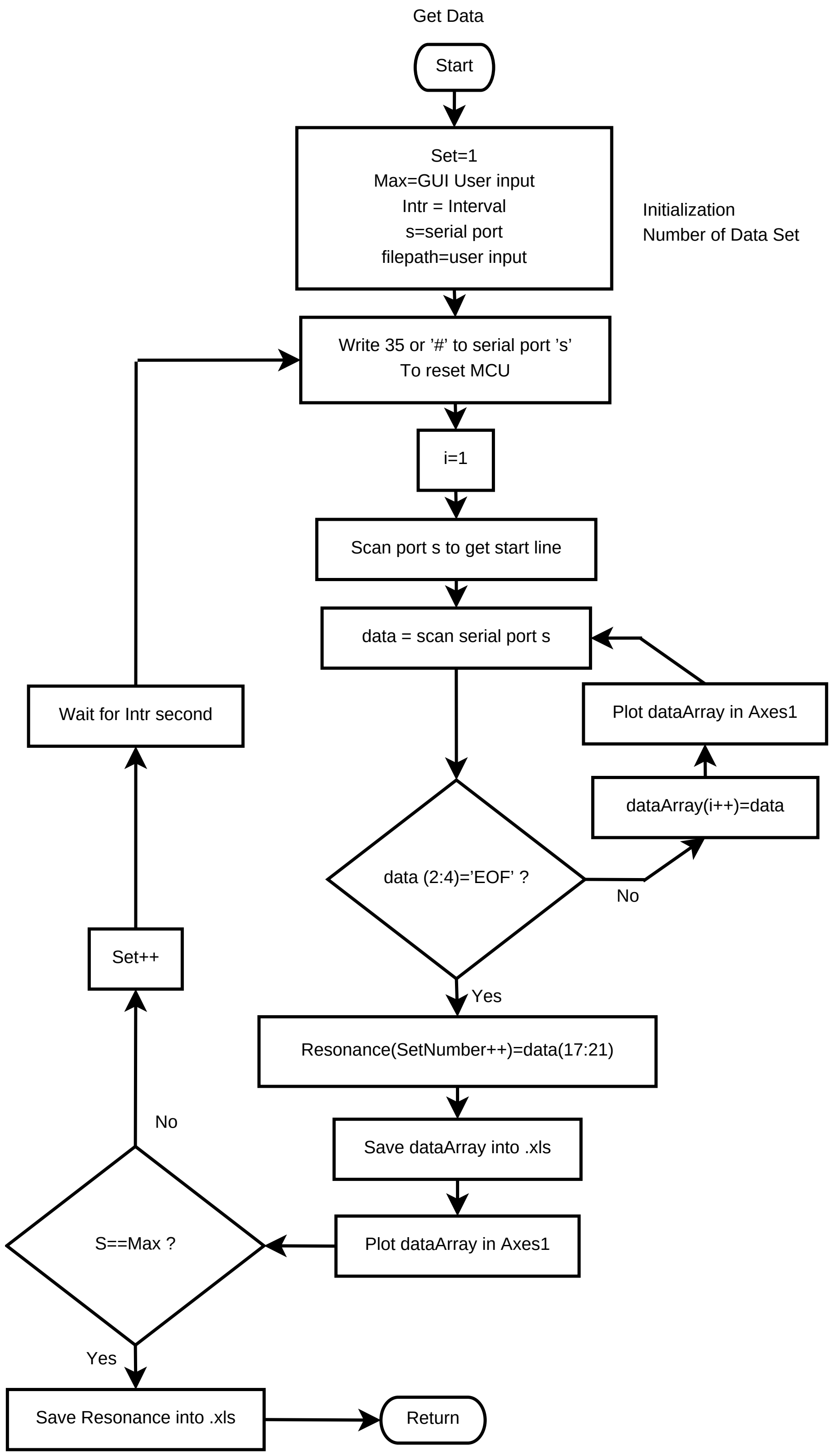


Figure 4.19: GUI Algorithm

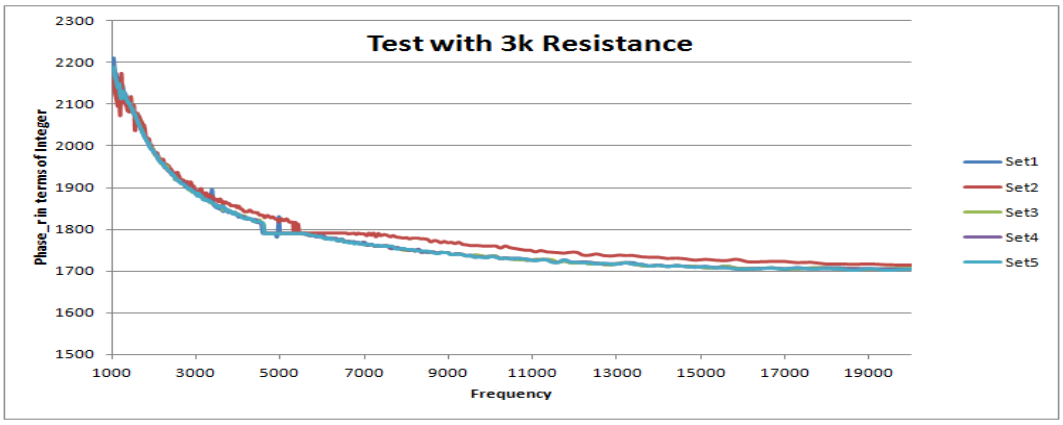
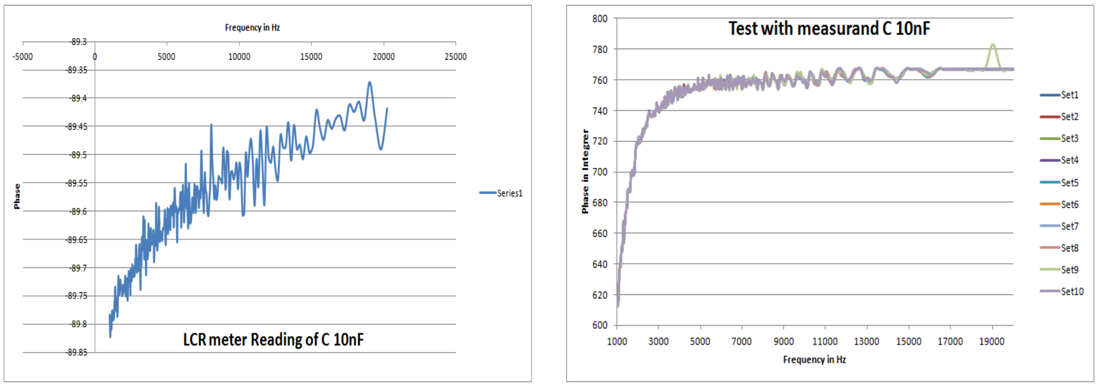


Figure 4.20: Test with resistance



(a) LCR reading 10nF capacitor (b) Test with capacitor

Figure 4.21: Phase response of 10nF capacitor by LCR meter and the instrument

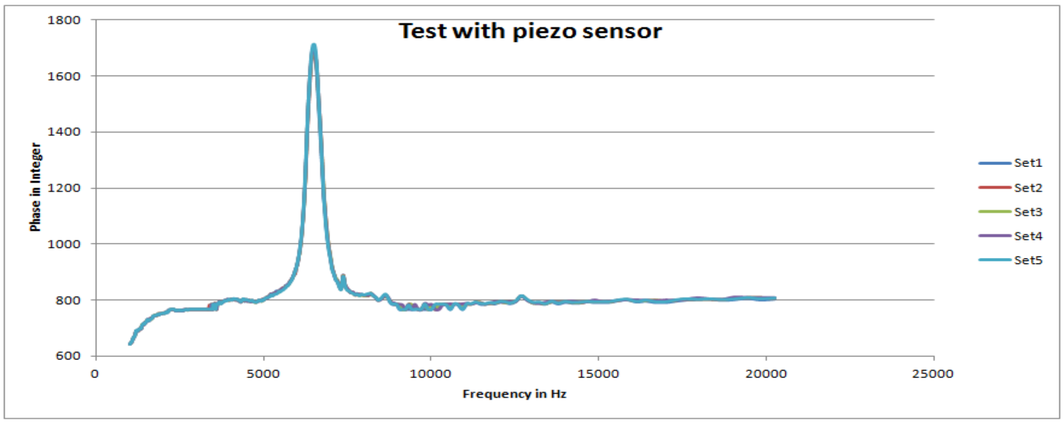


Figure 4.22: Test with piezo sensor

4.6.1.1 Removal of bias

To remove this bias from negative half only capacitor is used as the capacitor has -90^o constant phase response. So, ideally the output of ADC should be a fixed integer value. In practical case the reading is not exactly constant but has some fluctuations. Again the Impedance analyser reading is in degrees whereas the output from ADC is an integer so comparison of the two is not possible. To do so, 10 set of readings of 10nF capacitor are recorded with the instrument. The average value from these 10 sets are determined and assumed that it is the 'true value' of phase of 10nF capacitor in terms of integer.

The assumed 'true value', the ensemble mean of the 10 set of recorded data and the deviation of the two are plotted and shown in the Fig.4.23. The bias is

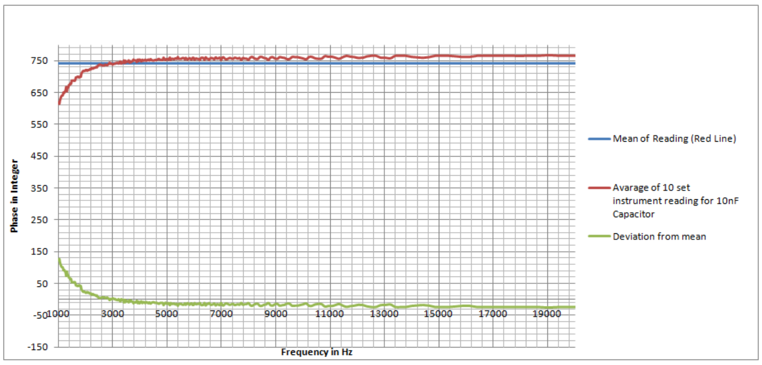


Figure 4.23: Pre Calibration

then removed crudely by subtracting the error from the actual reading. So, the error with respect to frequency is stored in a look up table and is subtracted from each and every phase reading with respect to frequency received from PDC. To check the effect of removal of bias the look up table of error thus generated is used for the instrument reading of three piezo sensors. The result is plotted in Fig.4.24. Comparing the plots visually with the phase response of same sensors obtained from Impedance analyser shows that the method works as the output has a flat response in low frequency range.

