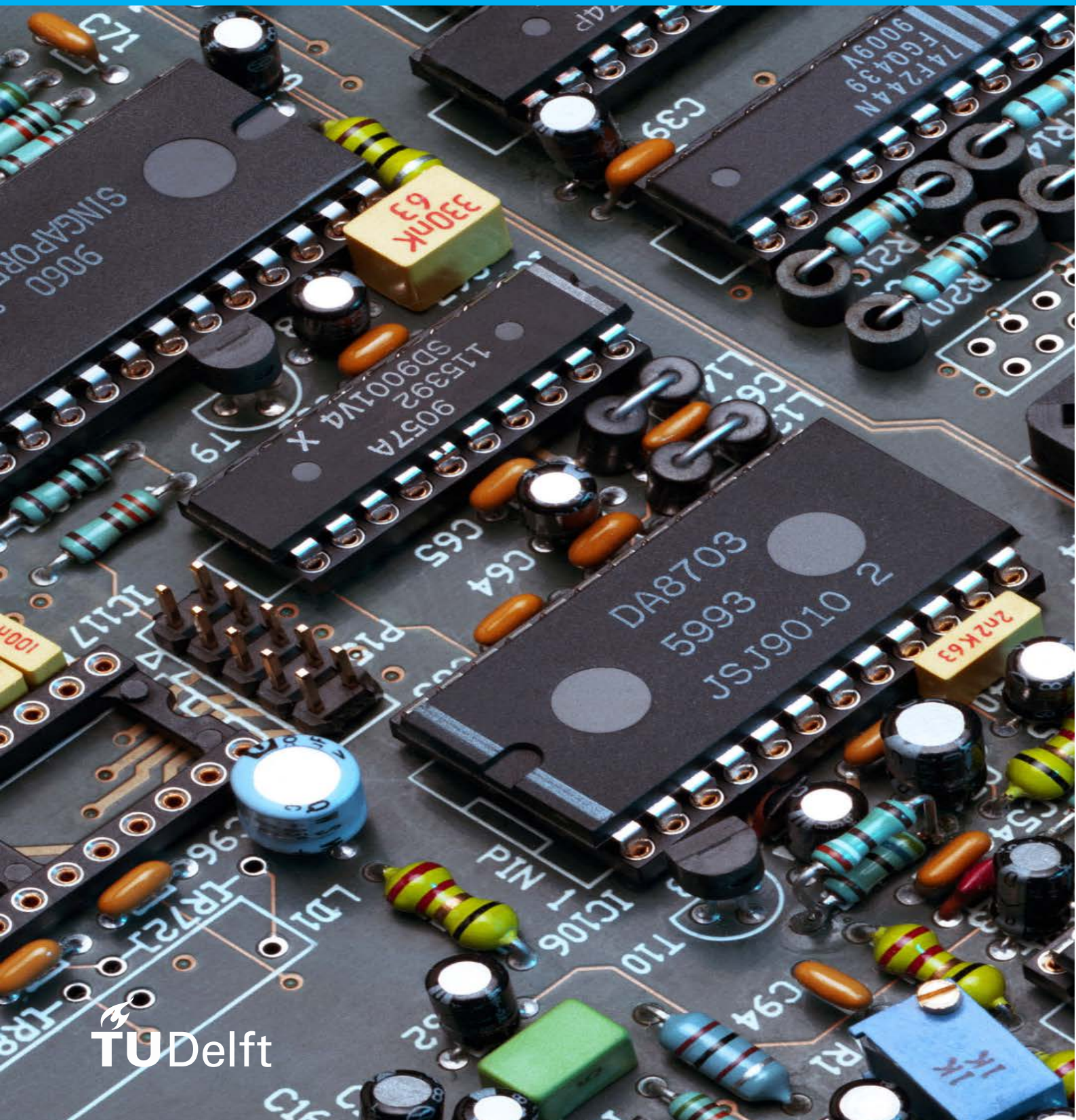
An atto-Farad resolution closed loop impedance measurement bridge for capacitive sensors

Dezhi Lin



TU Delft

An atto-Farad resolution closed loop impedance measurement bridge for capacitive sensors

By

Dezhi Lin

in partial fulfilment of the requirements for the degree of

Master of Science
in Electrical Engineering

at the Delft University of Technology,
to be defended publicly on Tuesday August 20, 2018 at 15:00 PM.

Supervisor:	Dr.ir. G. de Graaf	
Thesis committee:	Dr.ir. M.A.P. Pertijs,	TU Delft
	Dr. F. Sebastiano,	TU Delft
	Dr.ir. G. de Graaf,	TU Delft

TU Delft

Abstract

This thesis discusses the theory, circuit design, software design and measurements of an atto-Farad resolution closed-loop impedance measurement bridge circuit for capacitive sensors implemented using commercial off the shelf components.
A lock-in amplifier (LIA) method is used here. The capacitive sensor embedded in an impedance bridge is driven at 10MHz or higher by two inverting excitation sources (AD9959) that can adjust frequency, phase and amplitude with a certain resolution (32 bits, 14 bits and 10bits respectively). These parameters can be set through a graphical user interface (GUI). When the output signal is nulled by changing the amplitude of the excitation signal, the unknown capacitor value can be calculated. A simple test impedance bridge has been fabricated to measure a fixed impedance value of the capacitor sensor using the LIA measurement approach. The obtained results (capacitance and resistance values) are in good agreement with what we obtained using an alternative approach (AH2700A, is an ultra-precision capacitance bridge with 0.16ppm resolution at 1000Hz). The circuit has a 24.1ppm resolution at 10Hz bandwidth when the input frequency is 10MHz.

Keywords: Capacitive sensor, Lock-in amplifier, Closed-loop impedance bridge

Acknowledgements

I would like to thank Mr. Ger. de. Graaf for his role as my supervisor during the whole research project. Furthermore, I would like to thank my friends, Guanchu Wang, Chengyu Huang and Xinyu Gao for their support during the toughest times of my thesis work. Also, I would like to thank Jiyuan Zeng, Yuanbo Ji, Jinyi Liu and Jiahan Lu who supported me during my two-year MSc academic career. Besides, my special thanks to my girlfriend Kuang Li for her understanding, support, and care during the two-year life in the Netherlands. Last but not least, I would like to give special thanks to my dear parents for their moral and financial support, which gave me a chance to study abroad in this fantastic university.

August, 2018

Dezhi Lin

Contents

List of Figures

List of Tables

Introduction

1.1 Motivation

Nowadays, sensors play an important role in various fields such as industry, defense, and communications. Since the development of MEMS technology in the 1960s, the share of MEMS sensors has continued to increase. Compared with traditional sensors, MEMS sensors have some characteristics: small size, light-weight, low cost, low power consumption, high reliability, suitable for mass production, and easy to integrate [1,2]. What's more, sensors and processing circuits can be integrated on a single IC using CMOS technology, while having higher response speed and smaller package size than mechanical sensors. Meanwhile, the feature size at the micron level makes it possible to perform functions that some traditional mechanical sensors cannot achieve.

A capacitive sensor is a conversion device, based on capacitive coupling, that can measure and detect a physical quantity or mechanical quantity with the capacitance change. It is widely used in the measurement of displacement, pressure, humidity, the composition of compounds and so on. In particular, with the continuous feature of MEMS design and processing technology, lots of capacitive sensor products have been developed. For example, MEMS devices with comb drives [3], which are widely used in a variety of sensor applications, such as those that measure the position, speed and acceleration of moving objects, force, pressure, liquid levels, dielectric properties and flow materials [4]. Among the popular transduction mechanisms, capacitive sensing has been widely used because of its good noise performance, low-temperature coefficient, high sensitivity, and excellent compatibility [5]. The disadvantage is that the processing circuit is more complex [6]. The small size of the MEMS sensors determines that the capacitance of the sensitive capacitor is unlikely to be large, typically in the pF range. The change in the micro-capacitance caused by these physical quantities is even smaller, typically fF or even aF. Such a small amount of the capacitance change is a challenge to the design of the detection circuit. Conventional methods of building detection circuits using discrete components are not able to adapt to the decreasing trend of sensor capacitances. The use of dedicated interface integrated circuits for detection and processing can be the first choice for capacitive sensors.

According to the impedance spectroscopy technique [7], these sensors (especially gas sensor in this design) can be equivalent to a simple circuit shown in Figure 1-1, due to the properties of the dielectric materials, there is a parallel parasitic resistance (shunt resistance) as the loss term. Typical values for this resistance are usually hundreds of kiloohms to hundreds of megaohms when the sensing capacitance is in the range of picofarads. This model can help researchers to analyze and measure them easily.

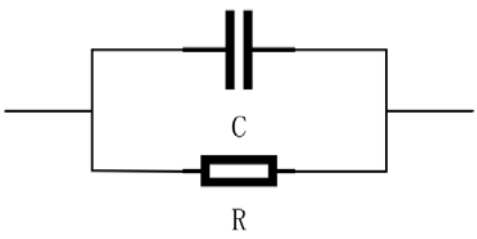


Figure 1-1 A simple electrical model of the capacitive sensor

In some applications, the change in the capacitance value due to a measurand is much smaller than the sensor offset capacitance [8]. There is a resolution problem when this offset capacitance is not stable. A half-a-bridge structure can be a possible solution [9]. This solution also has the advantages of CM rejection of interfering effects, including temperature drift, which means it can compensate for the environmental effects by using this structure [9]. To obtain high accuracy and resolution impedance sensing, as shown in Figure 1-2, a bridge configuration is chosen, where the unknown impedance is compared against a high accuracy or identical reference impedance.

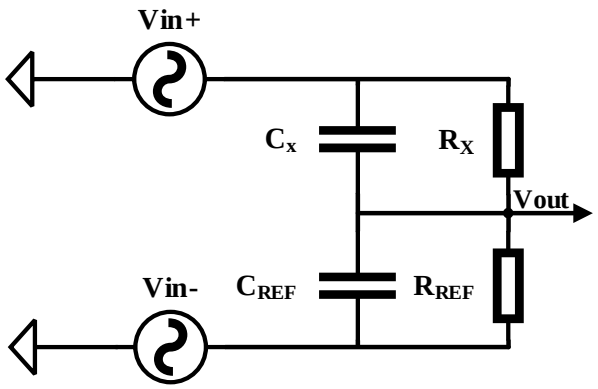


Figure 1-2 Half bridge configuration of a capacitive sensor

In many modern sensor applications, especially in MEMS devices, this reference sensor can be an identical sensor which is not exposed to the physical quantity to be measured. For instance, in a gas sensor, an identical sensor can be the reference which is in a reference gas at the same temperature or environment.

1.2 Readout approaches

Current-voltage (I-V), bridge and resonant methods are three fundamental ways to measure impedance (including capacitance).

For I-V method, the basic structure is shown in Figure1-3. According to the Ohm's law, the unknown impedance (Zx) can be calculated from measured voltage and current values. Calculate current using the voltage measurement across an accurately known reference resistor (R), Z_x can be expressed as:

$$Z_X = \frac{V_1}{I} = \frac{V_1}{V_2} R \qquad (1-1)$$

In practice, placement of resistor at high end impedes the requirement to accurately measure the differential signals in the presence of high common-mode voltages. Usually, a specialized operational amplifier is used for this purpose. Sometimes an RF transformer is used in place of R to obtain the high-end sensor [10]. The transformer, however, limits the low end of the applicable frequency range [11].

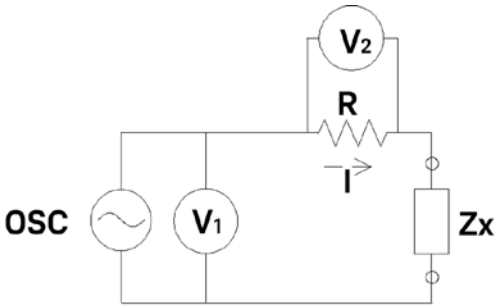


Figure 1-3 Simple structure of I-V method

The I-V method has a simple structure. It is mainly suitable for grounded device measurement and suitable to probe-type test needs. However, this method has low accuracy, and the operating frequency is limited based on the transformer used in the probe [12].

The basic structure of the bridge method is shown in Figure 1-4, connect the oscillator or signal generator to the two ends of the AC bridge. The four components of the bridge are Z_1, Z_2, Z_3, and Z_X, respectively. When the bridge reaches equilibrium (adjust the reference impedance (Z_2) until no current flows through the detector (D)), and the relationship between the unknown impedance (Z_X) and the other three components is:

$$Z_X = \frac{Z_1}{Z_2} Z_3 \qquad (1-2)$$

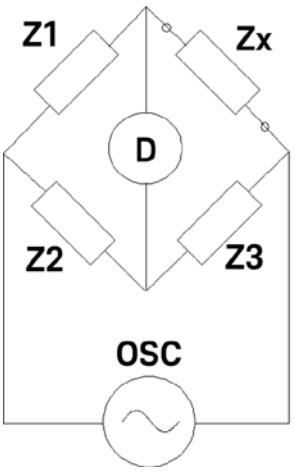


Figure 1-4 Simple structure of bridge method

Various types of bridge circuits for various applications use different combinations of L, C and R components to act as bridge elements. The bridge method has high accuracy, but because of the need to balance the bridge, it does not work in fast, repeated and continuous measurement.

As shown in Figure 1-5, the electrical model of the resonant approach in series mode is presented (parallel mode is also feasible, series and parallel connections are available for a wide range of impedance measurements). Adjust the oscillator or signal generator frequency to make the circuit resonate. At resonance, the series impedance of the RLC tank is at a minimum value, the capacitive reactance of C_X (often accompanied by a leakage resistance R_X) and the inductive reactance of L are equal ($1/\omega \cdot C_X = \omega \cdot L$, where ω is the angular frequency of oscillator), so that C_X can be obtained. Due to the very low loss of the measurement circuit, Q values (quality factor) should be as high as possible for the resonant method.

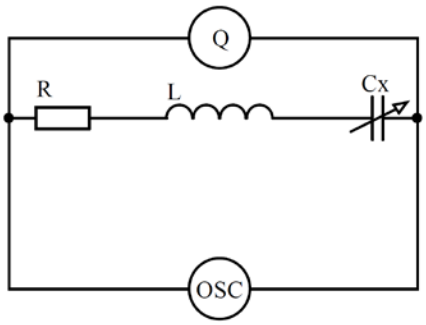


Figure 1-5 Basic structure of the resonant method

The resonant approach shows good accuracy in quality of inductance measurement (good Q accuracy up to high Q), but it has a low impedance measurement accuracy, and there is a need for resonance tuning [12].

Meanwhile, all these methods are sensitive to parasitic capacitances, which can impair the measurement capacitance and need more complex circuitry to eliminate their effect [13].

In the last few years, there have been many articles reporting the high-resolution readout circuits of the capacitive sensor [14-25]. These interface designs of the high-precision capacitive sensor are mainly divided into two directions: a continuous time processing method and a discrete time processing method.

1.2.1 Discrete-time processing method

Discrete-time processing circuits are mainly switched-capacitor (SC) readout circuit, including analog switches, capacitors, and operational amplifiers. It works by moving charge into and out of the capacitor when the switch is on and off, which makes them more suitable to use within integrated circuits, where the precisely specified resistors and capacitors are not economical to build [26]. So, this kind of way has been proposed in some articles with capacitive sensor [14,15,16], including a capacitive gas sensor [17]. The capacitive sensing is based on the capacitance-to-voltage converter, the same foundation on which SC circuit operates. The SC circuit provides a virtual ground and robust dc biasing at the sensing node so that the sensed signal is insensitive to parasitic capacitance and undesirable charging [27]. However, the drawbacks of this readout circuit are also apparent: Clock Feedthrough, Channel Charge Injection, Noise Aliasing, Input Signal Bandwidth Limits and so on. The basic structure of the SC circuit is shown in Figure 1-6.

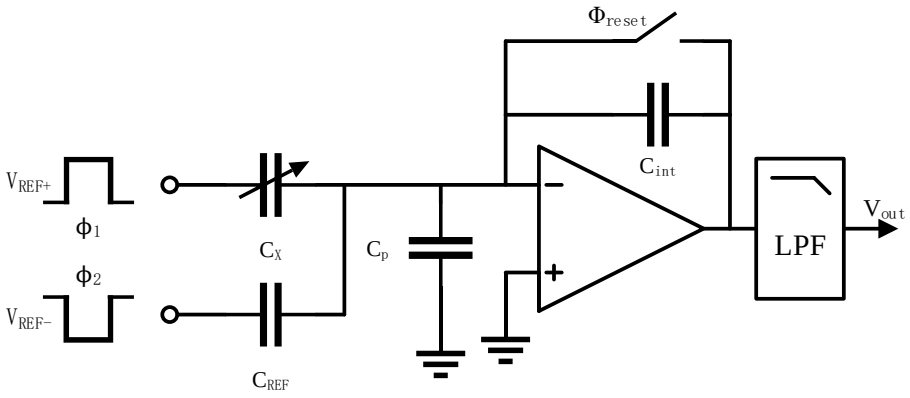


Figure 1-6 Switch-capacitor circuit

In a switched capacitor circuit, the sense (C_X) and reference (C_{REF}) capacitors are charged with opposite polarity voltages and a packet of charge proportional to the capacitance difference is integrated on the input feedback capacitor (C_{int}) [28], so the output voltage can be expressed as:

$$V_{out} = V_{ref} \frac{\Delta C}{C_{int}} \qquad (1-3)$$

For a general case, a single sense capacitor with a fixed reference capacitor equal to the sense rest capacitance ($C_{REF}=C_{X0}$) is assumed [28]. The readout circuit detects the capacitance change ($\Delta C= C_X-C_{REF}$).

The correlated double sampling (CDS) technique is usually used to eliminate 1/f noise of the circuit [29-32] effectively. The wideband thermal noise sources of the amplifier and the switches are sampled at the high impedance nodes of the circuit and aliased into the baseband frequency range. That of feedback capacitor dominates the sampled switch noise (also referred to as kT/C noise) because this capacitance is typically small to increase the output voltage. Also, by appropriately selecting the circuit topology, the sampling noise of the sensing and reference capacitor remains the same and cancel each other [28].

1.2.2 Continuous-time processing method

For continuous-time processing method, there are mainly two approaches to measure the capacitance of capacitive sensor: ac-bridge with voltage amplifier [18-21]and trans-impedance amplifier [22-25] with the fundamental principle of lock-in amplifier (LIA) techniques.

The basic structure of LIA is shown in Figure 1-7. An input signal (signal of interest) with high frequency is generated, compared with thermal noise, the flicker noise of the amplifier should be inessential at this time. Then, the signal with noise components is amplified and go through a phase sensitive detector (mixer, chopper or demodulator, which acts as a synchronous rectifier) via reference signal (sinewave or square wave). After this process, the signal of interest is demodulated back to the baseband, while the flick noise of the amplifier is modulated to a higher frequency. A low-pass filter with a suitable cutoff frequency will filter out the flicker noise and obtain a narrow noise bandwidth. Finally, an excellent output DC signal can be obtained without 1/f noise. The detail of this part will discuss in section 2.3.