4.6.2 Negative phase Calibration

To calibrate the negative phase a RC series combination is used. If the value of R and C is so chosen that its phase response is almost linear in the operating frequency range then it can be used for calibrating the negative phase of the instrument. The value of R and C is $3k\Omega$ and $10nF$. The curve fit equations are shown in the Fig.4.24. The linear fit equation is chosen which has a goodness of

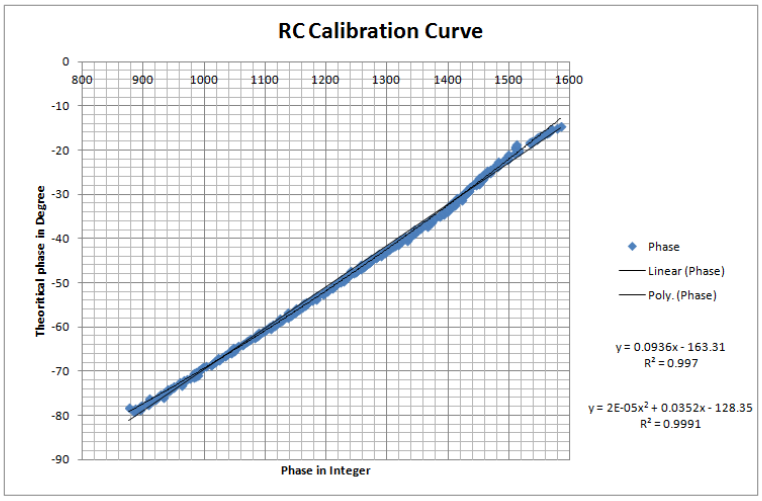


Figure 4.24: Negative phase calibration

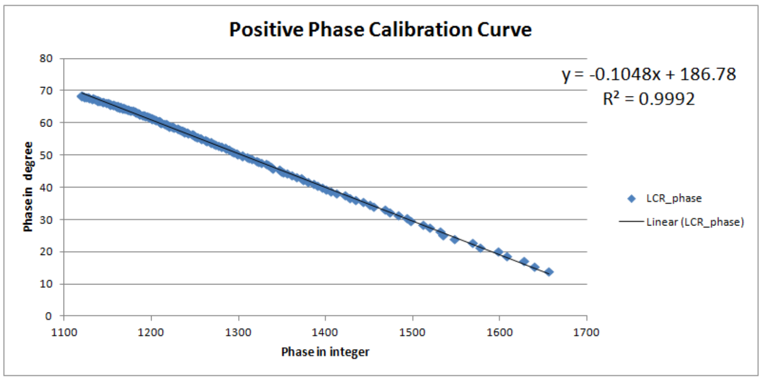


Figure 4.25: LCR meter reading and PDC reading for LR series circuit

fit 0.997.

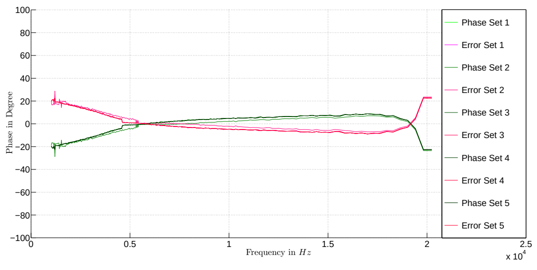
$$phase = 0.0936x - 163.31$$

4.6.3 Positive Phase Calibration

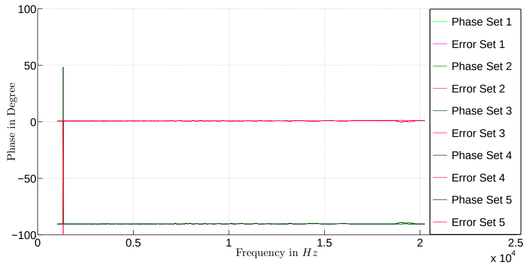
A RL circuit is used to calibrate positive phase. A choke coil of tube light is used as L connected in series with a resistance.

The curve fit equations are shown in the Fig.4.25. The linear fit equation is chosen which has a goodness of fit 0.999.

$$phase = -0.1048x + 186.78$$



(a) Resistance $3k\Omega$



(b) Capacitor $10nF$

Figure 4.26: Standardize Capacitor Resistance phase plot in bluish line and its error in red line

4.7 Testing of the instrument with different devices

The instrument is tested using R, C, RC series, RL series and a piezo based sensor.

4.7.1 Test with R

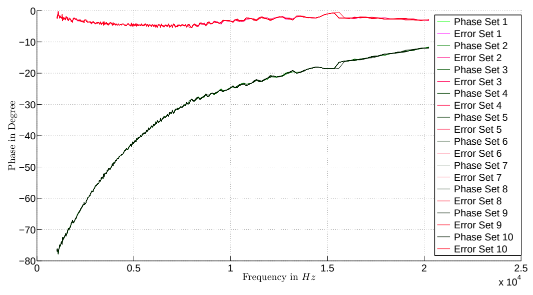
A test have been done with a resistance $3k\Omega$. At first the resistance is tested with LCR meter. Which gives a repeatable output. After that the resistance is marked as standardized resistance. 5 set of data have been taken from the instrument. Result of 5 set is plotted with green line shown in figure 4.26a. Result is subtracted from LCR meter reading and error is plotted in the figure 4.26a with red line. Statistics of error or the red line is given in the table 4.1. Which shows the instrument has mean error of 4.785^o and 1.3% of the total range(360^o). Standard deviation is 8.77^o 2.44% of the total range. Hence, The instrument will not read accurately if it has zero phase, it will read from 13.47^o to -4.07^o. As the instrument will measure resonance frequency the it will not have a problem until it has zero phase.

Table 4.1: Phase error plot and phase $3k\Omega$

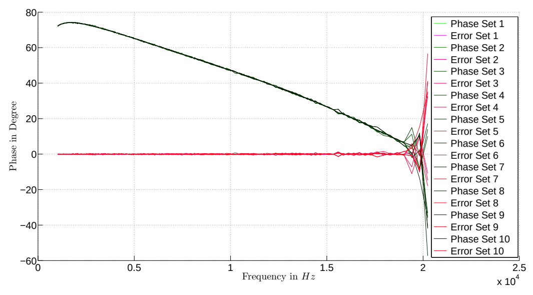
R	Set 1	Set 2	Set 3	Set 4	Set 5	Mean
Mean of Error in Degree	4.471	6.011	4.497	4.477	4.467	4.785
SD of Error in Degree	8.921	8.203	8.913	8.911	8.902	8.77

Table 4.2: Phase error statistics of 10nF capacitor

C	Set 1	Set 2	Set 3	Set 4	Set 5	Mean
Mean of Error in Degree	0.768	0.786	0.766	0.767	0.763	0.77
SD of Error in Degree	0.177	0.169	0.182	0.185	0.185	0.18



(a) RC: $3k\Omega$ and $10nF$



(b) RL:Choke Coil and $3k\Omega$ series

Figure 4.27: Standardize RL and RC series phase response in green line and its error plot red line

4.7.2 Test with C

A test have been done with a capacitor $10nF$. As the previous process for resistance, the capacitor also standardized. 5 set of data have been taken from the instrument. Result of 5 set is plotted with bluish green line shown in figure 4.26b. Result is subtracted from LCR meter reading and error is plotted in the figure 4.26b with red line. The figure also shows a disturbance at 1kHz.

Statistics of error or the red line is given in the table 4.1. Which shows the instrument has mean error of 0.77^o and 0.21% of the total range(360^o). Standard deviation is 0.18^o 0.05% of the total range. Means for -90^o phase it will have -90.59^o to -90.95^o. Which is quite good.

4.7.3 Test with RC

A test have been done with a RC series circuit. R is used $3k\Omega$ and capacitor $10nF$. As the previous process RC series is also standardized by LCR meter. 10 set of data have been taken from the instrument. Result of 10 set is plotted with green line shown in figure 4.27a. Result is subtracted from LCR meter reading and error is plotted in the figure 4.27a with red line. The figure also shows a disturbance at 1kHz. Statistics of error or the red line is given in the table 4.3. Which shows the instrument has mean error of -3.693^o and 1.03% of the total range(360^o). Standard deviation is 1.028^o or 0.3% of the total range. This error can be called as error of the instrument in negative half.

4.7.4 Test with RL

A test have been done with a RL series circuit. R is used $3k\Omega$ and a choke coil of fluorescent tube lamp. like the previous step, RL series is also standardized by

Table 4.3: Phase error plot and phase for 10nF & $3k\Omega$ Series

RC	Set 1	Set 3	Set 5	Set 9	Set 10	Mean
Mean of Error in Degree	-3.705	-3.702	-3.69	-3.695	-3.699	-3.693
SD of Error in Degree	1.032	1.022	1.015	1.033	1.025	1.028

Table 4.4: Phase error plot for choke coil with $1k\Omega$ in series

LR	Set 1	Set 3	Set 5	Set 9	Set 10	Mean
Mean of Error in Degree	0.066	-0.097	0.161	0.12	0.217	0.087
SD of Error in Degree	2.208	1.284	3.46	2.529	3.275	2.168

LCR meter. 10 set of data have been taken from the instrument. Result of 10 set is plotted with green line shown in figure 4.27b. Result is subtracted from LCR meter reading and error is plotted in the figure 4.27b with red line. The figure also shows a disturbance at 1kHz.

Statistics of error or the red line is given in the table 4.4. Which shows the instrument has mean error of 0.087^o and 0.024% of the total range(360^o). Standard deviation is 2.168^o 0.6% of the total range.

4.7.5 Impedance phase of Piezo

The device is tested with standardize piezo. The piezo are highly sensitive with environ mental hazards. To reduce environmental hazrds pizeo is sealed in a plastic jar and tested several times with Agilent LCR meter 4294A. After assuring that the piezo has fixed peak position, the piezo is used for the testing purpose. Result from the instrument verified with the LCR meter reading. Error and phase of six data set from each pizo are plotted. Total of three piezo are given in the figure 4.28. Which also shows the repeatability of the instrument. From these three figure it can be conclude that the maximum error is occurred when $\frac{\delta\phi}{\delta f}$ is very high. In that position small ammount of frequency error results large changes in phase. But the peak position is almost error free as $\frac{\delta\phi}{\delta f}$ is zero at peak. So, The error will not affect on resonance frequency measurement.

Mean and SD of error of six data set of three piezo is shown in the table 4.5.

4.7.6 Impedance phase of Piezo Electric sensor

A test have been done with a piezoelectric sensor named S1. S1 also sealed within a plastic container. As the previous process, S1 is also standardized by LCR meter. 5 set of data have been taken from the instrument. Result of 10 set is plotted with green line shown in figure 4.29. Result is subtracted from LCR meter reading and

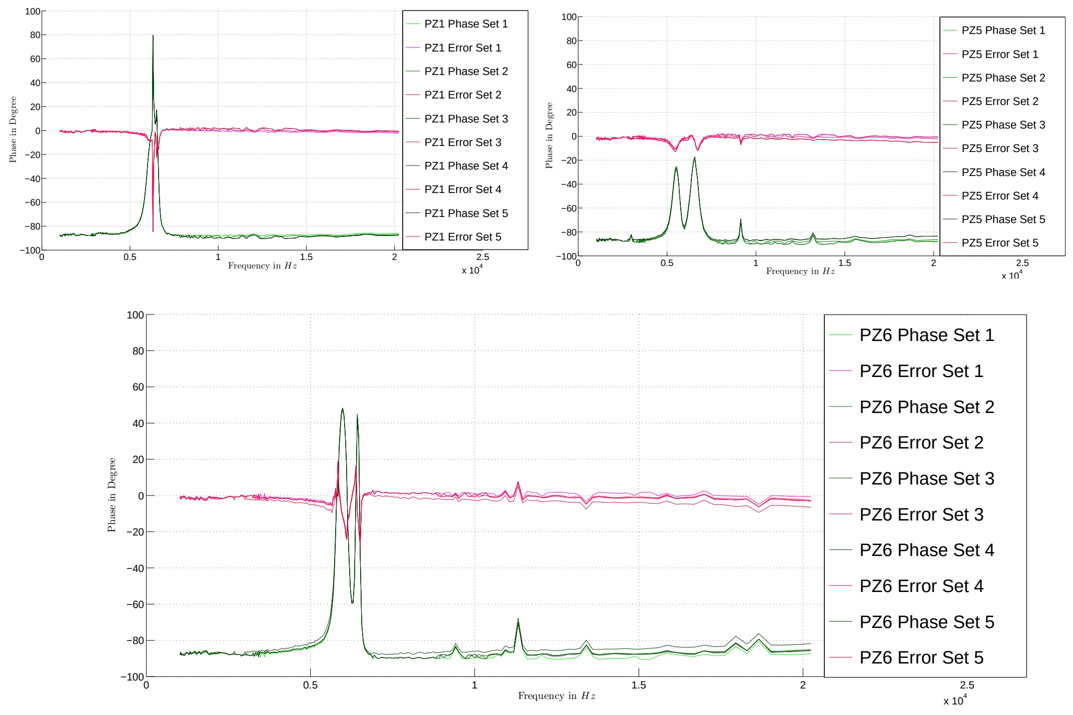


Figure 4.28: Standarize Piezo PZ1, PZ5, PZ6 plot and its error plot

Table 4.5: Error with piezo

Error of PZ1	Set1	Set2	Set3	Set4	Set5	Mean
Mean error	-1.973	-1.777	-1.276	-1.278	-1.283	-1.5174
Standard Deviation of error	5.724	5.795	5.9	5.9	5.898	5.8434
Error of PZ5	Set1	Set2	Set3	Set4	Set5	Mean
Mean error	-1.599	-1.605	-1.52	-1.493	-1.485	-1.5404
Standard Deviation of error	2.549	2.447	2.615	2.458	2.458	2.5054
Error of PZ6	Set1	Set2	Set3	Set4	Set5	Mean
Mean error	-0.818	-1.423	-1.387	-2.897	-1.364	-1.5778
Standard Deviation of error	3.666	3.684	3.75	3.789	3.861	3.75

error is plotted in the figure 4.29 with red line. The figure also shows a disturbance at 1kHz.

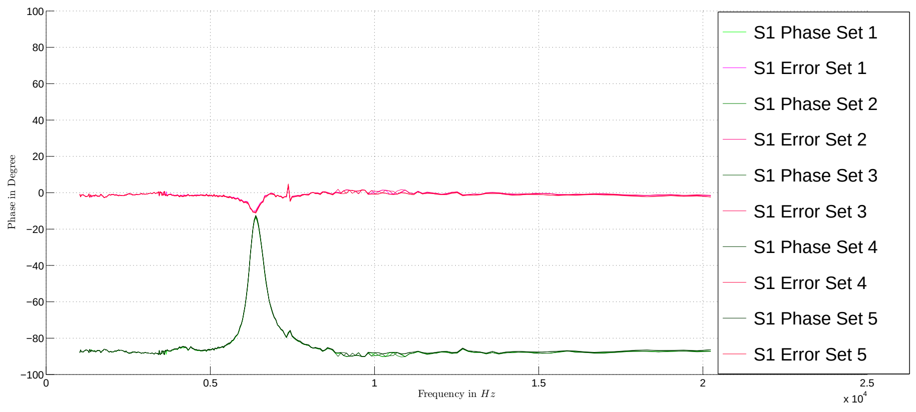


Figure 4.29: Standardized sensor S1

Table 4.6: Mean and SD of error of S1 sensor

Error of S1	Set1	Set2	Set3	Set4	Set5	Mean
Mean error	-1.377	-1.407	-1.363	-1.672	-1.615	-1.4868
Standard Deviation of error	1.725	1.818	1.831	1.789	1.815	1.7956

Statistics of error or the red line is given in the table 4.6. Which shows the instrument has mean error of -1.4868^o and 0.5% of the total range(360^o). Standard deviation is 1.7956^o 0.5% of the total range. Which is quite acceptable.

4.7.7 Conclusion

It can be conclude from above test that the instrument has an error $\pm 8.77^o$ at zero phase position. Though the other phase value has relatively low error. $\pm 1.028^o$ for negative phase and $\pm 2.168^o$ for positive phase.

4.8 Application of the instrument

The instrument can be used to measure impedance phase response. It has the facility to display it graphically. It is also capable to determine the resonance frequency from the phase response and display it. It can store the data in a PC. It is able to record the shift of resonance frequency with time.

4.8.1 Recording of Shift of Resonance frequency with time

One of the application of the instrument is to check the stability of dental implant. A shift in the resonance frequency indicates the stability of the dental implant if the sensor is attached to the dental implant placed in the jaw bone. The resonance frequency will reach a steady value when there is a good bonding of bone tissue implant interface. To use this instrument for this application first in-vitro test has been done.

4.8.1.1 In-vitro test

Phase response of the sensor designed for measuring dental implant stability has been tested in-vitro by dipping the implant into dental plaster. A specified amount of dental plaster is mixed with a specified amount of water and poured into a specified container. A dental implant is dipped into it upto a specified height of the implant. The sensor is attached with the dental implant by means of a screw. The dental plaster will set with time and gives a measure of bonding between plaster and implant. This experiment has been performed with the assumption that dental plaster from pouring into the container to setting is mimicking the process of bonding of bone-tissue implant interface. The instrument will record the impedance phase response of the sensor for 1kHz to 20kHz frequency range. Again each loop of phase response will be recorded with a time gap of 1min and for total time of about 100min. So, from every phase response it is possible to find a global maximum which is the resonance frequency at that instant of recording. For this case 100 such points will be obtained. If this resonance frequency is plotted with respect to time then the plot will show a steady value when implant is well bonded to the plaster. The plot of resonance frequency is shown Fig.4.30. The figure clearly shows that as the plaster become harder resonance frequency shifts from 5kHz and reaches almost staedy value after 60th instant.

4.8.1.2 In-vivo test

An In-vivo test has been done in Dr.T.K.Pal's Dental Clinic & Implant Centre. Three implants are placed on the upper jaw of a patient. From clinical observation it has been declared that the implants are almost stable. Instrument and recording setup (Laptop) is connected with proper isolation. Photograph is shown in Fig.4.31. The readings are taken under two condition: one with sensor attached to the implant when screw is normally tight and others with extra tight. The phase responses of three implants are shown in Figs. 4.32a, 4.32b and 4.32c.