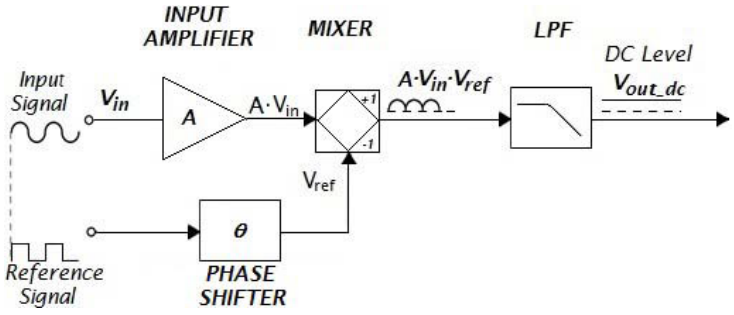


Figure 1-7 Principle of operation of LIA [33]

1.2.2.1 Ac-bridge with voltage amplifier

As shown in Figure 1-8, the basic configuration of ac-bridge is presented, it consists of a square-wave drive circuit, a half-bridge capacitive sensor, a voltage-mode amplifier, a synchronous demodulator, and a low-pass filter. The two ac signals (square wave) with 180˚ phase difference drive a half-bridge

consisting of the sense capacitance and reference capacitance. When a differential sensing capacitor is available, a full-bridge configuration can also be formed [28]. The amplitude of the bridge output is proportional to the capacitance change (ΔC), after this signal is amplified and demodulated, the output voltage can be expressed as:

$$V_{out} = V_{REF} \frac{\Delta C}{2C_{X0} + C_p} A_v \qquad (1-4)$$

where A_v is gain of the amplifier.

So, it can be seen that this kind of circuit is significantly affected by the parasitic capacitance (C_p), which will reduce the resolution of the readout circuit.

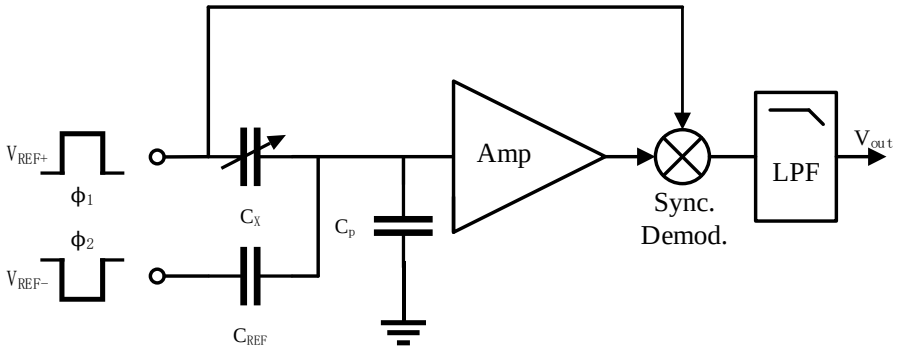


Figure 1-8 Basic structure of ac-bridge with voltage amplifier

1.2.2.2 Transimpedance amplifier

Figure 1-9 shows the transimpedance amplifier configuration. Similar to the ac-bridge with voltage amplifier, it is mainly composed of a sinewave drive circuit (The drive signal needs to be sinusoidal to avoid errors induced by harmonic distortion, and the phase difference between the two sine wave signals is maintained at 180˚), a half-bridge capacitive sensor, an operational amplifier, a synchronous demodulation circuit, and a low-pass filter circuit. Different from the ac-bridge with voltage amplifier, due to the presence of the op-amp and feedback resistor (R_F), the output of the half-bridge is held at the “virtual ground” point, which reduces the effect of parasitic capacitance (C_p). Meanwhile, because of the “virtual ground” point, the currents passing through Z_X (Input impedance) and Z_F (Feedback impedance) are balanced through the op-amp, and the current through the input impedance is proportional to the operational amplifier output voltage, the output voltage of the amplifier is given by:

$$V_{out} = -\frac{V_{in}}{Z_X} * Z_F \qquad (1-5)$$

where $Z_X = 1/(s*\Delta C)$ and $Z_F = R_F/(1+sR_FC_F)$. When R_F is bigger enough, Z_F can be approximated as $1/(s*C_F)$ and V_{out} can be expressed as:

$$V_{out} = -\frac{V_{in}}{\Delta C} * C_F \qquad (1-6)$$

The circuit can be regarded as a charge amplifier.

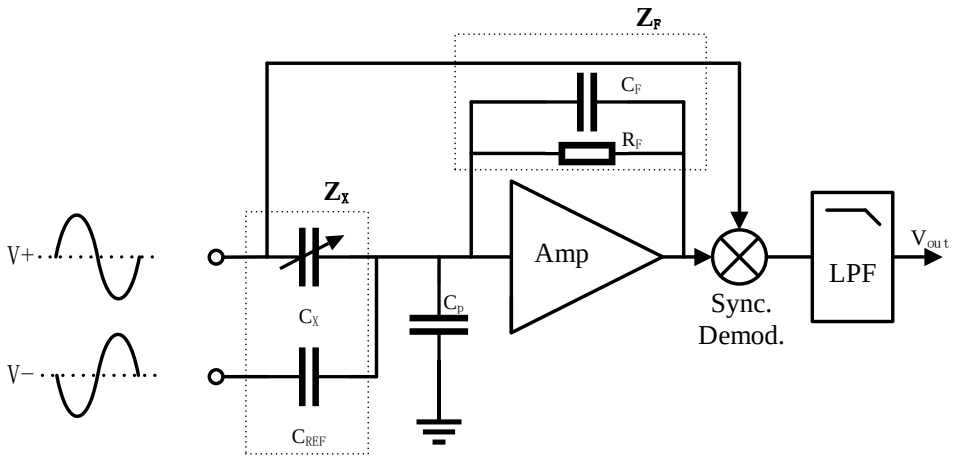


Figure 1-9 Transimpedance amplifier readout scheme

1.2.3 Comparison and choice

Compared with the SC readout method, the continuous-time detection method does not have the aliasing phenomenon (noise folding effects [34]) introduced in the discrete-time detection circuit due to the sampling principle. Also, charge integration using a switched capacitor front-end [35] suffers from the parasitic electrical coupling of switching noise, and a continuous-time (CT) charge integrator front-end [36] does not suffer from kT/C noise. Although SC readout circuit has high integration density and smaller area (switches area is smaller than resistor) by CMOS technology, it needs clock circuit to control switches and not suited for high frequency, which means this kind of circuit is complicated and cannot eliminate the effect of parasitic resistance very well. Therefore, continuous-time detection circuits that can operate at higher frequencies are a good choice. Also, for ac-bridge with voltage amplifier, there are several techniques, such as bootstrapping, that try to reduce the effect of these parasitic capacitances. In this technique, a unity-gain voltage amplifier is used along with a guard electrode surrounding the measurement electrodes to eliminate the voltage difference over them [37]. However, the resolution of capacitance is determined by the thermal noise floor of the amplitude and is still a function of the overall parasitic capacitance regardless of the feedback for boot-strapping [28]. Current measurement can be a good way to replace bootstrapping. It consists of a low-input-impedance transimpedance amplifier (TIA) that senses the current through the sensor and eliminates the voltage variations at the input nodes, which can minimize the effects of the capacitive parasitic [38]. So, the transimpedance amplifier with the LIA technique is chosen as a primary operation circuit in this design. In order to reach a higher resolution, the transimpedance amplifier with the LIA technique can operate in a closed-loop, which is called a self-balanced bridge or auto-balancing bridge. The unknown sensor impedance is obtained by balancing the bridge in “auto-tuning” configuration employing, as variable impedance, an automatic adjustable resistor or capacitor [39-42].

1.2.4 Basic principle and prior art of self-balanced bridge

The block diagram of the self-balanced bridge is shown in Figure 1-10.

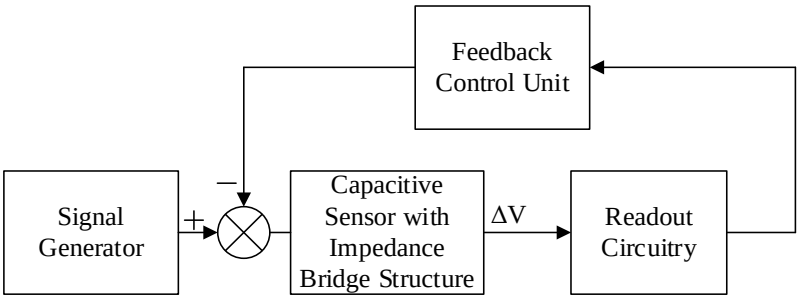


Figure 1-10 Block diagram of the self-balanced bridge measurement

In this technique, the signal generator generates AC signals to drive the impedance bridge. After the impedance bridge, there is an error voltage (ΔV), which is caused by the variation of capacitance. The readout circuit collects this voltage and sent to the feedback circuit for further processing. Then the feedback circuit produces a corresponding voltage signal or controls the signal generator to make the error voltage is equal to zero. Finally, the impedance bridge is in equilibrium. Generally speaking, a self-balanced strategy can be considered as a negative feedback-based system whose aim is to minimize or null a specific error signal [43]. The capacitance change can also be calculated from the voltage relationship of the feedback circuit, the signal generator and the reference impedance (The specific calculation will be discussed in Chapter 4).

1.2.4.1 The prior art

Some approaches based on the self-balanced bridge measurement have been published before. In this sub-section, some of the prior works are introduced, with a summary in the end.

- [P. Holmberg, IEEE Trans. Instrum. Meas. 1995] [44]

Figure 1-11 shows the block diagram of the capacitive sensor bridge circuit.

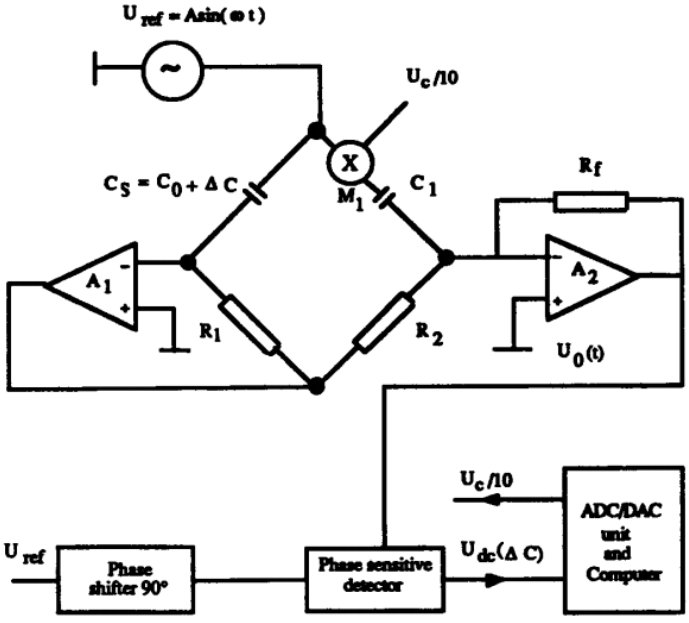


Figure 1-11 Schematic of the measurement system, where ΔC is the transducer variation of interest [44].

The impedance bridge circuit is originally of De Sauty type [45], which is an AC bridge works on the principle of Wheatstone's bridge. This bridge is used to determine the capacity of an unknown capacitor C_2 in terms of the capacity of a standard known capacitor C_3 [46]. R_1 and R_4 are pure resistors

(non-inductive resistors). The simplicity of this method is offset by the impossibility of obtaining a perfect balance if both the capacitors are not free from the dielectric loss [47]. A perfect balance can only be obtained if air capacitors are used. R_1, R_4, C_2 and C_3 are connected in a De Sauty type as shown in the Figure 1-12.

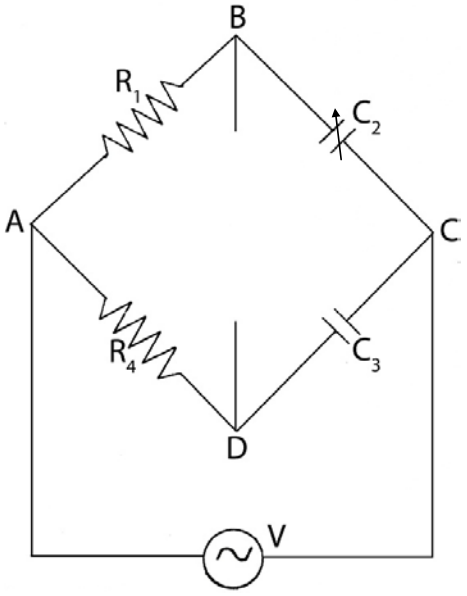


Figure 1-12 De Sauty bridge

However, in this approach (Figure 1-11), De Sauty bridge has been modified with two opamps (A_1 and A_2) and a multiplier (M_1). Amplifier A_1 set a zero voltage, and a capacitive current is generated that depends on Cs through R_1, and amplifier A_2 is a current-to-voltage converter to build a virtual ground (Eliminate the influence of parasitic capacitance). Any capacitance change (ΔC) generates a voltage at the A_1 output, which in turn develops a current for A_2 [44]. The multiplier (M_1) and the control signal (U_C) are used to balance the bridge circuit, or a voltage-controlled amplifier can be used to replace the multiplier (M_1). Through a control signal, the main task of this measurement system is to adjust the amplitude of an ac signal in an electronic way. U_C is selected in such a way that the bridge output (U_{dc}) is zero.

- [P. Mantenuto, IEEE Sensors J. 2014] [42]

This measurement technique also presents a capacitance-to-voltage conversion work in continuous time, by using a particular impedance bridge based on the modified De Sauty bridge structure (Fig. 1-13): a reference capacitance (C), a sensing capacitance (C_{SEN}), a fixed resistance (R) and a voltage-controlled resistor (VCR). In some proposed papers [48,49], the analog multiplier AD633 is used as VCR (named R_{VCR}), the equivalent resistance value can be expressed as:

$$R_{VCR} = \frac{10R_{IN}}{10 - V_{CTRL}} \qquad (1-7)$$

where R_{IN} is a user-settable internal resistance, and V_{CTRL} is the external signal (It should be a DC voltage with -10V to 10V range) to adjust the VCR properly.

In order to maintain the balance of the impedance bridge, a feedback circuit can be used to adjust the output of the multiplier (AD633) to generate a signal V_A that tends to follow V_B, the error signal $\Delta V = A(V_A - V_B)$ is therefore forced to zero, where A is the amplifier gain. Then, according to synchronous demodulation of ΔV performed with the mixer (Demodulator multiplier), while using the voltage

integrator (OA, R_{INT}, and C_{INT}) and the voltage divider (R_{D1}, R_{D2}), the high-frequency signal components are removed and provide the useful information (V_{CTRL}). The feedback signal V_{CTRL} is DC component of the error signal ΔV. So, the capacitance variation can be expressed as:

$$C_{SENS} = \mathrm{C}\frac{R_{IN}}{R}\left(\frac{10}{10 - V_{CTRL}}\right) \qquad (1-8)$$

in case of the bridge in equilibrium (ΔV = 0).

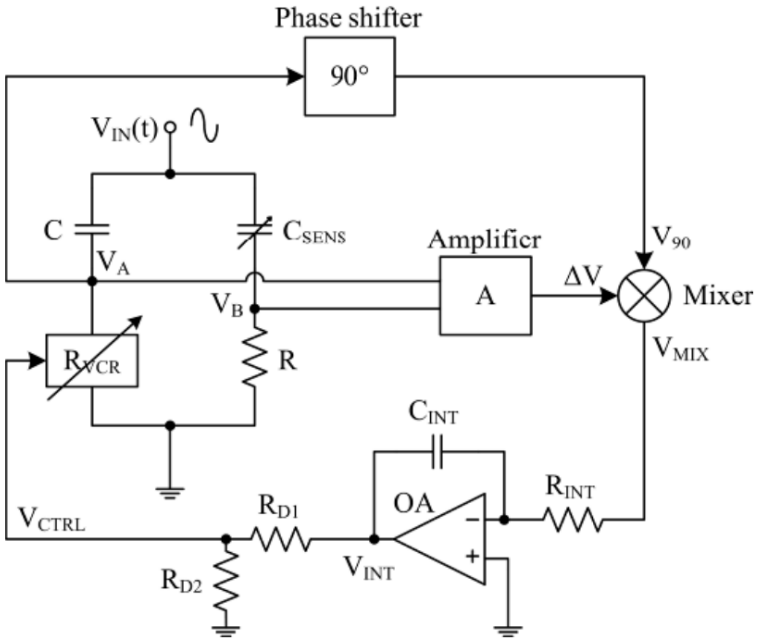


Figure 1-13 The interface with VCR balance [42]

- [B. Hu, Meas. Sci. Technol. 2016] [50]

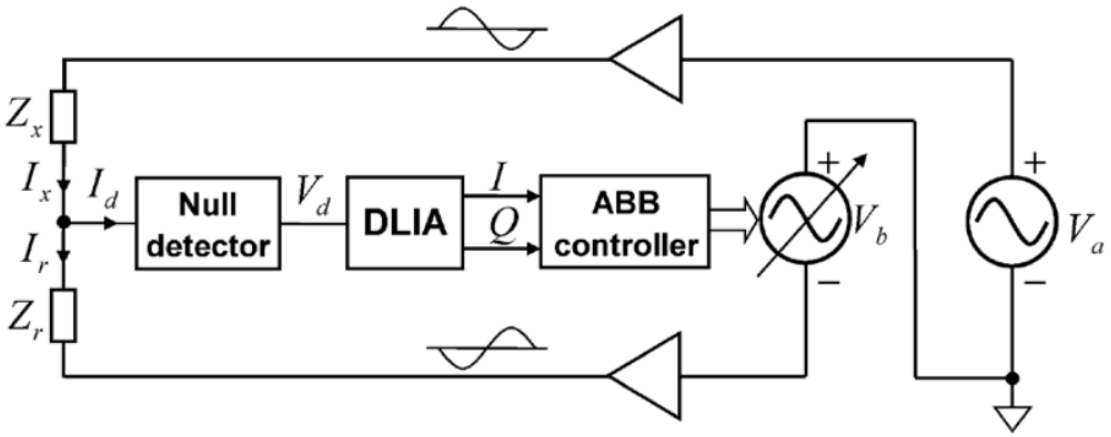


Figure 1-14 Block diagram of the digital auto-balancing bridge [50].

As shown in Figure 1-14, the main signal source V_a generates an ac signal to the DUT (Z_x). If the range resistor current I_r is not equal to the DUT current I_x, an unbalanced current Id is generated and flows into the null detector (D). DLIA demodulates the error voltage V_d into I (In-phase) and Q (Quadrature). After the ABB controller obtains them, a DSP algorithm is run to continuously adjust the amplitude and phase of the second signal source V_b. This signal is fed back through the range resistor Z_r to eliminate the DUT current. Therefore, the unbalanced current is closed to zero and the value of DUT

can be calculated.

There are also some instruments with self-balanced bridge method for impedance analysis, such as: Keysight 4294A [51], Keysight E4991B [52], Solartron 1260A [53], Andeen-Hagerling 2700A [54] and so on. For example, AH2700A has 0-20KHz frequency range and -0.165μF to 1.65μF capacitance range, the best resolution of the device is 0.16ppm @ 1KHz, it only needs 0.4s to do a full precision measurement and 0.03s to repeat measurement on the same DUT.