From the results as shown in the figures it seems that in the operating frequency range there are two comparable maxima. The normally tight condition shows more repeatable values than extra tight conditions. Phase values are close within -65°

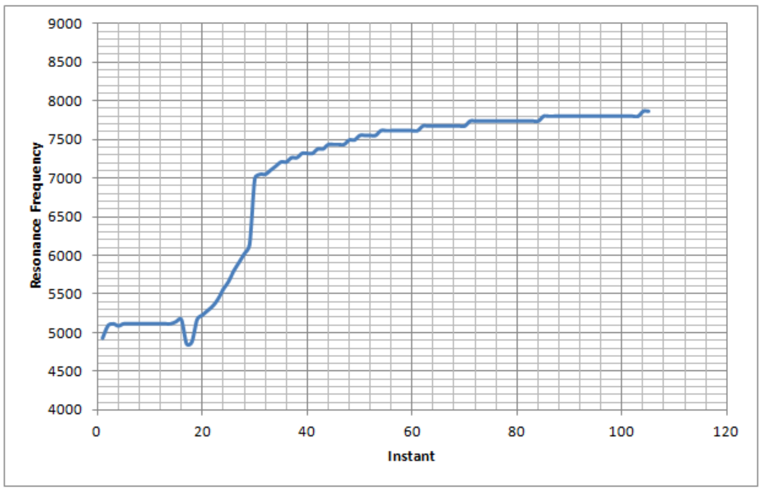


Figure 4.30: Measure of dental implant stability in-vitro by the instrument

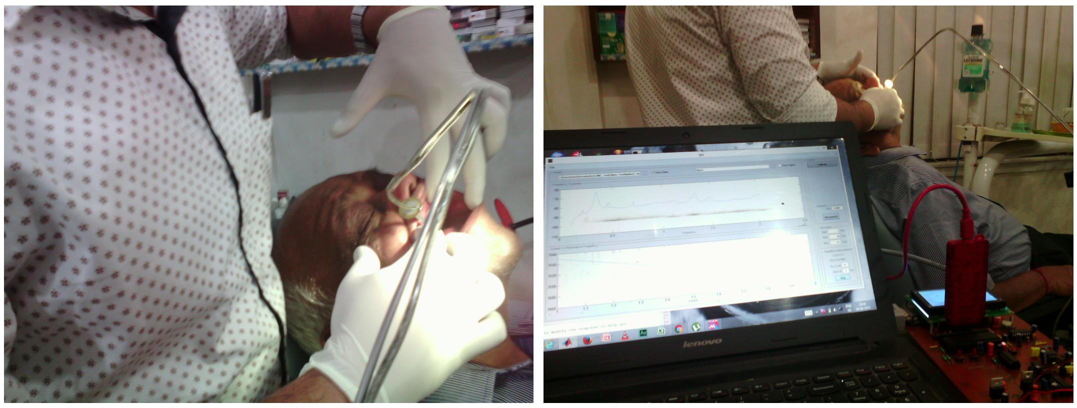
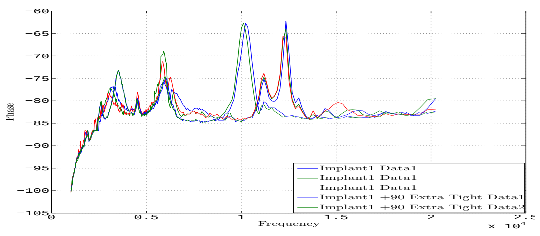


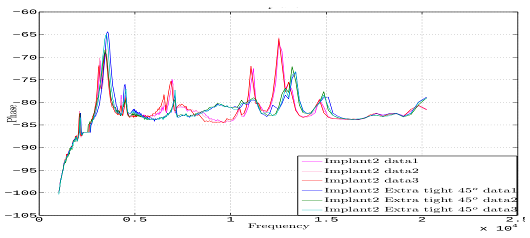
Figure 4.31: Photograph of In-vivo test

to -70°. The corresponding frequencies are near about 12kHz in higher range and 4kHz in lower range.

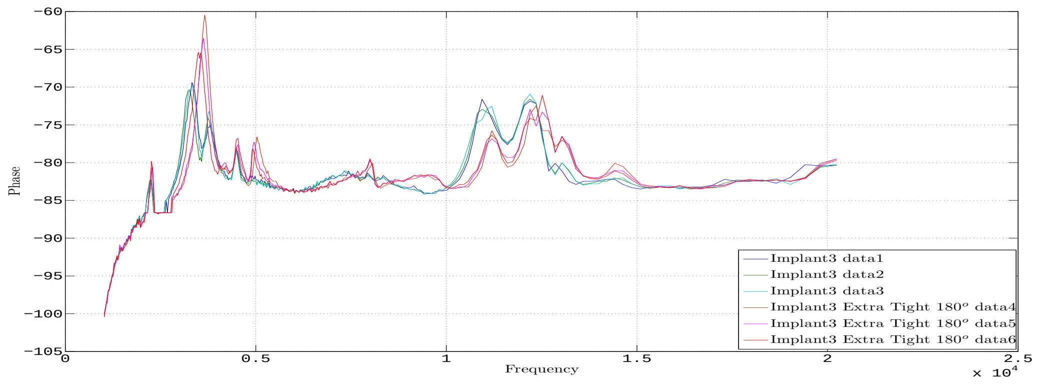
So, the readings from the instrument may be used for in-vivo tests. In this case the readings are taken after implant reaches a stable condition. To record the shift of resonance frequency the readings should be taken from start that is when the implant is just placed in the jaw. A detailed experimental observation can indicate the performance of the instrument.



(a) In-vivo reading with Implant 1



(b) In-vivo reading with Implant 2



(c) In-vivo reading with Implant 3

Figure 4.32: In-vivo test

4.9 Specification of the instrument

	Condition	MIN	TYP	MIN	Unit
Absolute impedance of measurand	Rf=3k	0.3	3	30	$k\Omega$
Phase impedance of measurand		-180		180	Degree
Phase with noise		-90		90	Degree
±ve power supply		7.5	9	15	Volt
Number of supply			2		Unit
+ve Supply Current		270	250	220	mA
-ve Supply Current		10	10	10	mA
+ve Power Consumption		10	2160	10	mW
-ve Power Consumption		10	90	10	mW
Without LCD +ve Current Consumption		1530	1440	1350	mA
Without LCD -ve Current Consumption		170	160	150	mA

Chapter 5

Conclusion and future scope

5.1 Conclusion

The aim of the work is to design and develop a standalone instrument for checking the dental implant stability. The sensor has already been developed and a compatible input and its corresponding readout device has been developed in this work.

The instrument can be used to measure impedance phase response of an element. It has the facility to display it graphically. It is also capable to determine the resonance frequency from the phase response and display it. It can store the data in a PC. It is able to record and plot the shift of resonance frequency with time.

The novelty of the work is firstly a microcontroller is used to generate a variable frequency sinusoidal signal whereas usually it is used to generate a sinusoidal signal of a particular frequency. The amplitude of the signal is set here at peak to peak 1volt with a bias of 0.5volt. Options are provided to introduce or remove the bias. The other novelty is detecting both positive and negative phase. To do so an innovative way of detection are used by two D flip-flops, one for positive and another for negative phase.

The performance of the instrument has been checked by using in-vitro and in-vivo tests. In this case the in-vivo readings are taken after implant reaches a stable condition. To record the shift of resonance frequency the readings should be taken from start that is when the implant is just placed in the jaw.

5.1.1 Errors in measurement

There are some error in the reading of the instrument.
• The recorded frequency does not exactly matches with the actual frequency generated by the microcontroller because it is obtained from the calculation and

also due to conversion from digital value to decimal value. The error is prominent for high frequency sine wave.
• The simulation as well as the reading shows that the flip-flop introduce large error when the phase is close to zero. It is also unable to track the sharp change of phase. This arises due to the logic used for detecting phases with the flip-flops.
• The instrument has a bias specially in the low frequency region which may be due to the phase bandwidth of the chip used for converting sine to square wave.
• Sometime the instrument starts generating very high frequency noise. Reason is yet not identified but may be due to stray electromagnetic effects. It works normally again if the instrument is reset and started afresh.

5.2 Future scope

The future scope of work for improvement of the instrument are listed below.

5.2.1 Phase detector

In some cases the phase change of piezo sensor, as observed from the record using Agilent Impedance analyser, is from -90° to some positive value close to 90°. As mentioned the performance of the instrument is very poor when phase values are close to zero. The uncertainty of detecting this range of phase is due to the logic used by the D flip-flop. This problem may be solved if the logic of an active high clear positive edge trigger D-Type flip-flop used. A preliminary survey has been done and can be implemented in future. The simulation study is discussed here for future scope of work.

The logic of the active high clear positive edge trigger D-Type flipflop, named as Type A is shown in Fig.5.1. Simulation result of the circuit is shown in Fig.5.2. Differential output voltage curve is shown in Fig.5.3. It has higher sensitivity 0.0125 Volt/^{o}phase and uncertainty state is shifted to $\pm 180^o$. Thus the circuit can detect phases close to zero and zero value correctly. If these design is used in the instrument then it will improve the measurement.

5.2.2 Other modification

Some more study and further modification will improve the performance of the instrument and also add some extra facility.

- A detailed experimental observation is required to study the performance of the instrument. For that purpose more in-vitro and in-vivo tests should be done.