1.2.4.2 Summary of Prior Art

In conclusion, the basic measurement principles for measuring capacitance variation based on the self-balanced bridge are known. The introduced techniques have mainly been implemented using PCBs or bench-top instruments.

For the De Sauty bridge, it gives accurate results only when the capacitances without dielectric losses, which means if capacitive sensors have parasitic resistances, this kind of bridge will cause some errors. The resolution of the capacitive sensor can also be affected by the resistance (thermal noise) in De Sauty bridge.

In addition, there are also some points that have not yet been investigated:

1. Use a simpler circuit structure.
2. Increase the upper-frequency limit of the input signal to eliminate the effect of parasitic resistance.
3. Improve the system resolution with interpolation method (software).

So, a simpler structure, higher resolution self-balanced bridge system with wide frequency range will be presented in this thesis.

1.3 Our solution

Our approach uses two impedances (sensors) in a half-bridge impedance measurement bridge configuration, as shown in Figure 1-15. A four-channel DDS (Direct Digital Synthesis) chips drive both impedances via buffer amplifiers. Both arms of the bridge are driven by sinusoidal signals with the same frequency and accurate phase relation. This can be conveniently implemented over a wide frequency range by modern DDS chips.

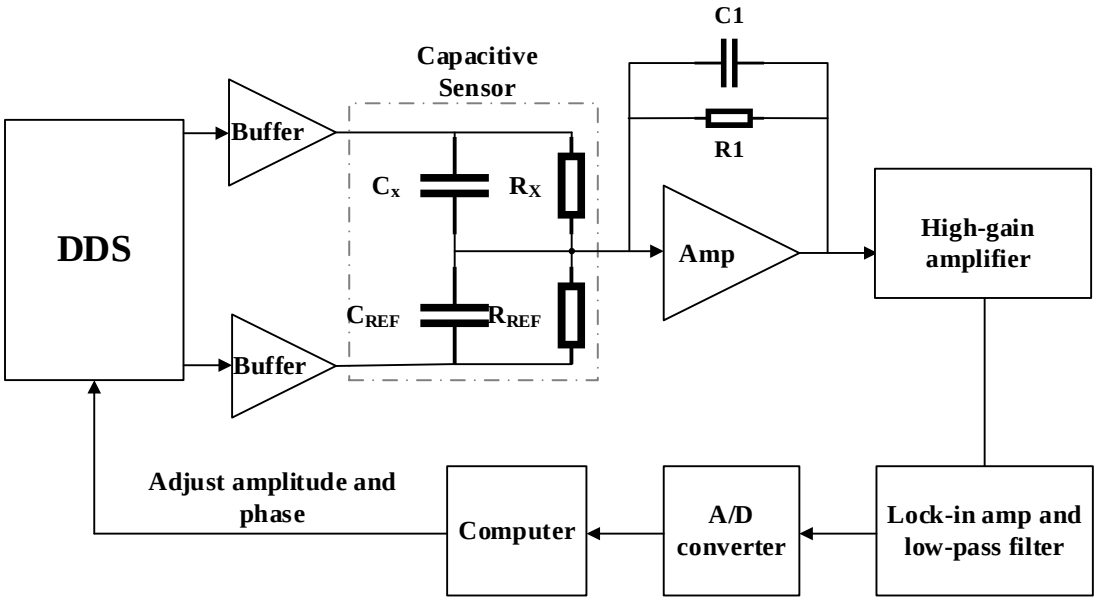


Figure 1-15 The whole structure of the readout system

Then the output of the impedance bridge is amplified by a high-gain amplifier and a narrow-band lock-in amplifier, after that, the output will be digitized by a high-resolution ADC and shows on the computer. By adjusting the amplitude and the phase by the computer, the output of the bridge can be nulled. Finally, the value of the unknown impedance can be obtained by the ratio of the two driving signals. The expressions of the impedance are:

$$R_X = \frac{V_1 R_{REF}}{V_2} \frac{1}{(cos\phi - \omega C_{REF} R_{REF} sin\phi)} \quad (1-9)$$

$$C_X = -\frac{V_2(sin\phi + \omega C_{REF} R_{REF} cos\phi)}{V_1 \omega R_{REF}} \quad (1-10)$$

Where V_1 is the input amplitude of the unknown impedance, V_2 is the input amplitude of the reference impedance, ω is the angular frequency of the input and ϕ is the phase difference between the two inputs. The specific formula derivation will be introduced in section 3.1.3.

The resolution of the used DDS in term of phase is very high (14bit) however the resolution in terms of amplitude is only 10-bit. So, the output signal of the bridge cannot be nulled completely, which means there will be a signal remaining. This remaining output signal should be interpolated to achieve the required very high resolution, which can be expressed as:

$$C_{REF} \times \left(\frac{output}{step\ value}\right) \times \frac{1}{1024} \quad (1-11)$$

where step value is change of 1bit value of the DDS at 10MHz.

In addition, in order to eliminate the influence of parasitic resistance further, an increase of the signal frequency is needed (10MHz or more).

The approach also aims for circuit solutions that can be integrated into a single CMOS chip. The prototype described in this thesis uses many commercially available components.

In the prototype, the aim of a system that can measure the impedance of capacitive sensor devices in the range of 22pF to 47pF with a 24ppm resolution @10Hz comparable with the best impedance bridges (the AH2700 from Andeen Hagerlin with the range of -0.165μF to 1.65μF and 0.16ppm resolution @1000Hz [54]) available now. The system described here should operate over a wider range of frequencies (up to 10MHz or more) to meet different types of capacitive sensors and aims to measure small capacitances of sensors typically in the pF range.

The ultimate goal would be to integrate the approach into a dedicated CMOS chip with a standard bus interface as a commercial product.

1.4 Outline of the Thesis

Chapter 1 Introduction

In this chapter, the need and basic idea of impedance measurement are provided at first, three basic detection ways of capacitive sensors are described, then the approach (self-balance bridge based on LIA measurement) of this thesis is presented.

Chapter 2 Lock-in amplifier (LIA)

The principle of the lock-in amplifier is shown in this chapter. Also, comparisons of analog and digital LIAs are introduced.

Chapter 3 Hardware design

Each part of the readout circuit is described in detail. Moreover, simulations of some circuits are also presented. The main emphasis is the charge amplifier.

Chapter 4 Software design

The programming methods of DDS, ADC and so on are given in this chapter.

Chapter 5 Measurement results

Performance test of the whole system is introduced in this chapter, and then these results will be compared with other works.

Chapter 6 Conclusion

Summarizes the contributions of the thesis. Furthermore, future performance improvements are highlighted.

Reference

[1]https://compliantmechanisms.byu.edu/content/introductionmicroelectromechanical-systems-mems
[2] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19710023921.pdf
[3] https://en.wikipedia.org/wiki/Comb_drive
[4] X. Li and G. C. M. Meijer, “A capacitive-sensor interface circuit based on a first-order charge-balanced SC-oscillator,” in Proc. IEEE 18th Instrumentation and Measurement Technology Conf., 2001, vol. 1, pp. 282–285.
[5] J. Shiah and S. Mirabbasi, "A 5-V 290 μV low-noise chopper-stabilized capacitive-sensor readout circuit in 0.8um CMOS using a correlated-level shifting technique," IEEE TCAS II, vol. 61, no. 4, pp. 254-258, 2014.
[6] J. W. Gardner, V. K. Varadan, O. O. Awadelkin, Microsensors MEMS and Smart Devices, New York: Wiley, 2001, pp. 259-260
[7] http://www.kirj.ee/public/Engineering/2007/issue_4/eng-2007-4-17.pdf
[8] B.E.Boser, “Capacitive sensor interfaces”, Lecture note, Berkeley sensor and actuator center, University of California, Berkeley
[9] A. Heidary, “A low-cost universal integrated interface for capacitive sensors,” Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 2011.
[10] I. Yokoshima, “RF impedance measurements by voltage-current detection,” IEEE Trans. Instrum. Meas., vol. 42, no. 2, pp. 524–527, Apr. 1993.
[11] V. Dumbrava, L. Svilainis, "The automated complex impedance measurement system", Electronics and Electrical Engineering, vol. 76, pp. 59-62, 2007.
[12] Impedance Measurement Handbook. Agilent Technologies Co. Ltd. USA. 2003.
[13] Da Silva, M.J. Impedance Sensors for Fast Multiphase Flow Measurement and Imaging. Ph.D. Thesis, Technische Universität Dresden: Dresden, Germany, 08 November 2008.
[14] J. Shiah, H. Rashtian, and S. Mirabbasi, "A low-noise high-sensitivity readout circuit for MEMS capacitive sensors," in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), pp.3280-3283, 2010.
[15] X. Li and G. C. M. Meijer, “An accurate interface for capacitive sensors,” IEEE Trans. Instrum. Meas., vol. 51, no. 5, pp. 935–939, Oct. 2002.
[16] B. George and V. J. Kumar, “Switched capacitor signal conditioning for differential capacitive sensors,” IEEE Trans. Instr. and Meas., vol. 56, No. 3, pp. 913-917, 2007.
[17] U. Schoneberg, H.G. Dura, B.J. Hosticka, W. Mokwa, Low- drift gas sensor with on-chip instrumentation, Proceedings of the 1991 International Conference on Solid-State Sensors and Actuators, San Francisco, CA, 1991, pp. 1006-1007.
[18] S.J. Sherman, et.al., “Low cost monolithic accelerometer”, Dig. VLSI Circuits Symp., June 1992, pp. 34-35.
[19] K. Chau, S.R. Lewis, Y, Zhao, R. T. Howe, S.F. Bart, and R.G. Marcheselli, “An integrated force-balanced capacitive accelerometer for low-g applications,” 1995 IEEE Conf. on Solid-State Sensors & Actuators, June 1995, pp. 593-596.
[20] J. Wu, G.K. Fedder, L.R. Carley, “A low-noise low-offset capacitive sensing amplifier for a 50-Pg/Hz monolithic CMOS MEMS accelerometer”, IEEE J. of Solid-State Circuits, vol. 39, May 2004, pp. 722-730.

[21] W. Yun, R.T. Howe, and P.R. Gray, “Surface micromachined digitally force-balanced accelerometer with integrated CMOS detection circuitry,” Solid-State Sensor and Actuator Workshop, Hilton-Head Island, SC, USA, June 1992, pp. 126-131.
[22] J. A. Geen, S. J. Sherman, J. F. Chang, S. R. Lewis, “Single chip surface micromachined integrated gyroscope with 50°/h Allan deviation,” IEEE J. of Solid-State Circuits, vol. 37, Dec. 2002, pp.1860-1866.
[23] M. J. Da Silva, Impedance Sensors for Fast Multiphase Flow Measurement and Imaging. Dresden, Germany: TUD Press, 2008.
[24] J-K Woo, C. Boyd, J. Cho, and K. Najafi, “Ultra-Low Noise Transimpedance Amplifier for High Performance MEMS Resonant Gyroscopes,” under review, Transducers 2017, June 2017.
[25] G. Royo, C. Sánchez-Azqueta, C. Gimeno, C. Aldea, S. Celma, “Programmable low-power low-noise capacitance to voltage converter for MEMS accelerometers”, Sensors, vol. 17, no. 1, pp. 67, 2017.
[26] https://en.wikipedia.org/wiki/Switched_capacitor
[27] M. Lobur and A. Holovatyy, “Overview and analysis of readout circuits for capacitive sensing in MEMS gyroscopes (MEMS angular velocity sensors),” in Proceedings of the 5th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH ’09), pp. 161–163, 2009.
[28] N. Yazdi, H. Kulah, K. Najafi, "Precision readout circuits for capacitive microaccelerometers", Proc. IEEE Sensors 2004, pp. 24-27.
[29] N. Yazdi, K. Najafi, “An interface IC for a capacitive silicon µg accelerometer,” 1999 IEEE Int. Solid-State Circuits Conf., Feb. 1999, pp. 132-133.
[30] N. Wongkomet, B. E. Boser, “Correlated double sampling in capacitive position sensing circuits for micromachined applications,” 1998 IEEE Asia-Pacific Conf. on Circuits and Systems, Nov. 1998, pp.723-726.
[31] M. Lemkin, B. Boser, “A three-axis micromachined accelerometer with a CMOS position-sense interface and digital offset-trim electronics,” IEEE J. of Solid-State Circuit, vol. 34, April 1999, pp. 456-468.
[32] H. Kulah, J. Chae, N. Yazdi, K. Najafi, “A multi-step electromechanical sigma-delta converter for micro-g capacitive accelerometers,” 2003 IEEE Int. Solid-State Circuits Conference, Feb. 2003, pp. 202-203.
[33] P. Maya-Hernandez, M. Sanz-Pascual and B. Calvo, "CMOS Low-Power Lock-In Amplifiers with Signal Rectification in Current Domain", IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 7, pp. 1858-1867, 2015.
[34] J. Wu, G. K. Fedder, and L. R. Carley, “A low-noise low-offset capacitive sensing amplifier for a 50µg/√Hz monolithic CMOS MEMS accelerometer,” IEEE J. Solid-State Circuits, vol. 39, no. 5, pp. 722–730, May 2004.
[35] V. Petkov and B. Boser, “A Fourth-order ΣΔ Interface for Micromachined Inertial Sensors,” in IEEE J. SolidState Circuits, vol. 40, no. 8, pp. 1602–1609, Dec. 2005.
[36] J. A. Geen, S. J. Sherman, J. F. Chang, and S. R. Lewis, “Single-Chip Surface Micromachined Integrated Gyroscope With 50˚/h Allan Deviation,” in IEEE J. Solid-State Circuits, vol. 37, no. 12, pp. 1860–1866, Dec. 2002.

[37] V. Kaajakari, Practical MEMS, Small Gear Publishing, 2009.
[38] G. Royo, M. Garcia-Bosque, C. Sánchez-Azqueta, C. Aldea, S. Celma, C. Gimeno, “Transimpedance amplifier with programmable gain and bandwidth for capacitive MEMS accelerometers”, I2MTC, 2017.
[39] A. De Marcellis, G. Ferri, and P. Mantenuto, “A novel 6-decades fullyanalog uncalibrated Wheatstone bridge-based resistive sensor interface,” Sens. Actuators B, Chem., vol. 189, pp. 130–140, Dec. 2013.
[40] A. De Marcellis, G. Ferri, and P. Mantenuto, “Analog Wheatstone bridge-based automatic interface for grounded and floating wide-range resistive sensors,” Sens. Actuators B, Chem., vol. 187, pp. 371–378, Oct. 2013.
[41] P. Mantenuto, G. Ferri, A. De Marcellis, "Uncalibrated automatic bridge-based CMOS integrated interfaces for wide-range resistive sensors portable applications", Microelectron. J., vol. 45, no. 6, pp. 589-596, 2014.
[42] P. Mantenuto, A. De Marcellis, G. Ferri, "Novel modified De-Sauty autobalancing bridge-based analog interfaces for wide-range capacitive sensor applications", IEEE Sensors J., vol. 14, no. 5, pp. 1664-1672, May 2014.
[43] G. Barile, G. Ferri, F. R. Parente, V. Stornelli, A. Depari, A. Flammini, and E. Sisinni, “Linear Integrated Interface for Automatic Differential Capacitive Sensing,” Proceedings, vol. 1, no. 4, p. 592, Aug. 2017.
[44] P. Holmberg, "Automatic balancing of linear AC bridge circuits for capacitive sensor elements", IEEE Trans. Instrum. Meas., vol. 44, no. 3, pp. 803-805, Jun. 1995.
[45] B. Hague, Alternating Current Bridge Methods. New York: Pitman, 1971.
[46]http://www.infoa2z.com/UploadFiles/Downloads/Solved/lab%20manuals%20PHY-101E/de-sautys-bridge1.pdf
[47] https://electronicsproject.org/de-sauty-bridge/
[48] C. Falconi, E. Martinelli, C. Di Natale, A. D’Amico, F. Maloberti, P. Malcovati, et al., “Electronic interfaces,” Sens. Actuators B, Chem., vol. 121, no. 1, pp. 295–329, 2007.
[49] P. Mantenuto, A. De Marcellis, and G. Ferri, “Uncalibrated analog bridge-based interface for wide-range resistive sensor estimation,” IEEE Sensors J., vol. 12, no. 5, pp. 1413–1414, May 2012.
[50] B. Hu, J. Wang, G. Song and F. Zhang, "A compact wideband precision impedance measurement system based on digital auto-balancing bridge", Measurement Science and Technology, vol. 27, no. 5, p. 055902, 2016.
[51]https://literature.cdn.keysight.com/litweb/pdf/59683809E.pdf?id=1000072161:epsg:dow
[52] https://literature.cdn.keysight.com/litweb/pdf/5991-3893EN.pdf?id=2466922
[53] https://www.ameteksi.com/products/frequency-response-analyzers
[54] http://www.andeen-hagerling.com/ah2700a.htm

Chapter 2
Lock-in Amplifier (LIA): Background & Theoretical analysis

In this chapter, the background and theoretical analysis of the lock-in amplifier are presented in four parts, starting with the principle of correlation detection, which includes autocorrelation and cross-correlation; followed by the principle of LIA, including basic concepts and compositions of LIA, among these compositions, the phase-sensitive detector is highlighted. Then, some characteristics between analog LIA and digital LIA is discussed, basic requirement of the circuit has been determined as well. Based on all these, the conclusion is given in the end. This chapter gives the overall information for the circuit-level design.

2.1 Principle of correlation detection

In the weak signal detection and extraction technology, two types of signals are involved, one is a useful signal (signal of interest) and the other is noise. The former has a certain law; it can be repeated and expressed as a time-related deterministic function. However, the latter does not show a certain pattern, because the noise at different times is not related. There are two ways of detection: the first one is called autocorrelation, it uses the characteristic of the signal itself to find the signal, which means it is the correlation of a signal with a delayed copy of itself as a function of delay [1]. The other way is called cross-correlation, it uses the relationship of correlation between two signals to eliminate the effect of noise and improve signal to noise ratio [2]. Correlation detection is a way that utilizes correlation theory to measure signal, mainly by suppressing noise and maximizing bandwidth limitation [3].

2.1.1 Autocorrelation method

Model of autocorrelation shows in Figure 2-1.

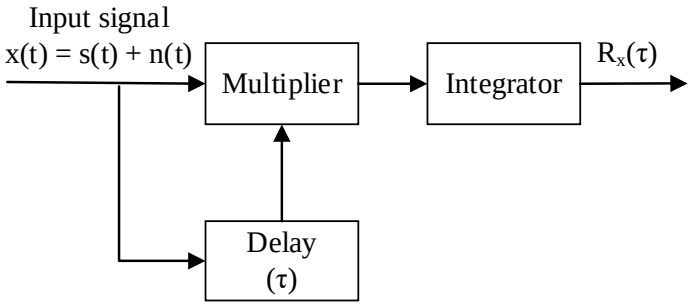


Figure 2-1 Schematic of autocorrelation

The useful input signal can be expressed as:

$$s(t) = Asin(\omega_0 t + \varphi) \quad (2-1)$$

Where A is amplitude, ω_0 is angular frequency, and φ is the initial phase.

So total input signal is:

$$x(t) = s(t) + n(t) = Asin(\omega_0 t + \varphi) + n(t) \quad (2-2)$$

The two channels of the correlation receiver are receiving input signals simultaneously; the delay plays a role in delaying the input signal for a period of time τ. The multiplier receives the signals from two paths respectively, after the operation of multiplier and integrator, in turn, it is easy to get output $R_x(\tau)$. In case of changing the delay time separately, the corresponding output signals are obtained one by one, so the relation curve between correlation function and the delay time can be found, which reflects the degree of correlation of information function values for different time τ, then the autocorrelation output is:

$$R_{xx}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} x(t)x(t-\tau)dt$$

$$= R_{ss}(\tau) + R_{sn}(\tau) + R_{ns}(\tau) + R_{nn}(\tau) \quad (2-3)$$

In which

$$R_{ss}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} A^2 sin(\omega_0 t + \varphi)sin(\omega_0 t - \omega_0\tau + \varphi)dt \quad (2-4)$$

$$R_{sn}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} AN(t-\tau)sin(\omega_0 t + \varphi)dt \quad (2-5)$$

$$R_{ns}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} AN(t)sin(\omega_0 t - \omega_0\tau + \varphi)dt \quad (2-6)$$

$$R_{nn}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} N(t)N(t-\tau)dt \quad (2-7)$$

The noise is a random quantity and cannot be expressed as a time-dependent deterministic function, which means s(t) and n(t) are uncorrelated. Therefore, their mean value is zero, and $R_{sn}(\tau) = 0$ and $R_{ns}(\tau) = 0$ is obtained. As τ increases, $R_{nn}(\tau) \to 0$, when τ increases sufficiently large, $R_{xx}(\tau) = R_{ss}(\tau)$ is obtained. The useful signal s(t) can be obtained from the autocorrelation function $R_{xx}(\tau)$, which carries

some information about the useful signal s(t)[4].

2.1.2 Cross-correlation method

Cross-correlation function reflects the correlation between two different signals, the realization of the schematic shown in Figure 2-2.

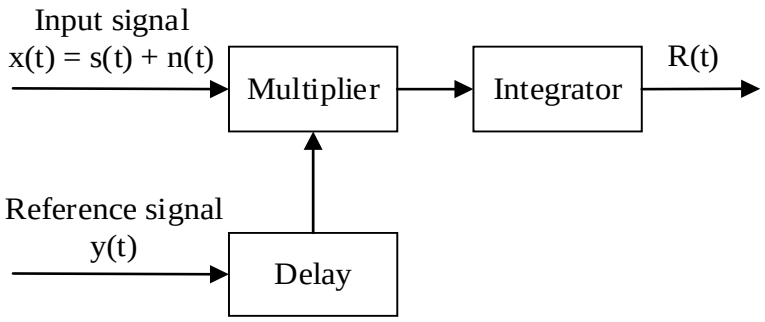


Figure 2-2 Schematic of cross-correlation

Set the total input signal as:

$$x(t) = s(t) + n(t) = \ Asin(\omega_0 t \ + \ \varphi) + n(t) \qquad (2-8)$$

Set reference signal as:

$$y(t) = Bsin(\omega_1 t \ + \ \varphi_0) \qquad (2-9)$$

Where B is amplitude, ω_1 is angular frequency and φ_0 is the initial phase.

When the two signals pass through the multiplier, the result is:

$$R(t) = \int_{t_0}^{t_1} x(t) \cdot y(t)dt = \int_{t_0}^{t_1} [(Asin(\omega_0 + \ \varphi) + N(t)) \cdot Bsin(\omega_1 t + \ \varphi_0)dt]$$

$$= \ AB/2 \int_{t_0}^{t_1} \{cos[(\omega_0 - \omega_1)t + \varphi - \varphi_0] - cos[(\omega_0 + \omega_1)t + \varphi + \varphi_0]\}dt$$

$$+ \int_{t_0}^{t_1} BN(t) \sin(\omega_1 t + \ \varphi_0) \, dt$$

$$= R_{sy}(\omega_0, \omega_1, \varphi, \varphi_0) + \ R_{ny}(\omega_0, \omega_1, \varphi, \varphi_0) \qquad (2-10)$$

Which

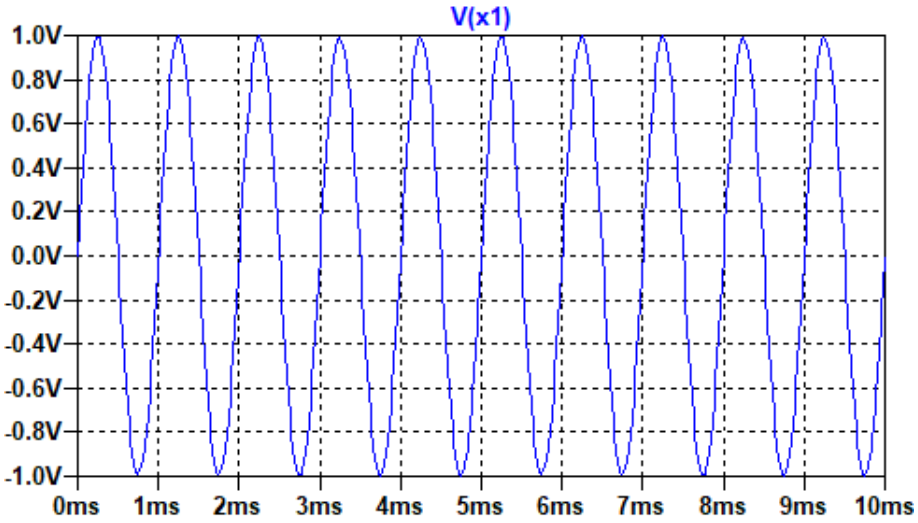
$R_{sy}(\omega_0, \omega_1, \varphi, \varphi_0)$

$$= AB/2 \int_{t_0}^{t_1} \{cos[(\omega_0 - \omega_1)t + \varphi - \varphi_0] - cos[(\omega_0 + \omega_1)t + \varphi + \varphi_0]\}dt \qquad (2-11)$$

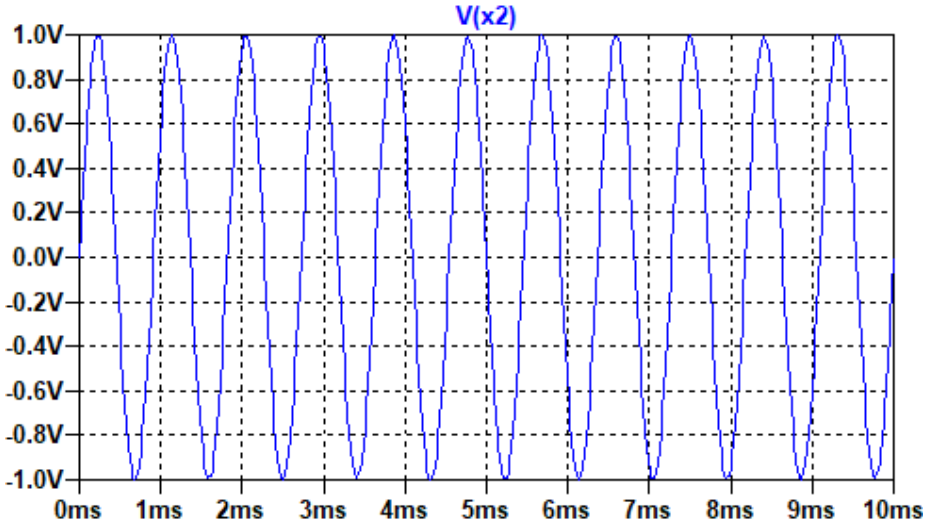
is the cross-correlation output of the useful signal (signal of interest) and the reference signal, and

$$R_{ny}(\omega_0, \omega_1, \varphi, \varphi_0) = \int_{t_0}^{t_1} BN(t) \sin(\omega_1 t + \ \varphi_0) \, dt \qquad (2-12)$$

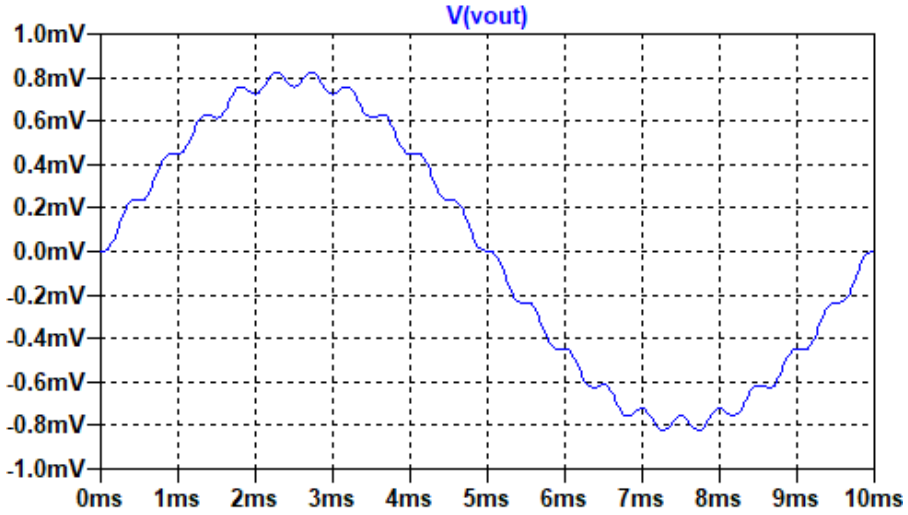
is the cross-correlation output of the noise and the reference signal.



(a) Input signal is 1KHz



(b) Input signal is 1.1KHz



(c) Output of integrator

Figure 2-3 The integral output of two signals

When $\omega_0 \neq \omega_1$, since the two components of R_{xy} are periodic functions and mean value is 0, the output signal of the correlator is 0 after the integrator and integration time are the common periods of the two signals, as shown in Figure 2-3. This means that the frequency ω_0 of the reference signal must be equal to the frequency ω_1 of the useful signal. The reference signal can be expressed as:

$$y(t) = Bsin(\omega_0 t + \varphi_0) \qquad (2-13)$$

From section 2.1.1, because the noise and the reference signal are uncorrelated, the equation 2-12 is equal to 0. Then, combining the equation 2-10 with the equation 2-13, and assuming that the integral time constant of the integrator is T, and the integration time t = T, the final result of cross-correlation is:

$$R(t) = \frac{1}{T}\int_0^T K_v \frac{AB}{2}\{cos[(\omega_0 - \omega_0)t + \varphi - \varphi_0] - cos[(\omega_0 + \omega_0)t + \varphi + \varphi_0]\}dt$$
$$= \frac{K_v AB}{2}\cos(\varphi - \varphi_0) \qquad (2-14)$$

where K_v is integral gain.

According to the equation 2-14, the output of the cross-correlation is a DC signal, and the value of this DC signal is related to the two input signals and phase difference.

Comparing equations 2-3 and 2-10, it is shown that the cross-correlation detection uses an external reference signal and can effectively avoid the noise impact of the signal under test in the self-correlation detection. Therefore, the signal to noise ratio of the cross-correlation detection system is higher than the self-correlation detection system. In addition, the cross-correlation function can also reflect the phase difference between the two signals, if the phase of one of the signals is known, then the phase of the other signal can be determined [5]. Thus, the cross-correlation algorithm is better than the autocorrelation algorithm. In this paper, the design of lock-in amplifier is based on the cross-correlation method.

2.2 Basic principle of a lock-in amplifier (LIA)

A lock-in amplifier is a device that can be used for weak signal detection by using the principle of cross-correlation, which is a phase-sensitive signal detection amplifier [6-8].

During the measurement, noise is a disturbing signal. Among these noises, white noise and 1/f noise provide more significant influence on instrument equipment [9,10]. The presence of noise in the system can adversely affect the useful signal and the useful signal is often hidden by it. In order to reduce the influence, a narrow-band filter is usually used to increase SNR of the signal. However, the filter also has its limitations: The Q value (The ratio of the center frequency to the passband width) is limited by the hardware part of the filter, which affects its ability to extract useful signals and filter noise at a higher level [11-14].

As shown in Figure 2-4, the elemental composition of the lock-in amplifier is introduced, which mainly includes: signal channel, reference channel, phase sensitive detector (PSD) and lowpass filter (LPF) [15-17].

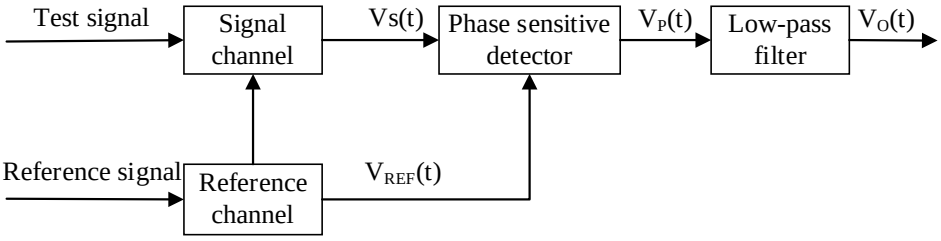


Figure 2-4 Structure diagram of the lock-in amplifier

2.2.1 Signal channel

The signal channel mainly performs the basic processing on the corresponding test signal (signal of interest). For example, the signal is amplified properly so that the test signal can reach the corresponding working level before entering the phase sensitive detector or use the band-pass filter to filter the signal in order to eliminate the influence of higher harmonics on the measurement results [18]. The first part of the signal channel is generally a preamplifier, the main consideration is the input signal is really small, it could be mV, μV or even more weak, then the input of the primary task is to amplify the input signal, which requires a high input impedance R_i, low output impedance R_o and higher voltage gain.

2.2.2 Reference channel

The reference signal generally uses the same frequency as the test signal with a sinusoidal signal or a square wave signal to achieve the selection of the signal under test. Reference channel (it consists of the trigger circuit, the frequency conversion circuit and the phase shift circuit [19,20].) can be used to adjust DC of the reference signal, amplify or attenuate the reference signal to meet the input requirements. In addition, the reference channel can also adjust the phase of the reference signal according to different system requirements so as to achieve the best detection result.

2.2.3 Phase-sensitive detector (PSD)

Phase-sensitive detector (PSD) is the most critical and essential part of the lock-in amplifier. It takes the reference signal as a standard and picks out the signal components of the reference signal with the same frequency from the input signal [8]. After the filtering process, the output contains the amplitude and phase information of the effective component, which can calculate the amplitude and phase of the valid part of the test signal.

2.3 The principle of phase sensitive detector (PSD)

The phase-sensitive detector is a vital part of the lock-in amplifier [19], usually using a multiplier as a phase-sensitive detector, which multiplies the test signal V_s with a reference signal V_{ref}. Therefore, in a sense, the phase-sensitive detector here is equivalent to a multiplier, which is also equivalent to a modulator [21,22].

Taking a sine wave as an example, let the test signal V_s and the reference signal V_{ref} be:

$$V_s = A_s \cos(\omega_n t) \qquad (2-15)$$

$$V_{ref} = A_r \cos(\omega_0 t + \theta) \qquad (2-16)$$

respectively.

Where A_s and A_r are the amplitudes of the test signal and the reference signal respectively, ω_n and ω_0 are the angular frequencies respectively, and θ is the phase difference between them. So, the output of the multiplier can be expressed as:

$$V_P = V_s V_{ref} = A_s \cos(\omega_n t)\, A_{ref} \cos(\omega_0 t + \theta)$$
$$= 0.5 A_s A_r \cos[(\omega_n - \omega_0)t + \theta] + 0.5 A_s A_r \cos[(\omega_n + \omega_0)t + \theta] \quad (2-17)$$

This shows that the output results appear the difference frequency component and sum frequency component. From section 2.1.2, ω_n and ω_0 should be equal, so equation 2-17 becomes:

$$V_P = 0.5 A_s A_r \cos\theta + 0.5 A_s A_r \cos(2\omega_0 t + \theta) \quad (2-18)$$

As shown in Figure 2-5, the frequency spectrum of the output signal is shifted to $\omega = 0$ and $\omega = 2\omega_0$. The shape does not change after the spectrum is moved, and the signal amplitude at this time depends on the product of A_s and A_r.

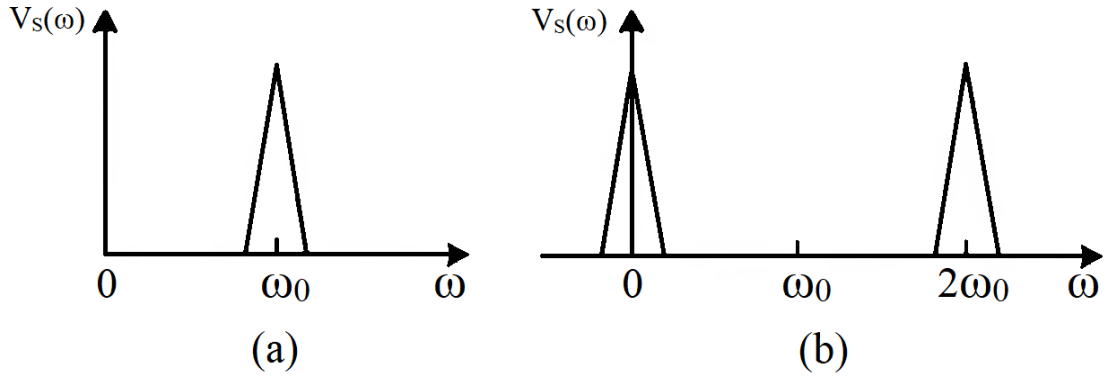


Figure 2-5 Schematic of spectrum shift

If the phase sensitive detection circuit connected to the low-pass filter, the sum frequency component of the high-frequency components will be filtered out. At this time, the circuit output is:

$$V_O = 0.5 A_s A_r \cos(\theta) \quad (2-19)$$

In practical applications, the reference signal also uses a square wave. Assuming that the test signal is a sine wave, as shown in formula 2-15 and the reference signal is a square wave, which can be expressed as:

$$V_{ref} = \frac{4}{\pi} A_r \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} \cos[(2n-1)\omega_0 t + \theta] \quad (2-20)$$

At this point, after the phase-sensitive detector, the output is:

$$V_P = V_s \times V_{ref} = A_s \cos(\omega_n t) \times \frac{4}{\pi} A_r \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} \cos[(2n-1)\omega_0 t + \theta]$$
$$= \frac{2A_s A_r}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} \cos[(\omega_n + (2n-1)\omega_0)t + \theta]$$
$$+ \frac{2A_s A_r}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} \cos[(\omega_n - (2n-1)\omega_0)t + \theta] \quad (2-21)$$

It can be seen that the sum frequency component and the difference frequency component also appear. After the low-pass filter, the sum frequency component can be filtered out, the difference frequency component will show in the output of the filter, and the output is:

$$V_O' = \frac{2A_sA_r}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{2n-1}\cos[(\omega_n-(2n-1)\omega_0)t+\theta] \qquad (2-22)$$

When ω_n = (2n-1)ω_0(n=1,2,3……), the DC component will appear in the above result, and their amplitudes will decrease by a factor of 1/(2n-1). This phenomenon is called the harmonic response of the phase-sensitive detector [20]. The schematic diagram is shown in Figure 2-6.