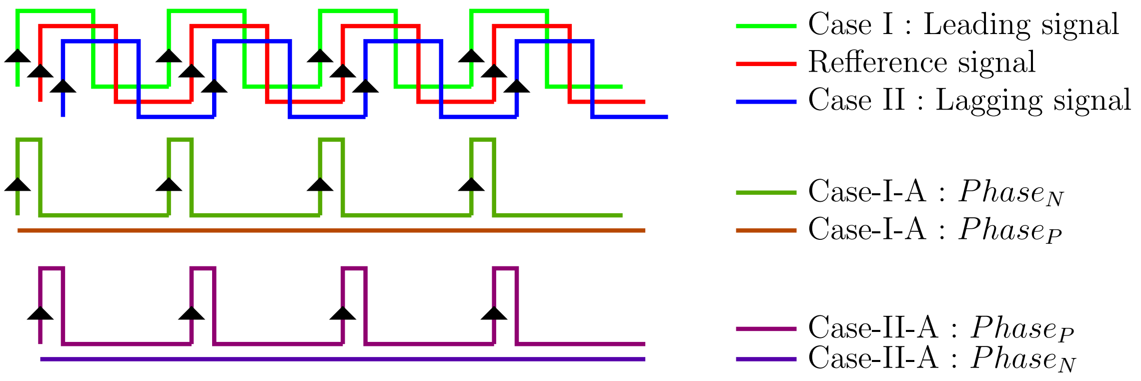


Figure 5.1: Input and output wave form of proposed phase detector circuit

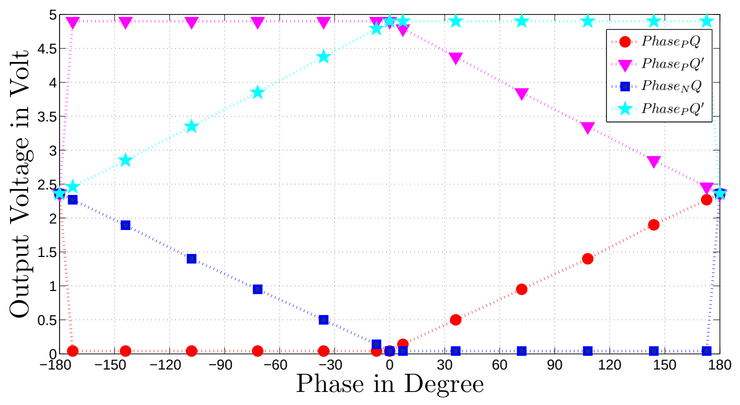


Figure 5.2: Type-A simulated characteristics curve

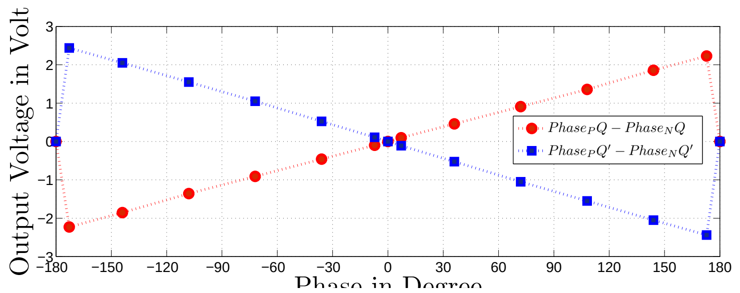


Figure 5.3: Diiferential output of Type-A simulated characteristics curve

- Using +ve to -ve DC generator (e.g. LMC7660) a single battery can be used instead of dual DC battery.
- A battery level sensor and a corresponding alert signal of low battery may help the user from regular checking of the condition of the battery.
- Space for SD card has been kept in the PCB. Development of code and inclusion of the card will help to solve the problem of storing data within the instrument. In that case it will be self sufficient and connection with PC will not be required for storing data.

Appendix A

GLCD specification

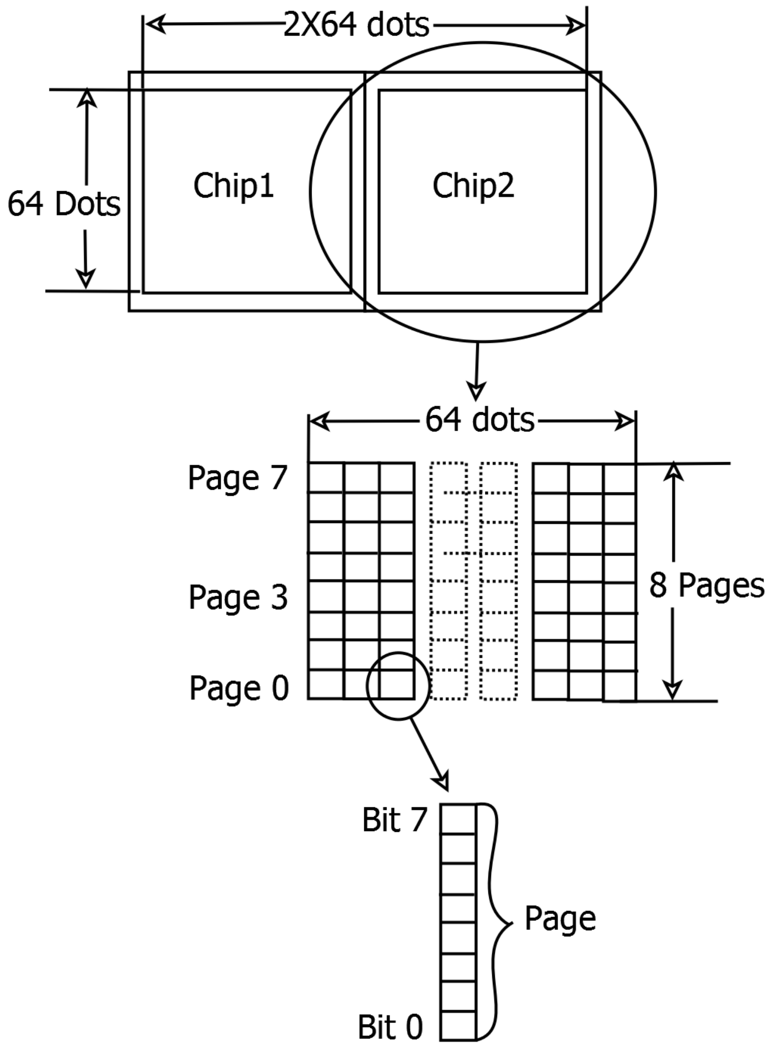
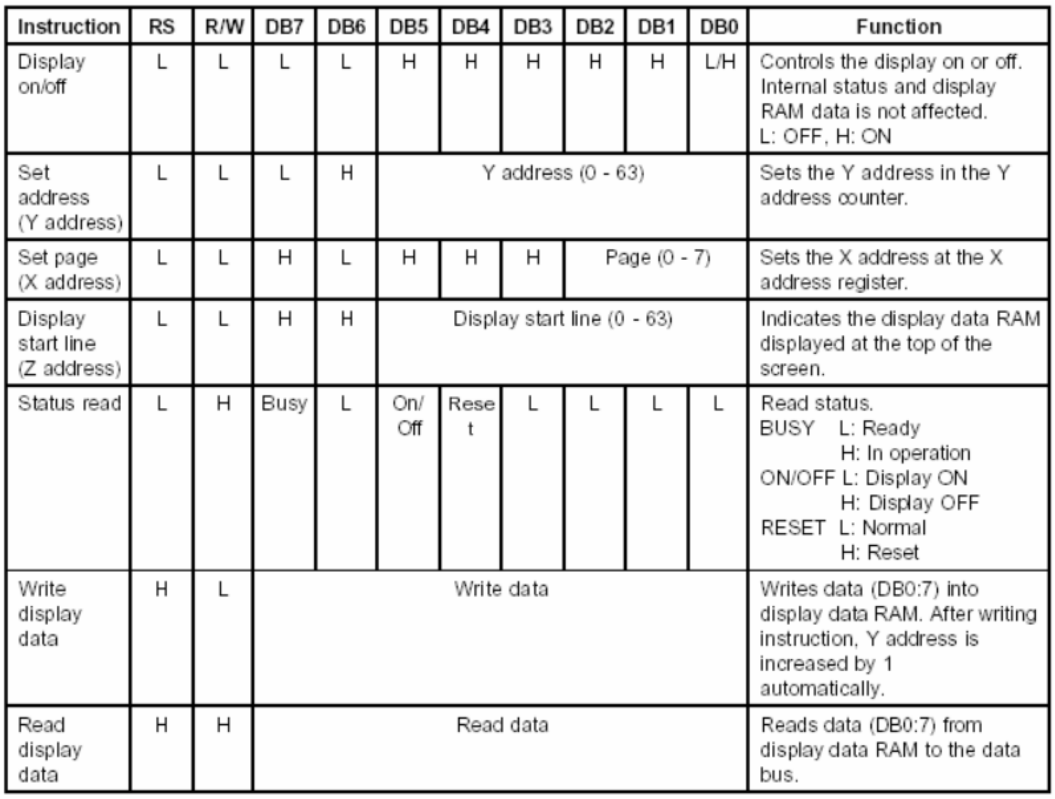


Figure A.1: GLCD dots arrangement



Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display on/off	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is not affected. L: OFF, H: ON
Set address (Y address)	L	L	L	H	Y address (0 - 63)						Sets the Y address in the Y address counter.
Set page (X address)	L	L	H	L	H	H	H	Page (0 - 7)			Sets the X address at the X address register.
Display start line (Z address)	L	L	H	H	Display start line (0 - 63)						Indicates the display data RAM displayed at the top of the screen.
Status read	L	H	Busy	L	On/ Off	Rese t	L	L	L	L	Read status. BUSY L: Ready H: In operation ON/OFF L: Display ON H: Display OFF RESET L: Normal H: Reset
Write display data	H	L	Write data								Writes data (DB0:7) into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read display data	H	H	Read data								Reads data (DB0:7) from display data RAM to the data bus.