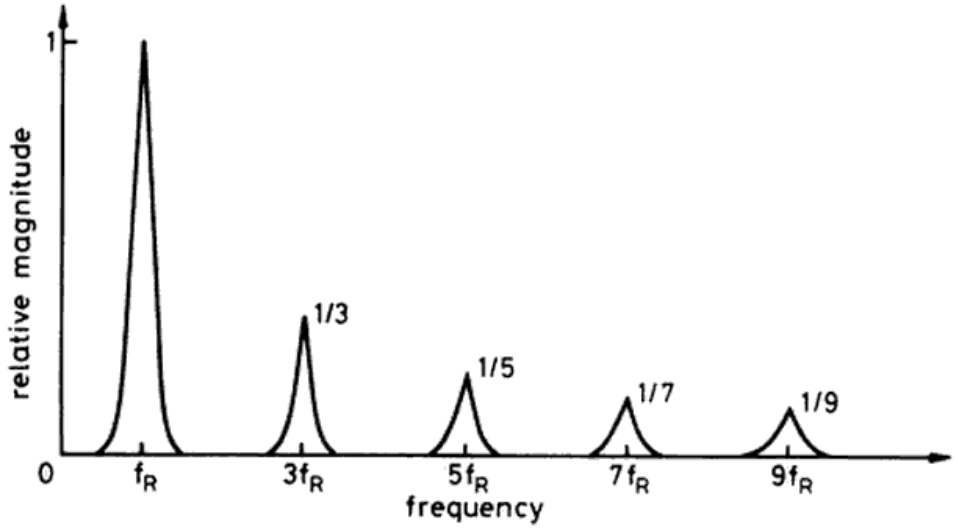


Figure 2-6 The first five harmonic transmission windows of PSD [20]

The transmission windows are centered on the odd harmonics of the reference frequency and the maximum magnitude of each window is weighted by the magnitude of its associated reference Fourier components [20]. The signal must be in one of the transmission windows before the phase sensitive detector outputs a response. A test signal must be coherent with One or more reference Fourier component so that a 'true' d.c. response can be obtained [20].

However, the most common way to eliminate the effects of harmonics is to add a band-pass filter in the signal path whose center frequency is ω_0, so that it filters out the higher harmonics first, leaving only the first harmonic information. Then, the output is:

$$V_O = \frac{2A_sA_r}{\pi}cos\theta \qquad (2-23)$$

It can be seen that no matter whether the reference signal is a sine wave or a square wave, the output result after the phase-sensitive detector and the filter are not only related to the signal amplitude but also to the phase information between the two signals,

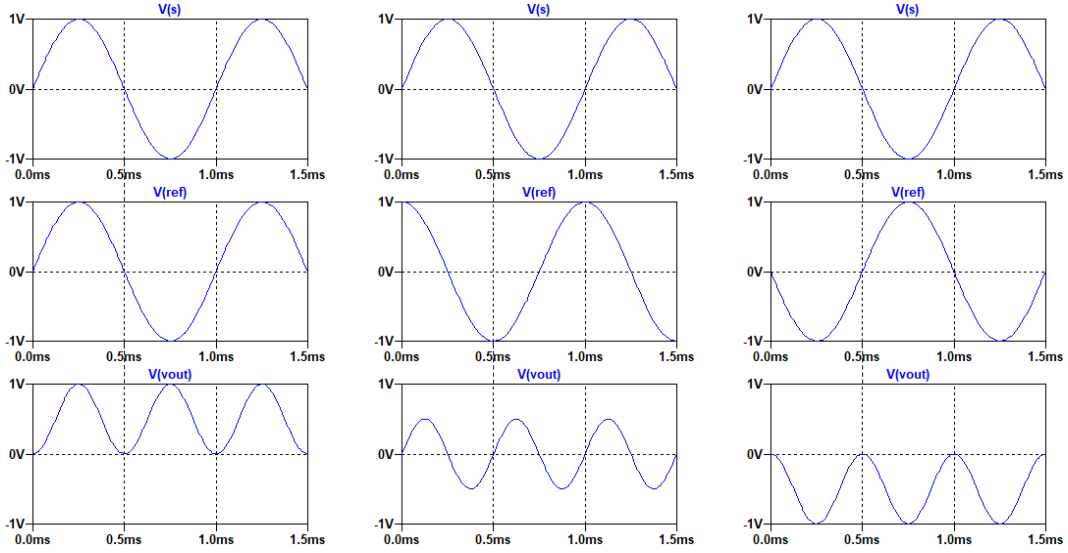
Therefore, the phase-sensitive detector can measure amplitude and phase simultaneously.

Sine waves are useful for systems that require high frequency (above 1MHz) carriers, for instance, systems that must measure very low capacitance. Sine waves are also preferred for high-precision circuits. Compared with the square wave excitation, the slew rate of the amplifier is reduced by a factor of 10, and low-frequency amplifiers can be used. Sine waves are essential for high-gain bridge circuits, as a good null is more easily achieved without the presence of harmonic energy from the square wave excitation [23].

Although square waves are easy to integrate on a single-chip system and are suitable for low-gain systems such as motion detectors with wide linear operation, it must be careful to avoid the effects of

instability and non-linearity which are produced by the amplifier when it reaches its limit slew rate for circuit using square wave modulation, and in order to obtain a good wave shape, amplifier bandwidth must be a factor of 10 higher than for sine wave circuits [23]. In this paper, Due to the relatively stable performance of the sine wave, the sine wave excitation is used as the reference signal in the lock-in amplifier system.

In order to show the output more intuitively, Figure 2-7 shows the waveforms of input and the output of the phase-sensitive detector under different phase shifts, where $\omega_n = \omega_0 = 1KHz$, and $A_s = A_r = 1V$ in the equations 2-15 and 2-16 respectively. As shown in Figure 2-7(a), when $\theta = 0°$, the output waveforms of the phase-sensitive detector are above 0V, and the maximum DC value can be obtained through the integrator or low-pass filter; when $\theta = 90°$, as shown in Figure 2-7(b), after the phase-sensitive detector, the output waveforms are evenly split between the positive part and negative part, then the DC component is 0 after filtering; when $\theta = 180°$, the output waveform is shown in Figure 2-7(c), the principle is the same as (a), but the output is opposite to (a) and below 0V, at this moment, the DC value is the negative maximum after filtering.



(a) $\theta = 0°$ (b) $\theta = 90°$ (c) $\theta = 180°$

Figure 2-7 Input and output waveform diagram of the phase-sensitive detector with different phase

2.4 Analog LIA and Digital LIA

Depending on the dynamic reserve of the instrument, signals up to 1 million times smaller than noise components and translates to a signal-to-noise ratio as low as -120dB, potentially fairly close by in frequency, can still be sure detected [24]. It can improve the Q factor by a million times. All these show that the system design can have many advantages by using the lock-in amplifier. With the ever-changing technology, the performance of the lock-in amplifier is even more powerful: the new dual-phase digital lock-in amplifier, multi-channel lock-in amplifier, precision lock-in amplifier and so on. They have a very high gain and gain accuracy. At present, the core device (PSD) of the commercial lock-in amplifier is being replaced by DSP (Digital Signal Processor), which further enhances its performance [25,26].

Currently, the commonly used lock-in amplifier on the market has two types: analog and digital. By comparing them, it finds that the former has characteristics such as early start, fast speed, but poor

parameter stability and flexibility, the latter is developed with the development of digital technology, and it uses high-speed ADC to sample signals at high speed, so this has high requirements for microprocessor [21,27-29].

2.4.1 Analog LIA

Analog LIA, as the name implies, which digitizes the signals only after the analog mixing stage before or after low-pass filter. There are some analog elements like voltage-controlled oscillators, low-noise amplifiers, mixers and simple RC filters for signal processing [30].

Figure 2-8 shows the functional block diagram of a typical analog lock-in amplifier, such as the PerkinElmer Instruments models 5109, 5110, 5209 and 5210. Dual-phase instruments include all of the sections shown whereas those sections within the dotted line are omitted in single phase units [29].

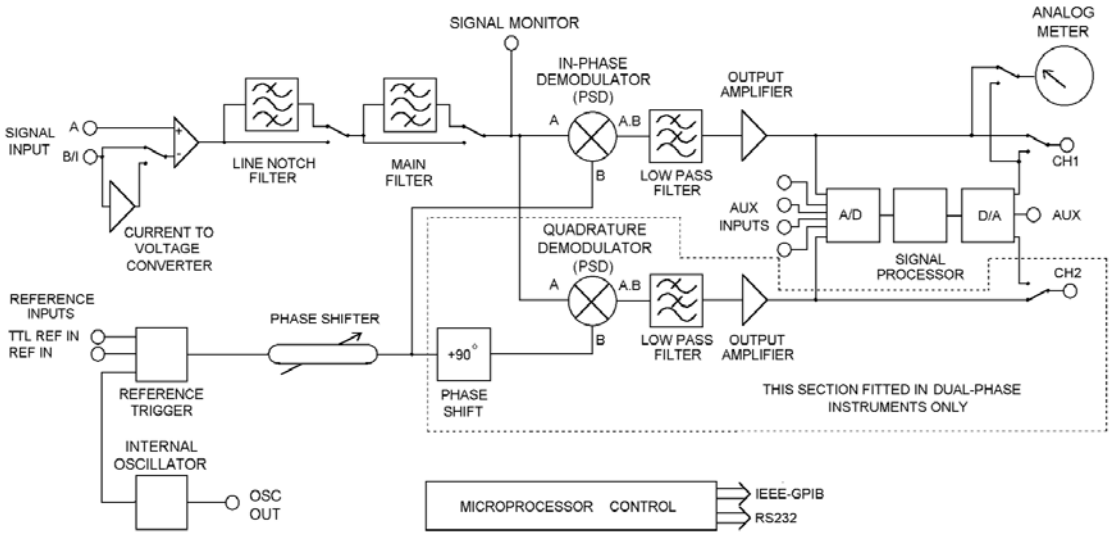


Figure 2-8 The functional block diagram of commercial analog LIA [29]

2.4.2 Digital LIA

For digital lock-in amplifier, the analog input signal is converted to the digital domain by an analog-to-digital converter (ADC) immediately, and all following steps are then carried out numerically by digital signal processing (DSP) [30].

Figure 2-9 shows the functional block diagram of a typical high-performance digital lock-in amplifier, such as UHFLI Instrument [31].

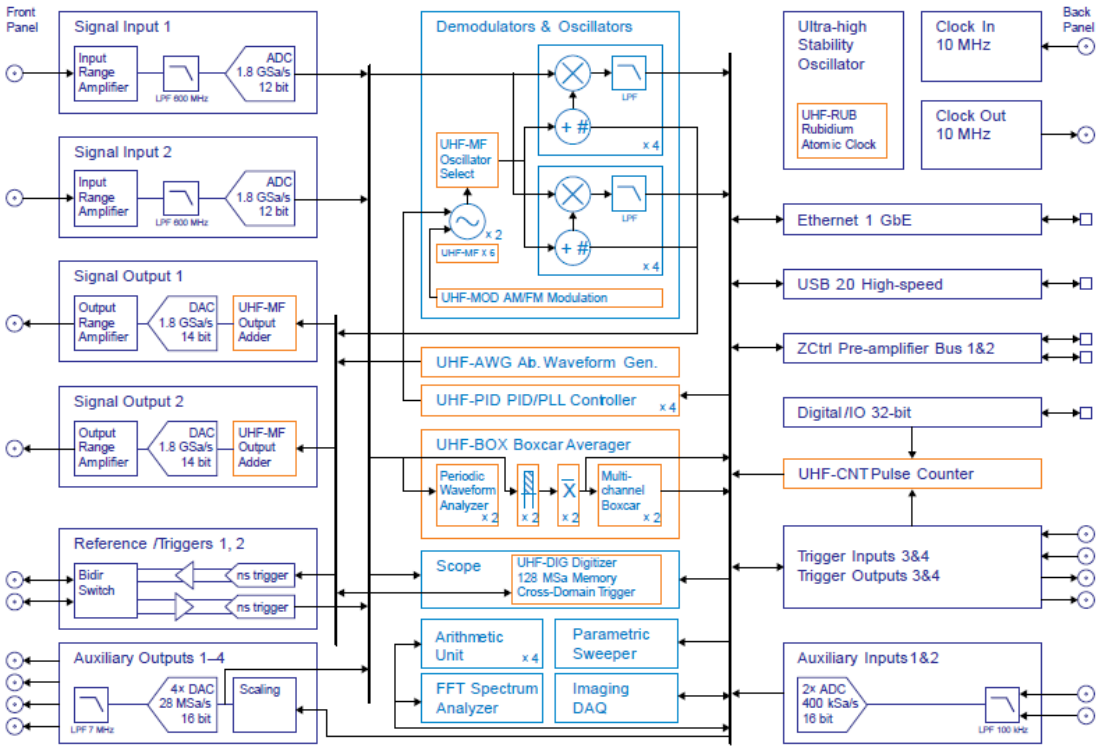


Figure 2-9 The functional block diagram of commercial digital LIA [31]

The speed, resolution and linearity of ADC and DAC are continuously improved, so the transformation of analog to digital is promoted. This development has helped to lift the frequency range, input noise and dynamic reserves to new limits. In addition, digital signal processing is less prone to errors due to s mismatches of signal paths, such as crosstalk and drift caused by temperature changes. Moreover, the biggest advantage of the digital approach may be that it can analyze signals in multiple ways simultaneously without losing SNR [31]. Compared with the commercial digital lock-in amplifier, the analog method shows in Figure 2-7 is outdated. The most of the lock-in amplifiers are implemented on DSP and microcontroller with DSP features at this moment.

However, these kinds of commercial lock-in amplifier show high costs and weights, having very complex architectures [21], which means they can only measure and analyze in the lab. There is also another limitation for digital LIA, in order to avoid DC offset and maintain a stable sampling rate, the maximum frequency of operation is limited to half the sampling rate.

For sensor applications, especially for capacitive gas sensors, a portable lock-in amplifier is needed, but most of them are designed for specific applications (e.g., motor and turbine fault control). In the literature, several implementations of analog LIA and digital LIA can be found. These LIAs have some characteristics, such as low power consumption, compact size, low price and lightweight but are mainly designed for low-frequency phase-sensitive detection with, typically, very high response times [18,32-35]. To overcome these limitations, a simple, portable, low-cost and analog LIA is proposed, which capable to measure amplitude variations of sinewave signals at frequency up to 10 MHz with response times of few milliseconds for fast and weak signal detection sensing applications, especially for the capacitive gas sensor.

2.5 Conclusion

This chapter makes an in-depth study of the Background & Theoretical analysis of the lock-in amplifier. The theoretical models and techniques of correlation detection are analyzed in depth, including autocorrelation and cross-correlation. The cross-correlation method provides the basic theory for the lock-in amplifier. By testing two sinewave signals with 1KHz and 1.1KHz and output by the integrator, it can be seen that the detection effect is best when the frequency of the reference signal is equal to the frequency of the useful signal. Then, the important part of the lock-in amplifier such as signal channel, reference channel and the phase-sensitive detector are described, and the core device (PSD) of the lock-in amplifier is emphatically introduced. After comparing the characteristics of analog lock-in amplifier and digital lock-in amplifier as well as their current status, the basic requirement of the lock-in amplifier for this design has been determined.

Reference

[1]https://en.wikipedia.org/wiki/Autocorrelation

[2]https://en.wikipedia.org/wiki/Cross-correlation

[3] Yang Hanxiang, “Research on Weak Signal Detection and Extraction”, Science and Technology Square, January, 2009, pp.27-28.

[4] R. M. Fano, “Signal-to-noise ratio in correlation detectors,” M.I.T. Research Laboratory of Electronics, Technical Report No. 186, February, 1951.

[5] G. de Graaf and R. F. Wolffenbuttel, “Lock-in amplifier techniques for low-frequency modulated sensor applications,” in Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC), May 2012, pp. 1745–1749.

[6] M.L. Meade Lock-in Amplifiers: Principles and Applications Peter Peregrinus Ltd. 1983.

[7] Lock-in amplifiers and pre-amplifiers Princeton Appl. Res. Corp. data sheets 1971.

[8] Lock-in amplifiers appl. notes Stanford Res. Sys. data sheets 1999.

[9] https://en.wikipedia.org/wiki/Noise_(electronics)

[10] http://www.engr.usask.ca/classes/EE/323/notes_2005/chapter8.pdf

[11] https://en.wikipedia.org/wiki/Q_factor

[12] S. Ghaffari et al., “Quantum limit of quality factor in silicon micro and nano mechanical resonators,” Sci. Reports, vol. 3, art. no. 3244, 2013.

[13] S. Pipilos, Y. P. Tsividis, J. Fenk, Y. Papananos, "A Si 1.8 GHz RLC filter with tunable center frequency and quality factor", *IEEE J. Solid-State Circuits*, vol. 31, pp. 1517-1525, Oct. 1996.

[14] F. Grine, T. Djerafi, M. Benhabiles, K. Wu and M. Riabi, "High-Q Substrate Integrated Waveguide Resonator Filter With Dielectric Loading", IEEE Access, vol. 5, pp. 12526-12532, 2017.

[15] I. J. Bhagyajyoti, L. S. Sudheer, P. Bhaskar, and C. S. Parvathi, “Review on Lock-in Amplifier,” Int. J. Sci. Eng. Technol. Res., vol. 1, no. 5, pp. 40–45, 2012.

[16] P. M. Maya-Hernández, L. C. Álvarez-Simón, M. T. Sanz-Pascual, and B. Calvo, “An Integrated Low-Power Lock-In Amplifier and Its Application to Gas Detection”, Sensors, vol. 14, no. 9, pp. 15880-15899, 2014.

[17] A. De Marcellis, E. Palange, N. Liberatore and S. Mengali, "Low-Cost Portable 1 MHz Lock-In Amplifier for Fast Measurements of Pulsed Signals in Sensing Applications", IEEE Sensors Letters, vol. 1, no. 4, pp. 1-4, 2017.

[18] Q. Wang, H. Zheng and M. Jiang, "Implementation of digital lock-in amplifier based on system generator", 2016 IEEE International Conference on Signal and Image Processing (ICSIP), 2016.

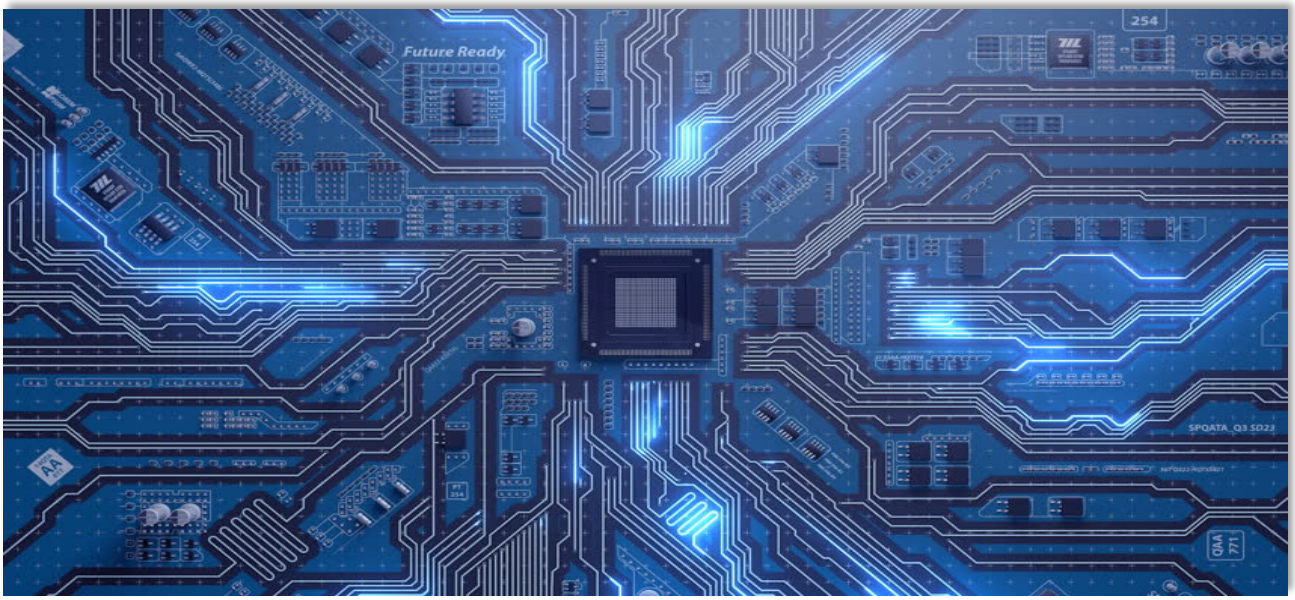
[19] J. Scofield, "Frequency-domain description of a lock-in amplifier", American Journal of Physics, vol. 62, no. 2, pp. 129-133, 1994.

[20] M. Meade, Lock-in amplifiers. London: Peregrinus, 1989.

[21] A. De Marcellis, G. Ferri, A. D’Amico, C. Di Natale, and E. Martinelli, “A fully-analog lock-in amplifier with automatic phase alignment for accurate measurements of ppb gas concentrations,” IEEE Sensors J., vol. 12, no. 5, pp. 1377–1383, May 2012.

[22] X. Chen, J. Chang, F. Wang, Z. Wang, W. Wei, Y. Liu and Z. Qin, "A portable analog lock-in amplifier for accurate phase measurement and application in high-precision optical oxygen concentration detection", Photonic Sensors, vol. 7, no. 1, pp. 27-36, 2016.

[23] L. Baxter, Capacitive sensors. New York: IEEE Press, 1997, pp. 58-59.
[24] https://en.wikipedia.org/wiki/Lock-in_amplifier
[25] U. Marschner, H. Grätz, B. Jettkant, D. Ruwisch, G. Woldt, W. Fischer and B. Clasbrummel, "Integration of a wireless lock-in measurement of hip prosthesis vibrations for loosening detection", Sensors and Actuators A: Physical, vol. 156, no. 1, pp. 145-154, 2009.
[26] M. Sonnaillon and F. Bonetto, "A low-cost, high-performance, digital signal processor-based lock-in amplifier capable of measuring multiple frequency sweeps simultaneously", Review of Scientific Instruments, vol. 76, no. 2, p. 024703, 2005.
[27] S. Bhattacharyya, R. Ahmed, B. Purkayastha and K. Bhattacharyya, "Implementation of Digital Lock-in Amplifier", Journal of Physics: Conference Series, vol. 759, p. 012096, 2016.
[28] L. Zhang and D. Zhang, Yun suan fang da qi ying yong ji shu shou ce. Beijing: Ren min you dian chu ban she, 2009.
[29] https://cpm.uncc.edu/sites/cpm.uncc.edu/files/media/tn1002.pdf
[30]https://www.zhinst.com/sites/default/files/li_primer/zi_whitepaper_principles_of_lock-in_detection.pdf
[31] https://www.zhinst.com/sites/default/files/ziUHF_UserManual_49900.pdf
[32] J. Ferreira and A. Petraglia, "Analog integrated lock-in amplifier for optical sensors", IEEE Instrumentation & Measurement Magazine, vol. 20, no. 2, pp. 43-50, 2017.
[33] A. De Marcellis, G. Ferri, and A. D. Amico, “One-decade frequency range, in-phase auto-aligned 1.8 V 2 mW fully-analog CMOS integrated lock-in amplifier for small/noisy signal detection,” IEEE Sensors J., vol. 16, no. 14, pp. 5690–5701, Jul. 2016.
[34] C. Azzolini, A. Magnanini, M. Tonelli, G. Chiorboli, and C. Morandi, “Integrated lock-in amplifier for contact-less interface to magnetically stimulated mechanical resonators,” in Proc. IEEE Int. Conf. Des. Technol. Integr. Syst. Nanoscale Era, 2008, pp. 1–6.
[35] P. M. Maya-Hernandez, M. T. Sanz-Pascual, and B. Calvo, “CMOS low-power lock-in amplifiers with signal rectification in current domain,” IEEE Trans. Instrum. Meas., vol. 64, no. 7, pp. 1858–1867, Jul. 2015.



Chapter 3
Hardware design and Simulation

In this section, circuit-level design and analysis of the interface circuit will be presented. The design of the front-end circuit is divided into three parts: DDS, pre-amplifier and charge amplifier. This is followed by the design of the second-order amplifier. After that, the mixer and lowpass filter are shown in section 3.3. Finally, a summary of this chapter will be provided.

3.1 Front-end design

The front-end can be divided into several sub-sections: DDS, Pre-amplifier, capacitive sensor and charge amplifier.

The whole structure of the front-end is shown in Figure 3-1.

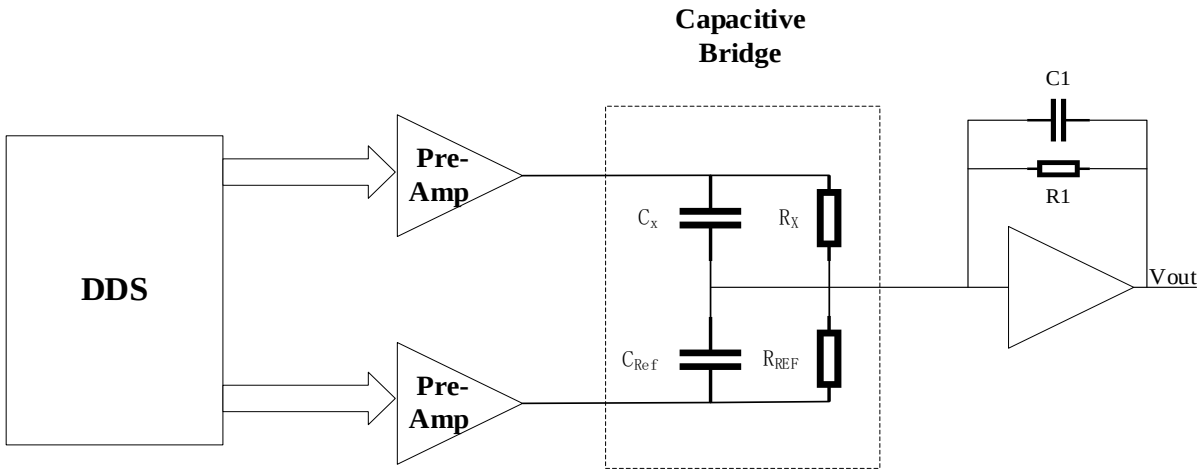


Figure 3-1 The structure of front-end

3.1.1 Direct digital synthesizer (DDS)

In order to drive the capacitive sensor and provide the reference signal for lock-in amplifier, a signal generator is needed. After considering the accuracy and cost of the entire system, DDS is selected. Direct Digital Synthesis (DDS) is a mixed/analog-signal processing technique that uses a fixed-frequency precision clock source as a reference to generate a frequency-tunable and phase-tunable output signal. In essence, by the scaling factor set forth in a programmable binary tuning word, the reference clock frequency is “divided down” in a DDS architecture. The tuning word is 24-48 bits long typically which allows a DDS implementation to provide a very high output frequency tuning resolution [1].

Simplest form of DDS is shown in Figure 3-2, it can be implemented from a frequency reference (often a crystal or SAW oscillator), a numerically controlled oscillator (NCO) and a digital-to-analog converter (DAC).

In this case, the reference oscillator provides a stable time base for the system and determines the frequency accuracy of the DDS. It provides the clock for the NCO, then, NCO generates a discrete-time, quantized version of the desired output waveform (mainly sinusoidal) at its output, and the digital word contained in the Frequency Control Register controls the period of the output waveform in the meantime. The DAC converts the sampled, digital waveform into an analog waveform. The output reconstruction filter rejects the spectral replicas produced by the zero-order hold inherent in the analog conversion progress [2].

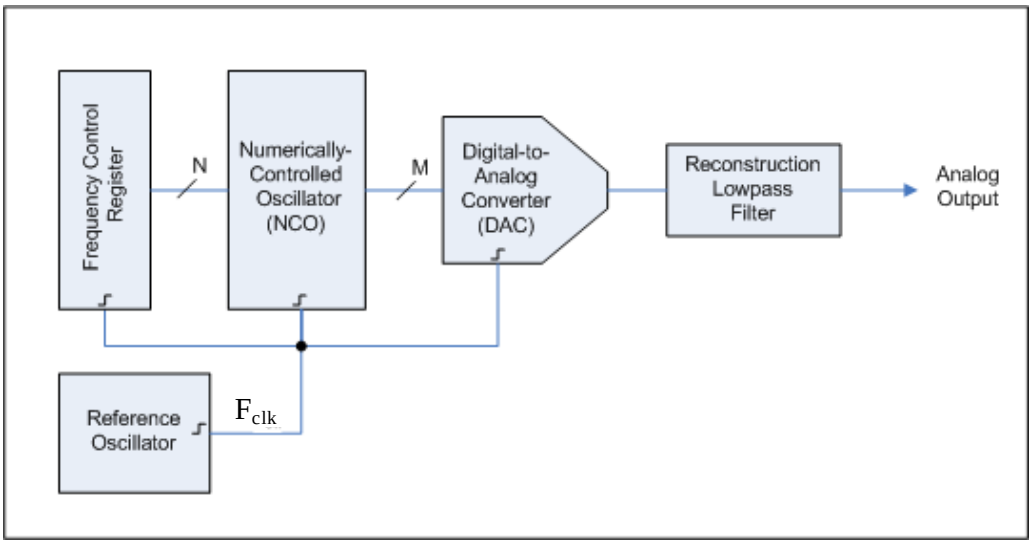


Figure 3-2 Simple block diagram of Direct Digital Synthesizer [2]

As a DDS, AD9959 is a good choice, since it consists of four independent channels and provides independent frequency, phase, and amplitude control on each channel, which is very suitable for this design. Some other characteristics of the AD9959: 0.12 Hz frequency tuning resolution, 14-bit phase offset resolution, 10-bit output amplitude scaling resolution, 200MHz bandwidth.

The serial I/O port of AD9959 provides an SPI-compatible mode of operation; this operation mode can be used to communicate with the microcontroller. The connection between STM32 and AD9959 is shown in Figure 3-3. By means of these commands, the simple sine wave with adjustable amplitude, frequency and phase could be generated. The specific software part will be described in section 4.2.3.

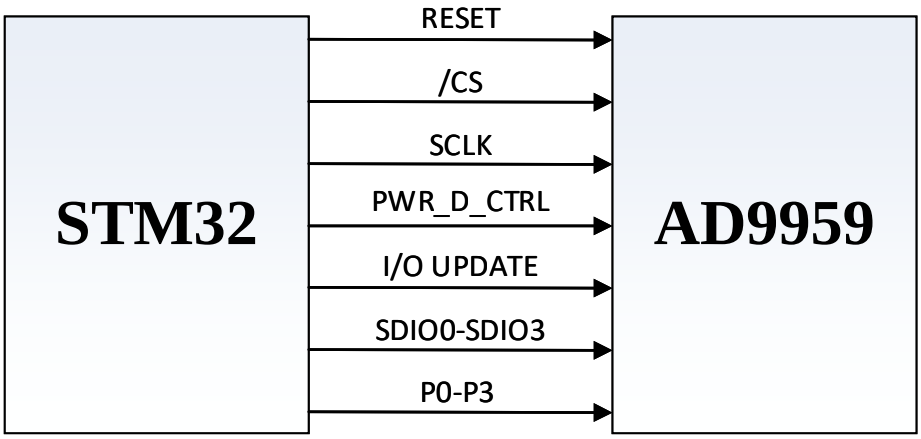


Figure 3-3 Control Pins

3.1.2 Pre-Amplifier

After measuring AD9959 through the oscilloscope, the amplitude is 0.236V when the frequency is up to 10MHz, so a pre-amplifier is required to increase the drive capability of the input signal. At the same time, due to the limited input range (-1V to 1V) of the mixer (AD835), the gain of the pre-amplifier can be set to 2-4V/V.

3.1.2.1 Design of Pre-amplifier

According to the above requirements, the gain is set to 2.4V/V, the bandwidth is 10MHz, a low-noise (For high resolution) is also needed. Finally, THS3001 is chosen as the pre-amplifier chip; it has a very fast slew rate (6500-V/μs), a 420-MHz bandwidth, and 40-ns settling time. In addition, it offers only 3mV (max) input offset voltage and 1.6nV/√Hz when frequency is 10MHz and gain is 2.

There are two ways for amplification: Non-inverting and Inverting. For non-inverting mode, it has a larger input impedance, but it does not have a virtual ground, which produces a large common-mode voltage. Thus, this way has weak anti-interference ability. For inverting mode, it has almost opposite cases. Also, in the inverting mode, there is no resistance from the non-inverting input to ground, but the non-inverting mode will certainly be driven from a source with a non-zero source resistance. This non-zero source resistance provides a path for the non-inverting input noise current and the source resistance also has an associated thermal noise, it causes the non-inverting configuration has two additional noise sources [4]. In reality, a stable amplifier is quite essential, so the inverting mode is could be better.

The one channel of pre-amplifier is shown in Figure 3-4, through the above analysis, the default way is set to inverting mode, it can be changed into non-inverting mode by replacing some resistors, the gain of inverting mode can be written as:

$$V_o = -\frac{VR_1}{R_{12}} * V_{IN} \qquad (3-1)$$

where VR_1 is a variable resistor. By setting the value of VR1, it could be easy to get a suitable value.

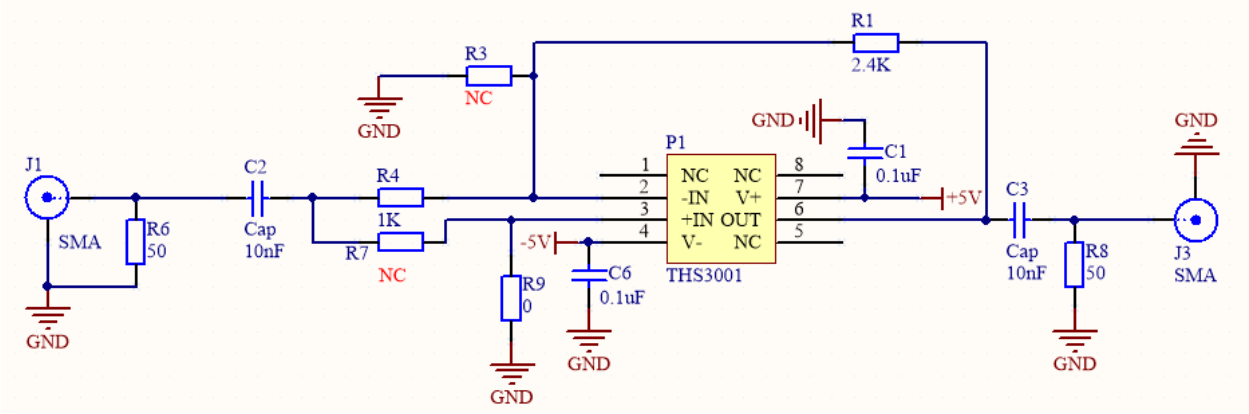


Figure 3-4 Schematic of pre-amplifier

3.1.2.2 Simulation

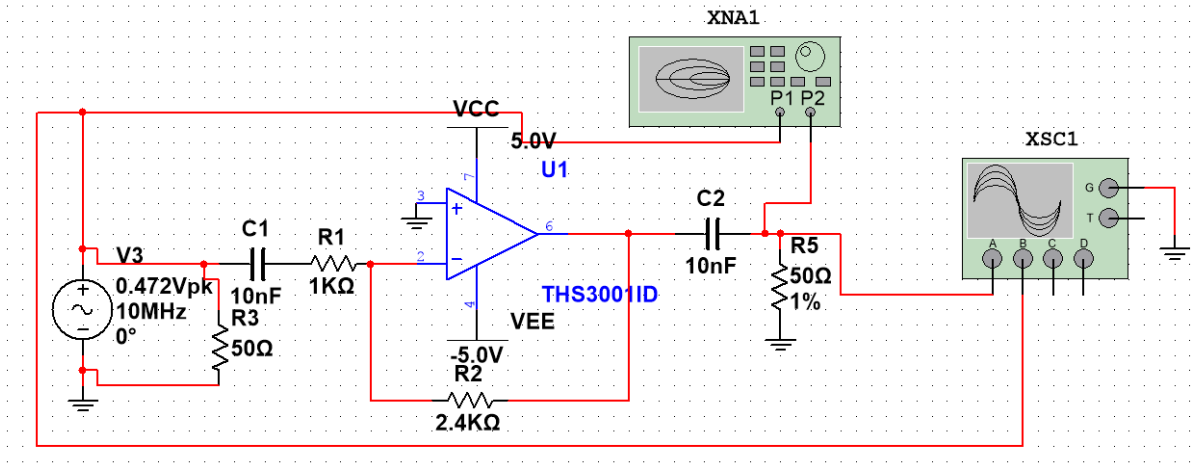
Before the real circuit is realized, a simulation is necessary; there are lots of useful information from simulation, such as gain, bandwidth and noise.

When gain is 2.4, the input signal is a sinewave signal with 10MHz frequency as well as 0.236V amplitude. The structure and some results show in Figure 3-5. The main characteristics of THS3011 with feedback network are simulated in Multisim and tabulated in Table 3-1.