Figure A.2: Instruction set for GLCD

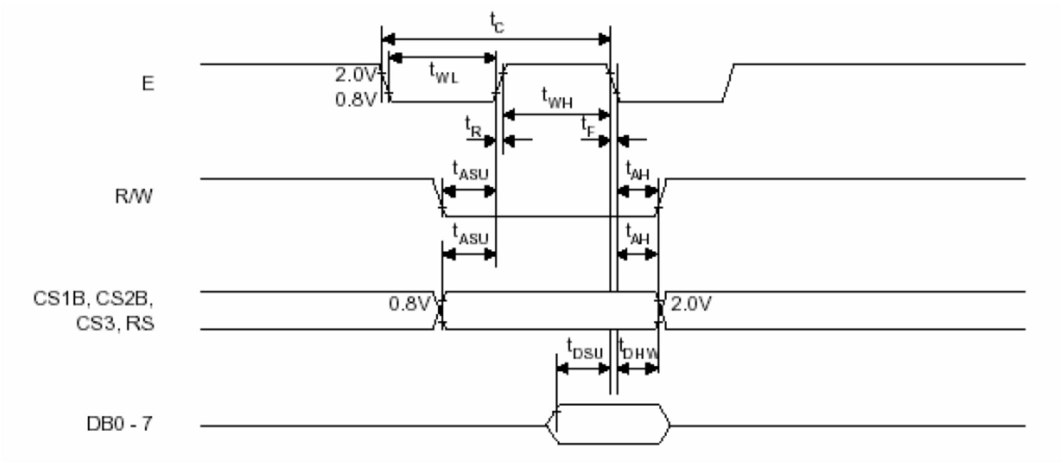


Figure A.3: Timing diagram for GLCD

Appendix B

Microcontroller Code

B.1 main.c

```
#include <p30f4013.h>

_FOSC(CSW_FSCM_OFF & XT_PLL16); //Run this project using an external
    crystal
                             //routed via the PLL in 8x multiplier mode
                             //For the 7.3728 MHz crystal we will
                                derive a
                             //throughput of 7.3728e+6*16/4 = 29
                                MIPS(Fcy)

_FWDT(WDT_OFF);              //Turn off the Watch-Dog Timer.
_FBORPOR(MCLR_EN & PWRT_OFF); //Enable MCLR reset pin and turn off the
                             //power-up timers.
_FGS(CODE_PROT_OFF);         //Disable Code Protection
extern void intsine(void); // Load external function
extern void startsine (void);
extern void stopsine(void);
extern void range(int , int);
extern void SPI_Init(void);
extern void ADC_Init(void);
extern void UART_Init (void);
extern void Timer2_Init(void);
extern unsigned char DATA[];
extern void glcdon(void);

int startFrequency=1000; //Frequency range
int stopFrequency=20000;
int actualStart;
int freq10;

int main()
{
  //              76543210
   TRISB = 0b1111111100000000; // Make port B as output port
   TRISD=0x0000;         // Make port D as output port
   PORTD=0;           // Initialize port D as 0
    PORTB=0;             // Initialize port B as 0
   glcdon();             // Switch on GLCD
   Delay1ms(50);         // Wait 50ms for GLCD
   UART_Init();          // Initialize UART
   startsine();          // Start Sine Calculation and generation

   while(1)              //Infinite loop
    {

    }

    return 0;
}
```

B.2 sine.c

```
#include <p30f4013.h>      //Load Header file for I/O and register
    specification for dsPIC30f4013
#include <math.h>        //Load for sin functon
#include <string.h>        //load to change string
#define PI 3.14159265359   //Value of Pi
int i=0;              //Initialize counter i
int start=0;             //Initialize start point of sine wave
int stop=100;            //Initialize stop for N sample/cycle
//Define Used functions
void __attribute__((__interrupt__)) _T1Interrupt(void);
extern void WriteUART_to_RS232(void);
extern unsigned char UartData[];
unsigned char DATA[1000]; //Initialize lookup table for sine
extern int resonance;    //Load variable resonance from other file
int freqHz,endf; //Initialize current frequency
```

```
unsigned long ClockCalibHz; //Sine calibrated frequency
extern int startFrequency; //Load info from main.c file
extern int stopFrequency;

void stopsine (void) //function for stop generating sine wave
{
   T1CONbits.TON=0;   //Stop sine interval timer
   IEC0bits.T2IE = 0;    //Stop wait time timer interrupt
   T2CONbits.TON=0;   //Off wait time timer
   ADCON1bits.ADON = 0; //Off ADC module
}


void intsine( void)
{
    T1CON = 0x0000;       //Timer1 set up to count on instruction cycle
                              //edge with 1:1 prescaler applied
                                  initially.
    T1CONbits.TSIDL=1;   //disable in IDLE mode
   PR1 = 30;          // Set for frequency 950kHz
   ClockCalibHz=951412;   //Load calibrated interval
        IEC0bits.T1IE = 1;    //Enable Timer1 Interrupt Service Routine
        T1CONbits.TON=0;      //Start Timer 1 (Sine sample Interrupt)
}
void range(int freqS, int freqE ) //Calculate range
{
stop=(unsigned long)ClockCalibHz/freqS+1;
endf=(unsigned long)ClockCalibHz/freqE;
int actualStart=ClockCalibHz/stop;  //Find possible frequency
int actualStop=ClockCalibHz/endf;
changesine(); //Change sine
  return;
}

void changesine(void)
{
   int j;
   float temp;
   T1CONbits.TON=0; //Off generation if its on
   if(stop<endf)//If reach to maximum frequency
      {  stopsine();
              //     01234567890123456789012
         strcpy(UartData, "EEOF_Resonance: ######\0");
        UartData[16] =resonance/10000%10+48;
        UartData[17] =resonance/1000%10+48;
        UartData[18] =resonance/100%10+48;
        UartData[19] =resonance/10%10+48;
        UartData[20] =resonance%10+48;
        UartData[21] =10;
        UartData[22] ='\0';

        WriteUART_to_RS232();
        T2CONbits.TON=0;
        Idle(); // access to IDLE mode
      }
   else
   {

      freqHz=ClockCalibHz/stop;

      float factor=2*PI/((int)stop+1);
      for(j=0;j<=stop;j++)
      {
        DATA[j]=(1.1+sin(factor*j))*115;
      }

      i=0;
   T1CONbits.TON=1;
   T2CONbits.TON=1;
   }
   stop--;
   return;
}
void startsine (void)
{
   intsine();
   ADC_Init();
   Timer2_Init();

   Delay5ms(200);
   range(startFrequency,stopFrequency);
}

//Sine generation interval
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
     PORTB=DATA[i++]; //Send 8bit discrete sine value from port B
       if(i==stop) //If reach to last value
        i=start; //then go to first
   //LATD=~LATD; //To calibrate sample frequency
      IFS0bits.T1IF = 0; //Clear interrupt flag
}
```

B.3 waitforsettlingtime.c

```
#include <p30f4013.h>
#include "system.h"

void Timer2_Init(void);
void __attribute__((__interrupt__)) _T2Interrupt(void);


void Timer2_Init(void)
{
       T2CON = 0x0020;       // pre scaler 1:64
     T2CONbits.TSIDL=1; //disable in IDLE mode
       PR2 =0x1FFF;      //wait time
       IFS0bits.T2IF = 0;    //Clear the Timer2 Interrupt Flag
       IEC0bits.T2IE = 1;    //Enable Timer2 Interrup Service Routine
}
After completion of wait time
void __attribute__((__interrupt__)) _T2Interrupt(void)
{

     //Turn on the A/D converter
       ADCON1bits.ADON = 1;
     ADCON1bits.SAMP=1; //Start ADC sampling
       IFS0bits.T2IF = 0; //Clear Timer1 Interrupt Flag
}
```

B.4 adc.c

```
//Pre-Processor Directives:
#include <p30f4013.h>
```

```
#include "system.h"

//Functions and Variables with Global Scope:
void ADC_Init(void);
void __attribute__((__interrupt__)) _ADCInterrupt(void);
extern void changesine(void);

extern void WriteSPI_to_LCD(void);
extern void WriteUART_to_RS232(void);
extern unsigned char UartData[];
extern void stopsine (void);
extern void changesine(void);
void UpdateDisplayBuffer(void);
int phaseP = 0; //Initialize value storage
int phaseN = 0; //Initialize value storage


void ADC_Init(void)
{
//ADCON1 Register
       //Set up A/D for Automatic Sampling, Auto-Convert
       //All other bits to their default state
       ADCON1bits.SSRC = 7;
       ADCON1bits.ASAM = 0;
       ADCON1bits.ADSIDL=1; //disable in IDLE mode
       //ADCON2 Register
       //Set up A/D for interrupting after 2 samples get filled in the
          buffer
       //Also, enable Channel scanning
       //All other bits to their default state
       ADCON2bits.SMPI = 1;
       ADCON2bits.CSCNA = 1;

       //ADCON3 Register
       //Total Sample Time = Acquisition Time + Conversion Time
       //All other bits to their default state
       ADCON3bits.SAMC = 31;
       ADCON3bits.ADCS = 7;

       //ADCHS Register
       //When Channel scanning is enabled (ADCON2bits.CSCNA=1)
       //AND Alternate mux sampling is disabled (ADCON2bits.ALTS=0)
       //then ADCHS is a "don't care"
```

```
        ADCHS = 0x0000;

        //ADCSSL Register
        //Scan channels AN12, AN11 fas part of scanning sequence
        ADCSSL = 0x1800;

        //ADPCFG Register
        //Set up channels AN2, AN3 as analog inputs and leave rest as
            digital
        //Recall that we configured all A/D pins as digital when code
            execution
        //entered main() out of reset
        ADPCFGbits.PCFG2 = 0;
        ADPCFGbits.PCFG3 = 0;

        //Clear the A/D interrupt flag bit
        IFS0bits.ADIF = 0;

        //Set the A/D interrupt enable bit
        IEC0bits.ADIE = 1;

        //Turn on the A/D converter
        //This is typically done after configuring other registers
        ADCON1bits.ADON = 0;

}

//_ADCInterrupt() is the A/D interrupt service routine (ISR).
//The routine must have global scope in order to be an ISR.
//The ISR name is chosen from the device linker script.
void __attribute__((__interrupt__)) _ADCInterrupt(void)
{
        T1CONbits.TON=0; //Stop Sine wave
      phaseP = ADCBUF0; //Store phase value
      phaseN = ADCBUF1;
      UpdateDisplayBuffer(); //Data Processing
      changesine(); //Call for sine value change
      IFS0bits.ADIF = 0;
}
```