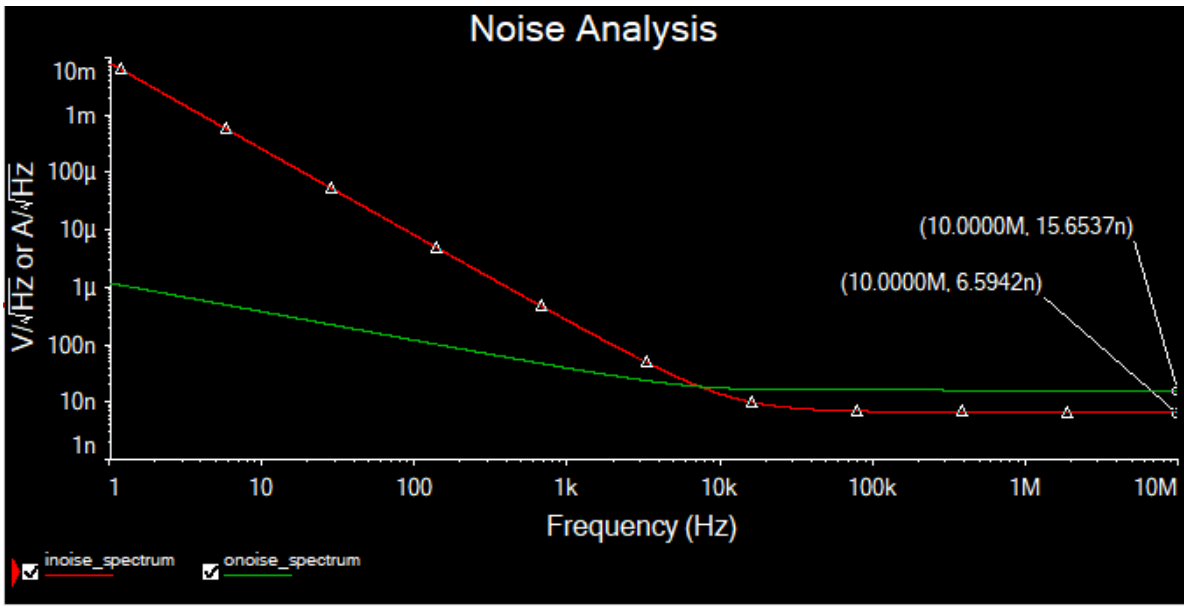
Table 3-1 Main characteristics of the pre-amplifier circuit

Gain	Input-referred noise [nV/√Hz]	Output-referred noise [nV/√Hz]	Input Offset [mV]	Bandwidth [MHz]
2.4	6.59@10M	15.65@10M	0.2	57.90

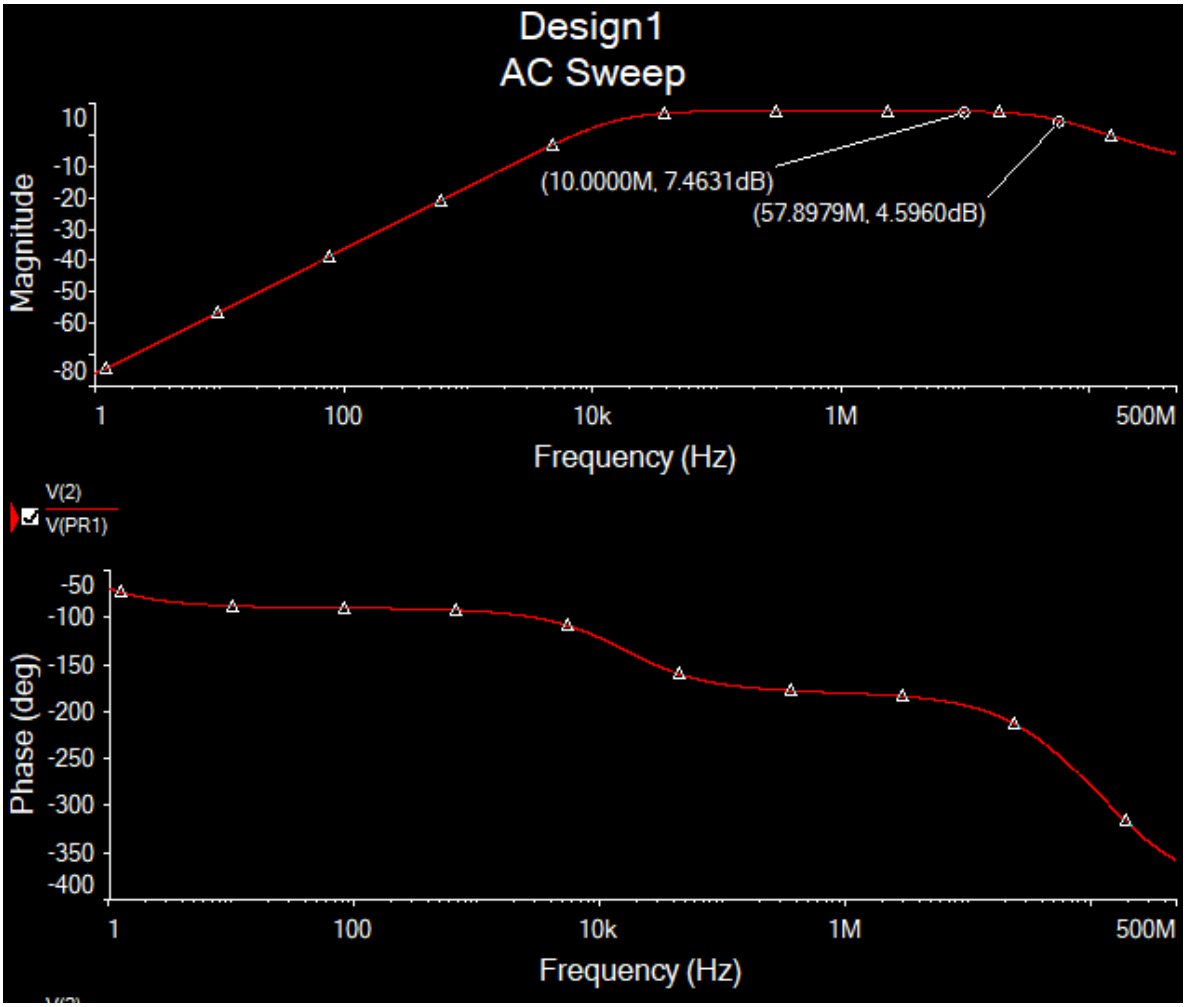
When the impedance bridge is balanced, the output current is 0, according to the equation 1-10, the capacitance value can be obtained. However, due to the resolution limit of the input signal (10-bit amplitude), there are some remaining bits. Taking the 32pF reference capacitor as an example, the capacitance to be measured is 47 pF, and equation 1-10 can be simplified to: $C_X = -V_2 * C_{REF} * \cos\phi / V_1$, where V_2 is 0.57V and ϕ is 180degree. At this point, it can be calculated that V1 is about 0.39V. In fact, V1 has only 10-bit in the system, so the remaining bits will generate a voltage of 0.27mV and step value is 0.23mV approximately. The output noise of the preamplifier is 49.49nV (15.65*√10, 10Hz is the bandwidth of the low-pass filter), which is much smaller than remaining voltage, so the noise of the preamplifier meets the requirements of the system.



(a) Simulation test bench



(b) Noise of the pre-amplifier circuit



(c) The gain in 10MHz and -3dB bandwidth (shows in -dB)

Figure 3-5 Result of simulation

3.1.3 Capacitive sensor and charge amplifier

As shown in Figure 3-6, after the input signal balances the capacitive sensor, a weak signal with specific information comes out (remaining bits of DDS, which is μV level), so an ultra-low noise charge amplifier is needed. Due to the properties of the dielectric materials, there is a parallel parasitic resistance (shunt resistance) as the loss term, typical values for this resistance are usually hundreds of kiloohms to hundreds of megaohms when the sensing capacitance is in the range of picofarads. According to equation 3-2 and 3-3, the parasitic resistance also affects the output signal accuracy of the sensor, in order to reduce this error and improve the accuracy of the system, a high-frequency (up to 10MHz) signal source is necessary (At higher frequencies, the impedance of the capacitive circuit becomes smaller, eliminating the effect of the resistive feedback path effectively). In order to bias the amplifier properly (offer a DC path for the input bias current of the amplifier) and prevent the output voltage from drifting over time until the op-amp saturates (due to the finite output offset voltage and the input bias current), a feedback resistor (R_F) is required.

$$Z_{REF} = R_{REF}//C_{REF} = \frac{R_{REF} * \frac{1}{j\omega C_{REF}}}{R_{REF} + \frac{1}{j\omega C_{REF}}} = \frac{R_{REF}}{1 + sC_{REF}R_{REF}} \qquad (3-2)$$

$$Z_X = \frac{R_X}{1 + sC_XR_X} \qquad (3-3)$$

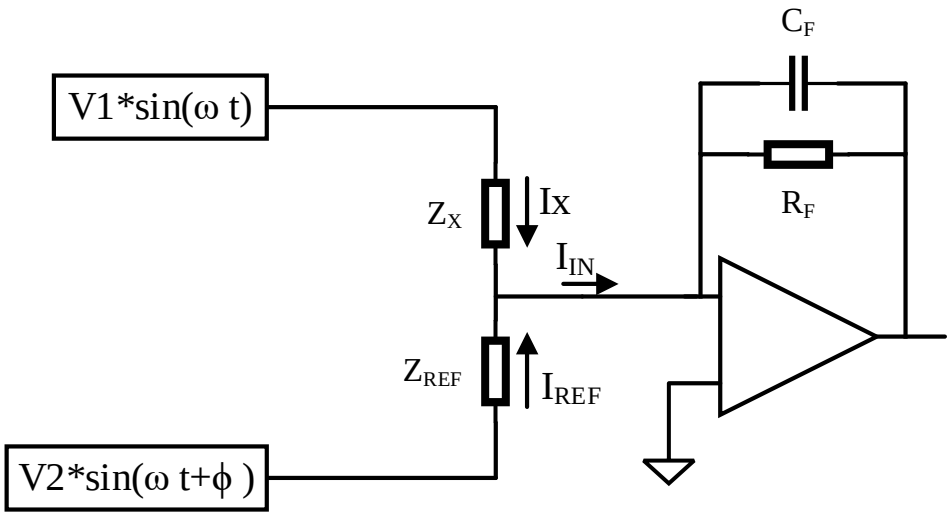


Figure 3-6 Basic structure of the capacitive sensor and charge amplifier

In order to calculate the capacitor and resistor value when the system become stable (output is equal to 0), the formula of R_x and C_x are needed.

When $I_{IN} = 0$,

$$\frac{V_X}{Z_X} = -\frac{V_{REF}}{Z_{REF}} \qquad (3-4)$$

Where $V_x = V_1*\sin(\omega t)$ and $V_{REF} = V_2*\sin(\omega t+\phi) = V_2*(\sin(\omega t)\cos\phi + \cos(\omega t)\sin\phi)$.

Then, according to the Laplace transform,

$$\frac{V_1 * \frac{\omega}{(s^2+\omega^2)}}{Z_X} = -\frac{V_2 cos\phi * \frac{\omega}{(s^2+\omega^2)} + V_2 sin\phi * \frac{s}{(s^2+\omega^2)}}{Z_{REF}} \quad (3-5)$$

Next, compute Z_X,

$$Z_X = -\frac{V_1\omega}{V_2 cos\phi\omega + V_2 sin\phi s}\frac{R_{REF}}{1+sC_{REF}R_{REF}} = -\frac{\frac{V_1\omega R_{REF}}{V_2\omega(cos\phi - \omega C_{REF}R_{REF}sin\phi)}}{1+\frac{V_2(sin\phi+\omega C_{REF}R_{REF}cos\phi)}{V_2\omega(cos\phi-\omega C_{REF}R_{REF}sin\phi)}s} \quad (3-6)$$

$$= \frac{R_X}{1+sC_XR_X}$$

So finally,

$$R_X = -\frac{V_1 R_{REF}}{V_2}\frac{1}{(cos\phi - \omega C_{REF}R_{REF}sin\phi)} \quad (3-7)$$

$$C_X = \frac{\frac{V_2(sin\phi+\omega C_{REF}R_{REF}cos\phi)}{V_2\omega(cos\phi-\omega C_{REF}R_{REF}sin\phi)}}{\frac{V_1\omega R_{REF}}{V_2\omega(cos\phi-\omega C_{REF}R_{REF}sin\phi)}} = -\frac{V_2(sin\phi+\omega C_{REF}R_{REF}cos\phi)}{V_1\omega R_{REF}} \quad (3-8)$$

3.1.3.1 Noise analysis

A brief theoretical noise analysis can be helpful before proceeding with the and design and the simulation, which can maximize the signal-to-noise ratio (SNR). Figure 3-7 shows the main noise sources in the charge amplifier [6]. The output noise density can be expressed as:

$$e_{no} = \sqrt{I_{NA}^2 \times (Z_F)^2 + e_A^2 \times \left(1+\frac{Z_F}{1/(C_{IN}+C_P)s}\right)^2 + e_{RF}^2 \times \left(\frac{1}{1+R_FC_Fs}\right)^2} \quad (3-9)$$

$$Z_F = \frac{R_F}{1+R_FC_Fs} \quad (3-10)$$

where s = j×2πf, C_{IN} is the sensing capacitance and C_P is the input parasitic capacitance of the amplifier.

If C_P can be ignored, the second term will be reduced even further if frequencies well above the high-pass filter's pole are considered [6]. So, equation 3-9 can be simplified to:

$$e_{no} = \sqrt{I_{NA}^2 \times \left(\frac{R_F}{1+R_FC_Fs}\right)^2 + e_A^2 \times \left(1+\frac{C_{IN}}{C_F}\right)^2 + e_{RF}^2 \times \left(\frac{1}{1+R_FC_Fs}\right)^2} \quad (3-11)$$

For further analysis, the pole (the term $1+R_FC_Fs$) can be considered constant when the C_F capacitance value is decreased and at the same time the R_F resistance value is increased. From equation 3-11, it can be seen that all three terms would increase when R_F increases. The voltage noise corresponding to the first term would increase; the voltage noise related to the op-amp (the second term of the equation) would increase linearly with R_F; and the voltage noise related to the feedback resistance (the third term) would also increase. Following the thermal noise of the resistance, expressed as $e_{RF} = \sqrt{4KTR_F}$, where K is the Boltzmann's constant, $1.38*10^{-23}JK^{-1}$, and T is the temperature in K. Simultaneously, the gain($-C_{IN}/C_F$) of charge amplifier would increase with R_F as C_F becomes smaller. This increase of

signal with R_F will be similar to any increase of the first two noise terms in equation 4-13, but bigger than the increase of the third noise term, therefore improving the overall SNR [6]. Another thing to notice that the R_F resistance value cannot be increased without limit, because high resistance values are extremely difficult to implement on a PCB. Also, the R_F combined with the input bias current (I_{ib}) of the amplifier generates a non-negligible output DC offset of the amplifier. Furthermore, from the noise point of view, a sensor with more parasitic capacitance is less desirable.

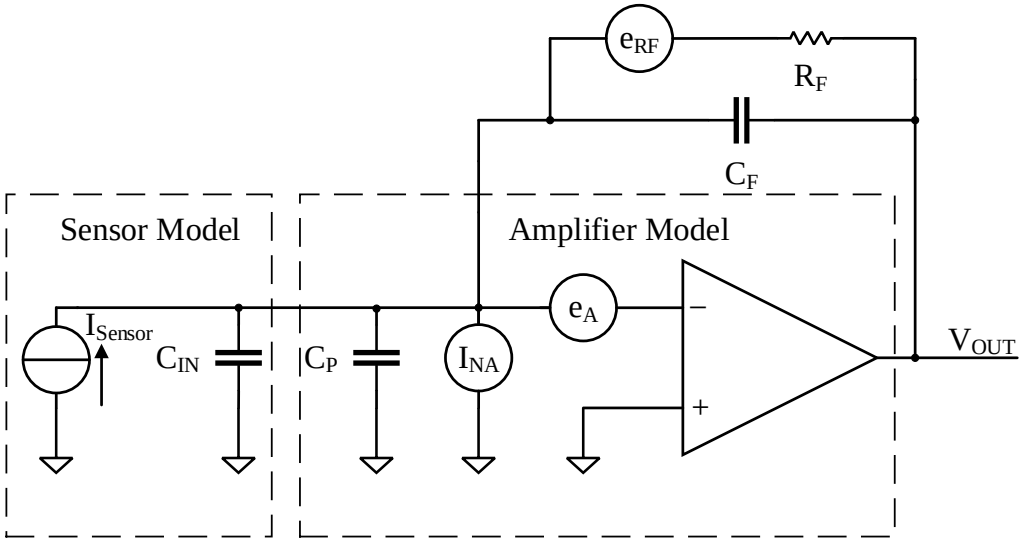


Figure 3-7 Noise sources of the charge amplifier

Based on the above analysis, LTC6268-10 is a good choice. It is a single 4GHz FET- input operational amplifier with extremely low input bias current (±3fA) and low input capacitance (0.45pF). It has an ultra-low input-referred current noise (7fA/√Hz@100KHz) and voltage noise(4.0nV/√Hz@1MHz) making it an ideal choice for high-impedance sensor amplifiers. It is also a decompensated op amp that is gain-of-10 stable [7]. According to the equation 3-11 and the example in section 3.1.2.2, it is possible to calculate the output noise and the input-referred noise of the charge amplifier when T, R_F and C_F are 298 in K, 250MΩ and 5pF respectively, e_{no} is equal to 67.20nV/√Hz and e_{ni} is equal to 4.25nV/√Hz. So, input-referred noise is 13.44nV (4.25 * √10, 10Hz is the bandwidth of the low-pass filter), compared to the remaining voltage in section 3.1.2.2, which is much smaller than remaining voltage, so the noise of the charge amplifier meets the requirements of the system.

If parasitic resistance is considered, then equation 3-9 can be expressed as:

$$e_{nop} = \sqrt{\begin{array}{c} I_{NA}^2 \times (Z_F)^2 + e_A^2 \times \left(1 + \frac{Z_F}{Z_{IN}}\right)^2 + \\ e_{RF}^2 \times \left(\frac{1}{1 + R_F C_F s}\right)^2 + e_{RIN}^2 \times \left(\frac{1}{1 + R_{IN} C_{IN} s}\right)^2 \times \left(1 + \frac{Z_F}{Z_{IN}}\right)^2 \end{array}} \qquad (3-12)$$

$$Z_{IN} = \frac{R_{IN}}{1 + R_{IN} C_{IN} s} \qquad (3-13)$$

where R_{IN} is a parasitic resistance in parallel with the sensing capacitor.

Because R_{IN} is a MΩ level so that equation 3-12 can be simplified to:

$$e_{nop} = \sqrt{\begin{matrix} I_{NA}^2 \times \left(\frac{R_F}{1+R_F C_F s}\right)^2 + e_A^2 \times \left(1+\frac{C_{IN}}{C_F}\right)^2 \\ +e_{RF}^2 \times \left(\frac{1}{1+R_F C_F s}\right)^2 + e_{RIN}^2 \times \left(\frac{1}{1+R_{IN} C_{IN} s}\right)^2 \times \left(1+\frac{C_{IN}}{C_F}\right)^2 \end{matrix}} \quad (3-14)$$

Do the same calculations as above, the output-referred noise and input-referred noise density with different parasitic resistance values(e_{nop}) are tabulated in Table 3-2.

Table 3-2 Input/output referred noise with different R_{IN}

R_{IN} / Noise density	1MΩ	3.3MΩ
Input-referred noise	4.25 nV/√Hz	4.25 nV/√Hz
output-referred noise	67.201 nV/√Hz	67.200 nV/√Hz

As we can see from Table 3-2, the noise provided by the parasitic resistance is very small compared to when there is no parasitic resistance, and the larger the parasitic resistance, the less noise, but these differences are entirely negligible.

3.1.3.2 Charge amplifier based on LTC6268-10

The schematic of the LTC6268-10 circuit is shown in Figure 3-8, it operates on ±2.5V supply and works at inverting configuration. When setting the value of the feedback resistor, the trade-off between bandwidth and DC offset must be made, according to the input bias current, input offset, signal frequency and capacitor value of the sensor, R_F and C_F are 250MΩ and 5pF respectively.