B.5 dataprocessing.c

```
#include <p30f4013.h>
#include "system.h"
extern int phaseP;
extern int phaseN;
int resonance=0;
int maxphase=-18000;
//Declaration to Link External Functions & Variables:
extern int   k;
extern int freqHz;
//Functions and Variables with Global Scope:
void UpdateDisplayBuffer(void);
extern void WriteToGlcdX(int,int);
unsigned char UartData[]="FreqH\tPhaseD\tPhs+\tPhs-\n\0"; %Data Format
unsigned char adones=0;
unsigned char adtenths=0;
unsigned char adhundredths=0;
void UpdateDisplayBuffer(void)
{
      UartData[0] =freqHz/10000%10+48;
      UartData[1] =freqHz/1000%10+48;
      UartData[2]= freqHz/100%10+48;
      UartData[3]= freqHz/10%10+48;
      UartData[4]= freqHz%10+48;
      UartData[5]='\t';
float temp;
//Calibration curve
if(phaseP>phaseN)
      temp=-0.08486*(float)phaseP+151.72;//temp=-0.0857*(float)phaseP+157.17;
else
      temp=0.08486*(float)phaseN-151.72;//temp=0.0896*(float)phaseN-163.12;
int phase=(int)(100*temp);
if(maxphase<phase) //Resonance finder
   {
      maxphase=phase;
      resonance=freqHz;
   }

UartData[6]= '+';
int p=phase;
if (phase<0)
{
```

```
UartData[6]= '-';
p=-1*phase;
}


// ADCResult2Decimal(); //Convert the A/D hex value to
      UartData[7]= p/10000%10+48;
      UartData[8]= p/1000%10+48;
      UartData[9]= p/100%10+48;
      UartData[10]= '.';
      UartData[11]= p/10%10+48;
      UartData[12]= p%10+48;
      UartData[13]='\t';

      UartData[14] = phaseP/1000%10+48;   //decimal and update the
          display buffer
      UartData[15] =phaseP/100%10+48;
      UartData[16] = phaseP/10%10+48;
      UartData[17] = phaseP%10+48;
      UartData[18] = '\t';
      UartData[19] = phaseN/1000%10+48;   //decimal and update the
          display buffer
      UartData[20] =phaseN/100%10+48;
      UartData[21] = phaseN/10%10+48;
      UartData[22] = phaseN%10+48;
      UartData[23]='\n';
WriteUART_to_RS232();
WriteToGlcdX(freqHz,phase);
}
```

B.6 UART.c

```
#include <p30f4013.h>
#include "system.h"

#define BAUDRATE      9600                  //Desired Baud Rate
#define BRGVAL        ((2*FCY/BAUDRATE)/16)-1 //Formula for U2BRG
    register
                                            //from dsPIC30F Family
                                            //Reference Manual
```

```
//Declaration to Link External Functions & Variables:
extern unsigned char UartData[];
extern void startsine (void);
//Functions and Variables with Global Scope:
void UART_Init (void);
void WriteUART_to_RS232(void);
void __attribute__((__interrupt__)) _U2TXInterrupt(void);

unsigned char *UARTCharPtr;
extern int freq10;

//Functions

//UART_Init() sets up the UART for a 8-bit data, No Parity, 1 Stop bit
//at 9600 baud with transmitter interrupts enabled
void UART_Init (void)
{
       U2MODE = 0x0000;      //Clear UART2 registers
   // U2MODEbits.PDSEL=1;   //even parity
       U2STA = 0x0000;
      //  U2MODEbits.ALTIO = 1; //Enable U2ATX and U2ARX instead of
                             //U2TX and U2RX pins

       U2MODEbits.UARTEN = 1; //Enable UART1 module
       U2BRG = BRGVAL;       //Load UART1 Baud Rate Generator

       IFS1bits.U2RXIF = 0;  //Clear UART1 Receiver Interrupt Flag
       IFS1bits.U2TXIF = 0;  //Clear UART1 Transmitter Interrupt Flag
       IEC1bits.U2RXIE = 1;  //Disable UART1 Receiver ISR
       IEC1bits.U2TXIE = 1;  //Enable UART1 Transmitter ISR
       U2STAbits.UTXISEL = 1; //Setup UART1 transmitter to interrupt
                             //when a character is transferred to the
                             //Transmit Shift register and as result,
                             //the transmit buffer becomes empty.

       U2STAbits.UTXEN = 1;  //Enable UART1 transmitter
       UARTCharPtr = &UartData[0]; //Initialize UARTCharPtr to point
                             //to the first character in the Display
                                buffer
}

//WriteUART_to_RS232() triggers interrupt-driven UART communication by
   writing
```

```
//the first character in the Display buffer to the UART Transmit
    register
void WriteUART_to_RS232(void)
{
        if ((UARTCharPtr > &UartData[0]) &&
                (UARTCharPtr < &UartData[15])) return;
        else
        {
                UARTCharPtr = &UartData[0]; //Re-Initialize UART display
                                            //buffer pointer to point to
                                            //the first character
                U2TXREG = *UARTCharPtr++;   //Load the UART transmit
                                            //register with first
                                                character
        }
}


//_U2TXInterrupt() is the UART1 Interrupt Service Routine.
//The routine must have global scope in order to be an ISR.
//The ISR name is the same name provided for the module in the device
    linker
//script.
//The UART1 ISR loads the UART2 4-deep FIFO buffers with the next
//4 characters in the Display buffer unless it encounters a null
    character.
void __attribute__((__interrupt__)) _U2TXInterrupt(void)
{
        int i = 0;
        while ((*UARTCharPtr != '\0') && (i < 4))
        {
                U2TXREG = *UARTCharPtr++;
                i++;
        }
        IFS1bits.U2TXIF = 0;  //Clear the UART1 transmitter interrupt
            flag
}


void __attribute__((__interrupt__)) _U2RXInterrupt(void)
{
       int temp=U2RXREG;
   // PORTD=0xFFFF;
```

```
      if(temp=='#')
     {
// startsine();
     asm("RESET");

       }
       IFS1bits.U2RXIF = 0;  //Clear the UART2 transmitter interrupt
          flag
}
```

B.7 GLCD.c

```
#include<p30f4013.h>
//Disable Code Protection
// Glcd module connections
#define GLCD_DataW LATB
#define GLCD_DataR PORTB
#define GLCD_Dir  TRISB
#define GLCD_CS1 LATCbits.LATC13
#define GLCD_CS2 LATCbits.LATC14
#define GLCD_RS LATBbits.LATB10
#define GLCD_RW LATBbits.LATB9
#define GLCD_RST LATDbits.LATD3
#define GLCD_EN LATDbits.LATD0

#define D_CS1 TRISCbits.TRISC13
#define D_CS2 TRISCbits.TRISC14
#define D_RS TRISBbits.TRISB10
#define D_RW TRISBbits.TRISB9
#define D_RST TRISDbits.TRISD3
#define D_EN TRISDbits.TRISD0
void clrall(void);
extern void Delay5us(int);

void enable(void)
{
//Delay_us(1);
GLCD_EN=1;
Delay5us(3);
GLCD_EN=0;
}
```

```
void glcdon(void) //GLCD on function to switch on
{
D_CS1=0;
D_CS2=0;
D_RS=0;
D_RW=0;
D_RST=0;
D_EN=0;

//GLCD_Dir=0X0000;
GLCD_RST=0;
enable();
GLCD_RST=1;
GLCD_CS1=1;
GLCD_CS2=1;
// on
GLCD_RS=0;
GLCD_RW=0;
GLCD_DataW=0x003F;
enable();
clrall();
}
void xselect(unsigned char x)
{
if(x>=64)
   {
      GLCD_CS1=0;
      GLCD_CS2=1;
      x=x-64;
   }
else
   {
      GLCD_CS2=0;
      GLCD_CS1=1;
   }
GLCD_RS=0;
GLCD_RW=0;
GLCD_DataW=(0x0040 | x);
enable();
}
void yselect(unsigned char y)
{
   GLCD_RS=0;
   GLCD_RW=0;
   GLCD_DataW=(0x00B8 | y/8);
enable();
}
void clrall(void)
{
   xselect(0);
   GLCD_CS2=1;
   int i,j;
   for(j=0;j<8;j++)
   {
      yselect(j*8);

      for(i=1;i<=64;i++)
      {

         GLCD_DataW=0x0000;
         GLCD_RS=1;
         GLCD_RW=0;
         enable();

      }
   }
}
void clrx(unsigned char x)
{
   xselect(x);
   int j;
   for(j=0;j<8;j++)
   {
      yselect(j*8);
      GLCD_DataW=0x0000;//
      GLCD_RS=1;
      GLCD_RW=0;
      enable();

   }
}
void write(unsigned char x,unsigned char y,unsigned char s)
{
   xselect(x);
   yselect(y);
   switch (s)
```

```
    {
        case 0:        //Light spot
            GLCD_DataW = ~(1<<(y%8));
        break;
        case 1:        //Dark spot
           GLCD_DataW = (1<<(y%8)) ;
        break;
    }
   GLCD_RS=1;
   GLCD_RW=0;
   enable();
   GLCD_CS1=0;
   GLCD_CS2=0;
   GLCD_RS=0;
   GLCD_RW=0;
}
void WriteToGlcdX(int frequency,int phase)
{
   int xaxis=(int)(frequency-4014)*(float)0.025163463;
   clrx(xaxis);
   int temp=(int)(10000+phase)*(float)0.0032;
   write(xaxis,63-temp,1);
}
```

B.8 Delay.asm

```
;Symbol/Literal/Immediate Operand Definitions:
;".equ" directives are similar to "#define" pre-processor directives in
    C
.equ    NANOSEC62, 2         ;62.5ns approx = 2*TCY at 29.5 MIPS
.equ    MICROSEC, 16*NANOSEC62 ;16*62.5ns =1us
.equ    MILLISEC, 1000*MICROSEC ;1000*1us = 1000*16*62.5ns = 1ms

;Global Declarations for Functions and variables:
.global _Delay5us
.global _Delay5ms

;Code Sections:
;Code sections are named ".text"
.section .text
```

```
_Delay5us:
;Function Prototype for C:
; void Delay5us(int Count)
; Note: "_" prefixed to the function name is not used when calling the
;      function from a C file.
; Function Execution Time when Count=1 is 5 microseconds at 29.5 MIPS
;
       push   w1             ;Store w1 on to stack
       mov    #MICROSEC, w1  ;w1 = MICROSEC
       dec    w1, w1         ;w1 = w1 - 1
       bra    nz, $-2        ;If w1 != 0 then Branch to previous
          instruction
       dec    w0, w0         ;else w0 = w0 - 1
       bra    nz, $-8        ;If w0 != 0 then Branch back 4 instructions
       pop    w1             ;else restore w1 from stack
       return                ;Return to calling routine

_Delay5ms:
;Function Prototype for C:
; void Delay5ms(int Count)
; Note: "_" prefixed to the function name is not used when calling the
;      function from a C file.
; Function Execution Time when Count=1 is 5 milliseconds at 29.5 MIPS
       push   w1             ;Store w1 on to stack
       mov    #MILLISEC, w1  ;w1 = MILLISEC
       dec    w1, w1         ;w1 = w1 - 1
       bra    nz, $-2        ;If w1 != 0 then Branch to previous
          instruction
       dec    w0, w0         ;else w0 = w0 - 1
       bra    nz, $-8        ;If w0 != 0 then Branch back 4 instructions
       pop    w1             ;else restore w1 from stack
       return                ;Return to calling routine


.end                         ;End of File
```

Appendix C

Interface code for PC or laptop

C.1 GUI.m

```
function varargout = gui(varargin)
clc;
% GUI M-file for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*. Starting from the left, property value pairs
   are
%      applied to the GUI before gui_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property
   application
%      stop.  All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
   one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 23-Feb-2016 18:30:24

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',   mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @gui_OpeningFcn, ...
    'gui_OutputFcn', @gui_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% varargin  command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in getdata.
function getdata_Callback(hObject, eventdata, handles)
% hObject   handle to getdata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
if strcmp(get(handles.connect,'String'),'Disconnect')
    if strcmp(get(handles.getdata,'String'),'Stop')
        set(handles.getdata,'String','Wait');
    else
        set(handles.getdata,'String','Stop');
        serialport(hObject, eventdata, handles);
    end
else
    set(handles.status,'String','Connect to port');

end

% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)
% hObject   handle to browse (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
[filename,
    pathname]=uiputfile({'*.xls';'*.xlsx'},'MultiSelect','on','Select
    files to execute');


% --- Executes on button press in connect.
function connect_Callback(hObject, eventdata, handles)
% hObject   handle to connect (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

if strcmp(get(handles.connect,'String'),'Connect')
    global s;
    s = serial(get(handles.com,'String'),'BaudRate',9600);
    fopen(s);
    set(handles.connect,'String','Disconnect');

else
    global s;
    fclose(s)
    delete(s)
    clear s
    set(handles.connect,'String','Connect');

end


% --- Executes on button press in savefile.
function savefile_Callback(hObject, eventdata, handles)
% hObject   handle to savefile (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
if(get(handles.savefig,'Value'))
    set(handles.datafile,'Enable','on','string','C:\---\phase.xlsx')
else
    set(handles.datafile,'Enable','off')
end
% Hint: get(hObject,'Value') returns toggle state of savefile

%Disconnect serial port at the time of closing window
% --- Executes during object deletion, before destroying properties.
function uipanel6_DeleteFcn(hObject, eventdata, handles)
% hObject   handle to uipanel6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

if strcmp(get(handles.connect,'String'),'Connect')
    disp('Thank You')

else
    global s;
    fclose(s)
    delete(s)
    clear s
    set(handles.connect,'String','Connect');
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
    browse1.
function browse1_ButtonDownFcn(hObject, eventdata, handles)
% hObject   handle to browse1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
[file,path] = uiputfile('phase.xlsx','Save file
    name',get(handles.datafile,'String'));
if path~=0
    set(handles.datafile,'String',strcat(path,file));
end
```

C.2 serialport.m

```
function [ ] = serialport(hObject, eventdata, handles)
%This code will receive data from instrument
clc
%Initialize
out='xxxxxxxxxx';
data='xxxxxxxxxx';
global s; %Take serial port specification from gui
%get number of set required by user
y=str2double(get(handles.npoints,'String'));
%loop runes upto desired set
for c=1:y
    fprintf(s,35) %Send 35 or '#' to start from begining
    set(handles.status,'String','Waiting for data...' ); %change status
    data=fscanf(s); %scan port to receive first data
    disp('Received first information') %display
    set(handles.status,'String',strcat({num2str(c)},{' out of'},{' '},
        {num2str(y)} ) ); %show running set number to user
    %Transmition started
    set(handles.waitxt,'String',strcat( 'Reciving data...' ) );
```

```
i=1; %set counter
while 1 % goto infinite loop until EOF reach
    data=fscanf(s); %scan port until receive '\n' or new line command
    %Analysed scaned value is there EOF if yes then extract
        resonance and brake the loop
    if data(2)=='E' || data(3)=='O' || data(4)=='F'
        resonance(c)=str2num(data(17:21));
        break;
    end
    %get frequency,phase,phaseP(int),phaseN(int)
    freqph(i,1)=str2double(data(1:5));
    freqph(i,2)=str2num(data(7:13));
    freqph(i,3)=str2double(data(15:18));
    freqph(i,4)=str2double(data(20:23));

    axes(handles.axes1); %select axes
    cla;
    plot(freqph(1:size(freqph,1),1),freqph(1:size(freqph,1),2));
    drawnow; %draw graph now
    save test.mat %save current variable for testing purpose
    i=i+1; %increment data counter
end % end of one data set
dt=datestr(now,'yyyy_mmm_dd_HH_MM_SS'); % get current time from
    computer
xlswrite(get(handles.datafile,'String'), freqph, dt ); %save to .xls
    file with sheet name current time and file name as user provided
    in GUI
instant(c)={dt}; %time also stored for resonance frequency list
axes(handles.axes2); %Select second axes to plot instant vs resonance
cla;
plot(resonance, 'MarkerEdgeColor','k',
    'MarkerFaceColor','g','MarkerSize',10); %plot instant vs
    resonance
%is user requested for stop? then return
if ~strcmp(get(handles.getdata,'String'),'Stop')
    set(handles.getdata,'String','Get Data');
    return;
else
    set(handles.getdata,'String','Stop');
end
if i~=y % show status if it is not a last set
    set(handles.status,'String',strcat({num2str(c)},{' out of'},{'
        '}, {num2str(y)},{' '},{'recived'} ) );
        set(handles.waitxt,'String',strcat( {'Waiting for'},{'
            '},{get(handles.wait,'String')},{' sec'} ) );

        %set(handles.status,'String',strcat('Recived ',num2str(c),' out
            of ', num2str(y), 'wait ',get(handles.wait,'String'),'sec' )
            );
        pause(str2double(get(handles.wait,'String'))); % wait for
            desired time
    end
end %completion of all set
%write resonance to .xlsx file with sheet name 'Resonance'
xlswrite(get(handles.datafile,'String'), instant', 'Resonance','A' );
xlswrite(get(handles.datafile,'String'), resonance', 'Resonance','B' );
set(handles.waitxt,'String','Completed'); % show status complete
set(handles.getdata,'String','Get Data'); % rest button text to get data
end %End function serial
```

Bibliography

[1] Ostell. [Online]. Available: www.osstell.com

[2] N. Meredith, D. Alleyne, and P. Cawley, "Quantitative determination of the stability of the implant-tissue interface using resonance frequency analysis," *Clinical oral implants research*, vol. 7, no. 3, pp. 261–267, 1996.

[3] E. Sunil, A. Chakraborty, R. Ghosh, and B. Goswami, "Design of transducers for resonance frequency measurement to assess the dental implant stability in vitro," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE.* IEEE, 2007, pp. 2158–2162.

[4] S. Ghosal, B. Goswami, R. Ghosh, and P. Banerjee, "Determination of stability of dental implant from impedance studies using resonance frequency analysis," in *Emerging Applications of Information Technology (EAIT), 2011 Second International Conference on.* IEEE, 2011, pp. 71–74.

[5] P. Banerjee, R. Ghosh, and B. Goswami, "Modeling of a sensor used for rfa based dental implant stability measurement," in *Informatics, Electronics & Vision (ICIEV), 2012 International Conference on.* IEEE, 2012, pp. 242–247.

[6] M. Baruah, P. Banerjee, A. Nandy, B. Goswami, and R. Ghosh, "A low cost instrumentation system for determining stability of dental implants," in *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on.* IEEE, 2012, pp. 81–84.

[7] D. Chowdhury, B. Goswami, and R. Ghosh, "Monitoring the change in resonance frequency of a dental implant in hardening dental plaster using microcontroller," in *Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference of.* IEEE, 2014, pp. 89–92.

[8] Microchip, *dsPIC30f family refference manual*, 2006. [Online]. Available: ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf

[9] ——, *dsPIC30f4013 datasheet*, 2004. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/70138c.pdf

[10] ——, *MPLABC30*, 2007. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/c30_users_guide_51284f.pdf

[11] N. S. Corporation, *DAC0808 8-Bit D/A Converter*, 1999. [Online]. Available: http://www.ti.com/lit/ds/symlink/dac0808.pdf

[12] wikipedia. (13 April 2016) Low-pass filter. [Online]. Available: https://en.wikipedia.org/wiki/Low-pass_filter

[13] *ELECTRONIC DEVICES AND CIRCUIT THEORY*. PRENTICE HALL.

[14] M. Corporation, *Basic Total Harmonic Distortion (THD) Measurement*, 2015. [Online]. Available: http://www.microsemi.com/document-portal/doc_view/134813-an30-basic-total-harmonic-distortion-thd-measurement

[15] Adam. (2009) Sine to square wave converter. [Online]. Available: http://www.simplecircuitdiagram.com/2009/12/11/self-powered-sine-wave-to-square-converter/

[16] Microchip. (2005) dspicdem2. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/dsPICDEM2%20Overview%20Document%20DS70147a.pdf

[17] L. M. SPECIFICATION, *JHD12864E*, 10/12/03.