In this design, in order to reduce the interference of the external environment, the metallic shield is used. Meanwhile, it is also possible to protect high-impedance nodes in the circuit from surface leakage currents by using an active guard ring (The guard ring is a ring of copper driven by a low-impedance source and has the same voltage as the high impedance node. It is usually input pin of the amplifier) on PCB.

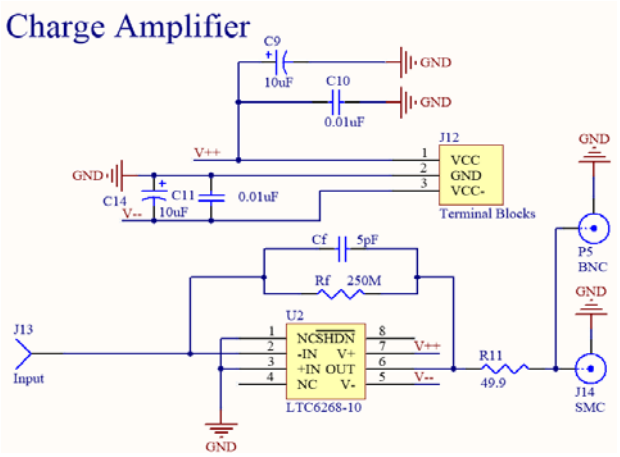
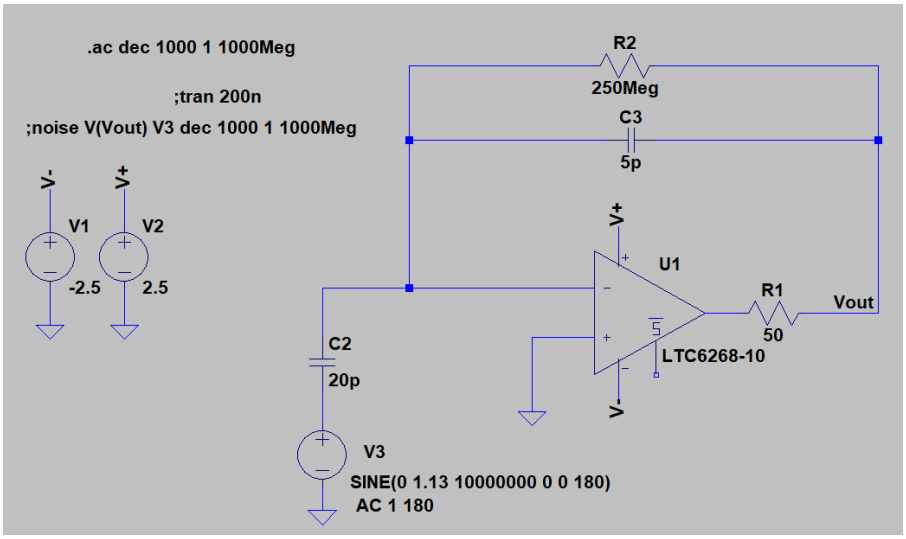


Figure 3-8 Schematic of the LTC6268-10

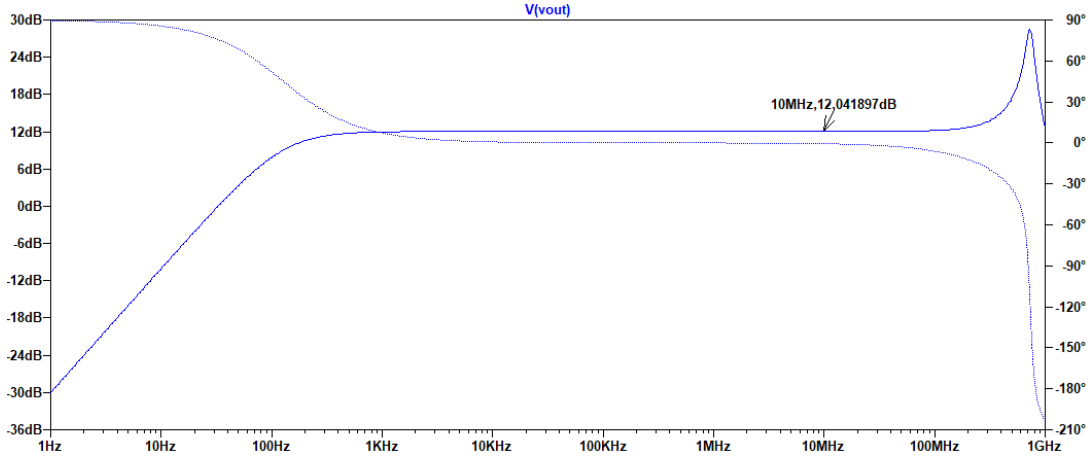
3.1.3.3 Simulation

As shown in Figure 3-9 (a), the basic simulation structure is built in LTspice, it has sinewave input with 10MHz frequency and 0.236V amplitude, R_2 (R_F) and C_3 (C_F) are 250MΩ and 5pF respectively, C_1 is regarded as a capacitive sensor with 20pF due to the real capacitive sensor.

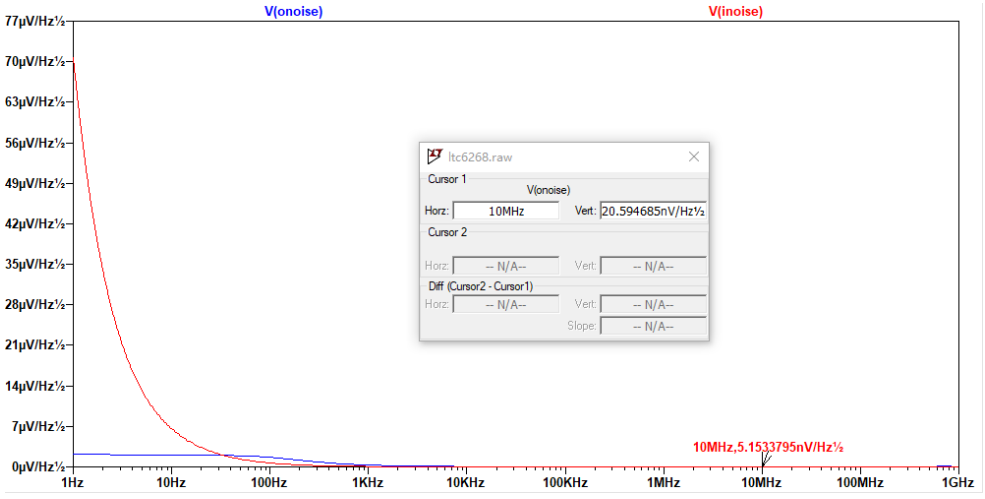
From Figure 3-9 (b) and (c), the gain is 12.04dB (4V/V) and the input/output referred noise density is 5.15nV/√Hz and 20.61nV/√Hz respectively at 10MHz. According to the above analysis, this simulation meets the requirements of the charge amplifier.



(a) Simulation test bench



(b) Gain @10MHz



(c) Input/output referred noise density

Figure 3-9 Result of simulation

Furthermore, in order to verify the noise and parasitic resistance analysis in section 3.1.3.2, the same structure was built in the simulation (Figure 3-10), and the results are shown in Table 3-3.

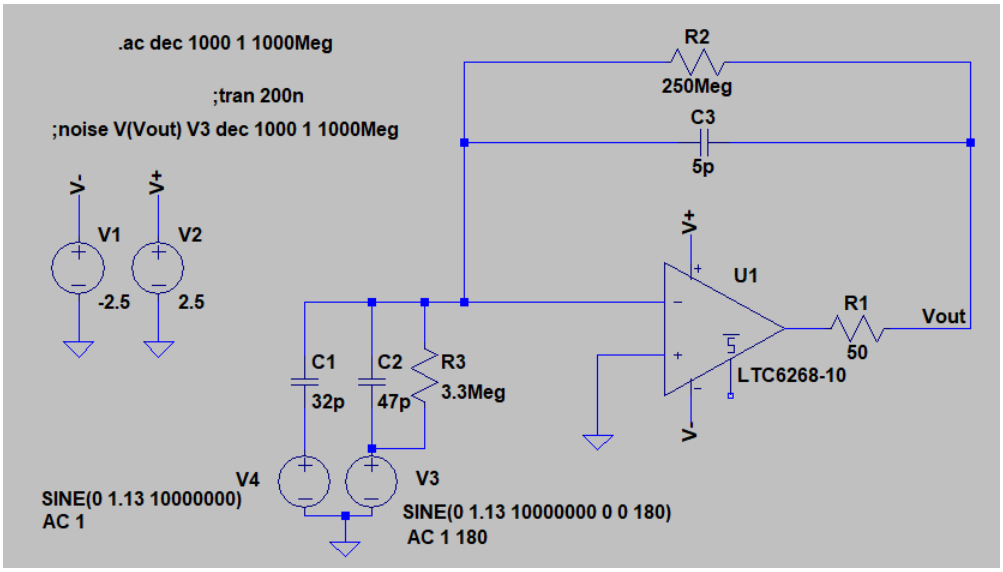


Figure 3-10 Simulation test bench with an impedance bridge

It can be seen that these simulation results are closed to the calculation in section 3.1.3.2, gain @10MHz meets the requirement also. However, there are many ideal components in the simulation; the actual circuit still needs to be verified by measuring these characteristics.

Table 3-3 Charge amplifier performance with impedance bridge in simulation

Characteristics	Without R_{IN}	R_{IN} =3.3MΩ
Gain	9.54dB (3V/V)	9.54dB (3V/V)
Input-referred noise density	4.29 nV/√Hz	4.29nV/√Hz
output-referred noise density	67.84nV/√Hz	67.84nV/√Hz

3.2 Second amplifier

Due to the bandwidth limitation of the charge amplifier, the balanced output is still weak (μV level), in order to provide enough amplitude (mV) for the further stage, the high gain amplifier (20V/V) with large GBP is required.

3.2.1 High-gain amplifier

According to the above request, OPA846 is suitable for this design; it has a low input noise voltage ($1.2nV/\sqrt{Hz}$), a high gain-bandwidth product (1.75GHz) and a large CMRR (110dB). It is also optimized for a flat frequency response at a gain of +10V/V and is stable down to gains as low as +7V/V.

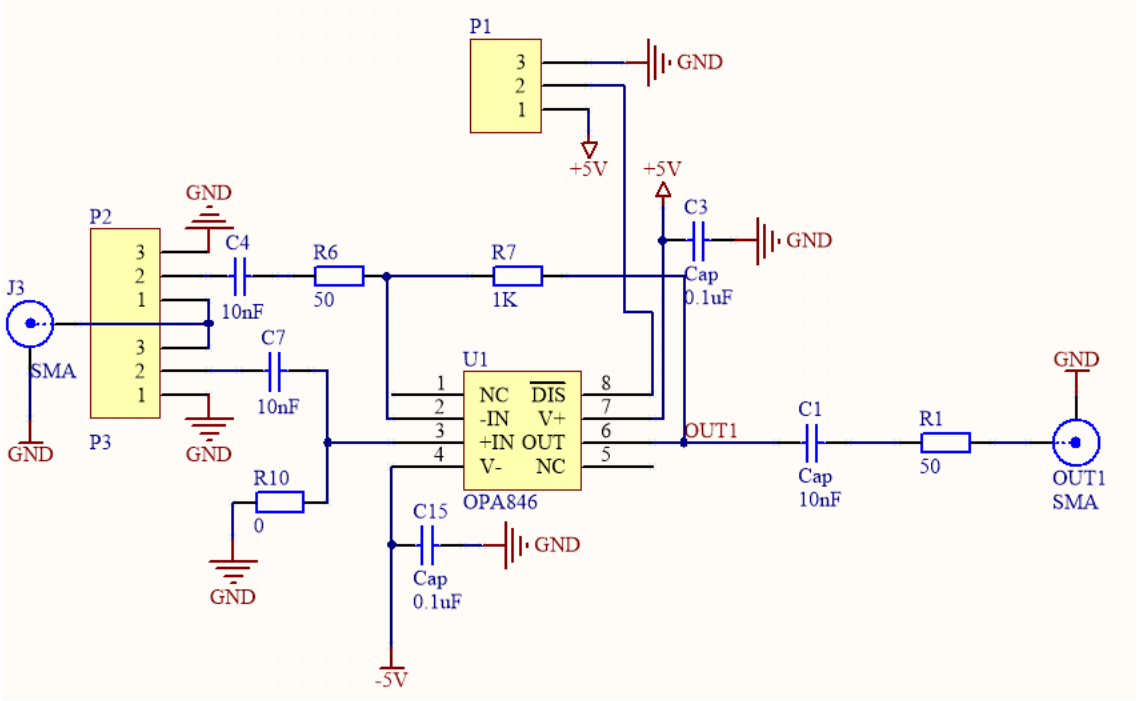
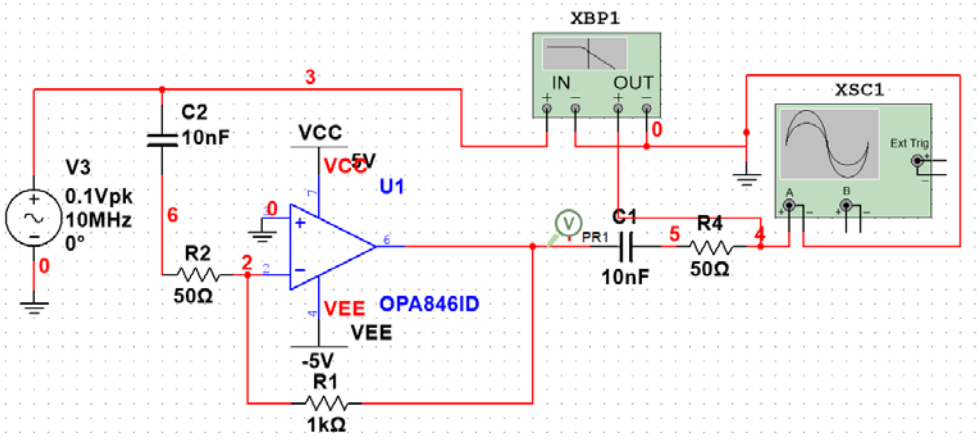


Figure 3-11 Structure of the Second amplifier

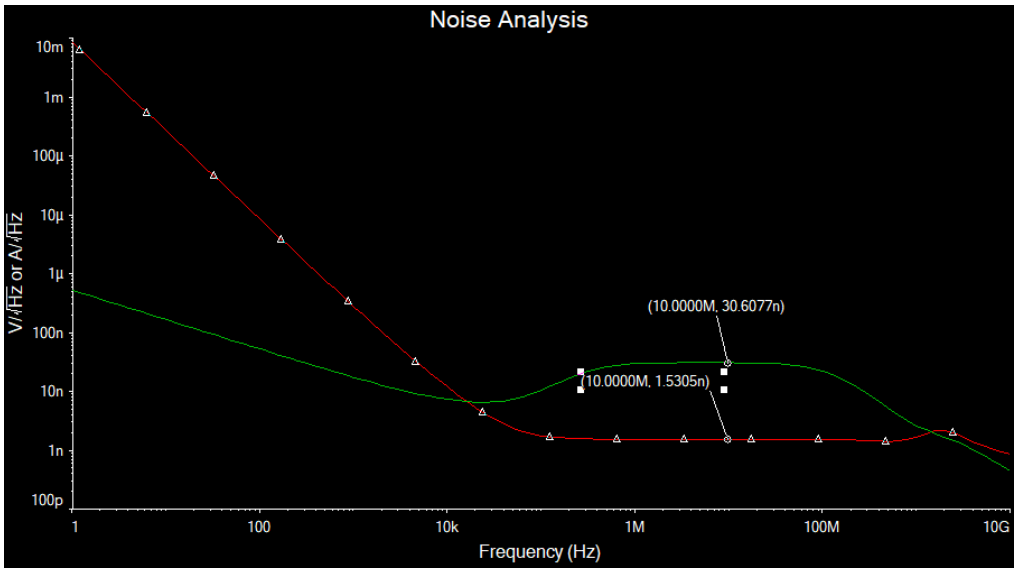
The design way is shown in Figure 3-11, there are two pins connect with non-inverting input and inverting input respectively, which provide an easy way to choose the different mode for input. In the schematic design, taking into account the DC offset at output result, a capacitor is usually applied at the output of the op-amp (10nF is suitable for 10MHz frequency).

3.2.2 Simulation

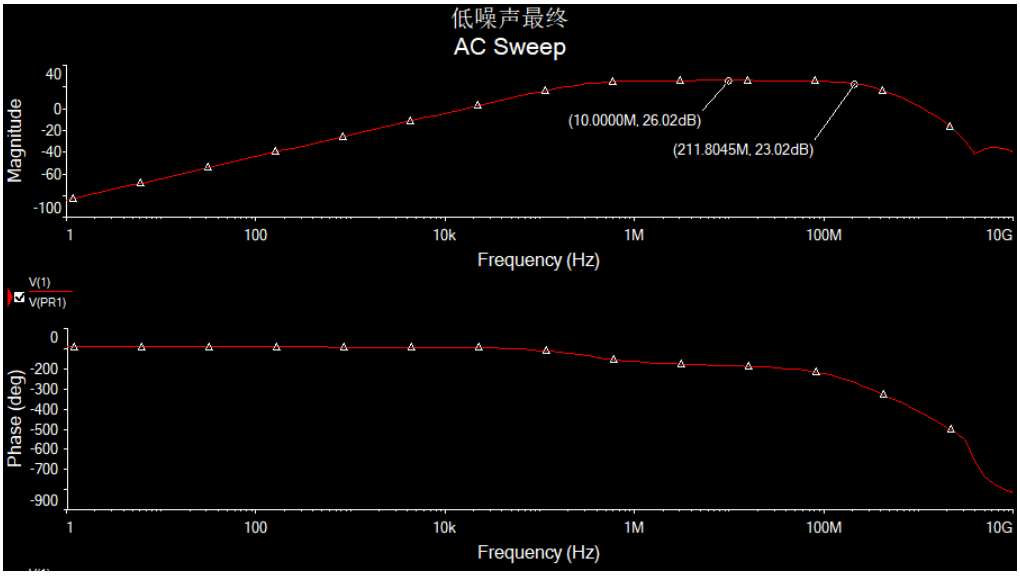
For high gain amplifier in section 3.2.1, the basic simulation circuit is shown in Figure 3-12, it has sinewave input with 10MHz frequency, 0.05V amplitude and 20V/V (About 26.02dB) gain, works in inverting mode. It can be seen that the gain is 26.02dB @10MHz, the -3dB bandwidth is 211MHz and the input voltage noise density is 1.53nV/√Hz at 10MHz, which meet the requirement of the high-gain amplifier.



(a) Simulation test bench



(b) Noise analysis



(c) AC sweep

Figure 3-12 Simulation result

3.3 Phase-sensitive detector and low-pass filter

Since the signal works at high frequency, MCU cannot collect it directly, phase sensitive detector and low-pass filter can bring their real effect here. The schematic of PSD and LPF are shown in Figure 3-13.

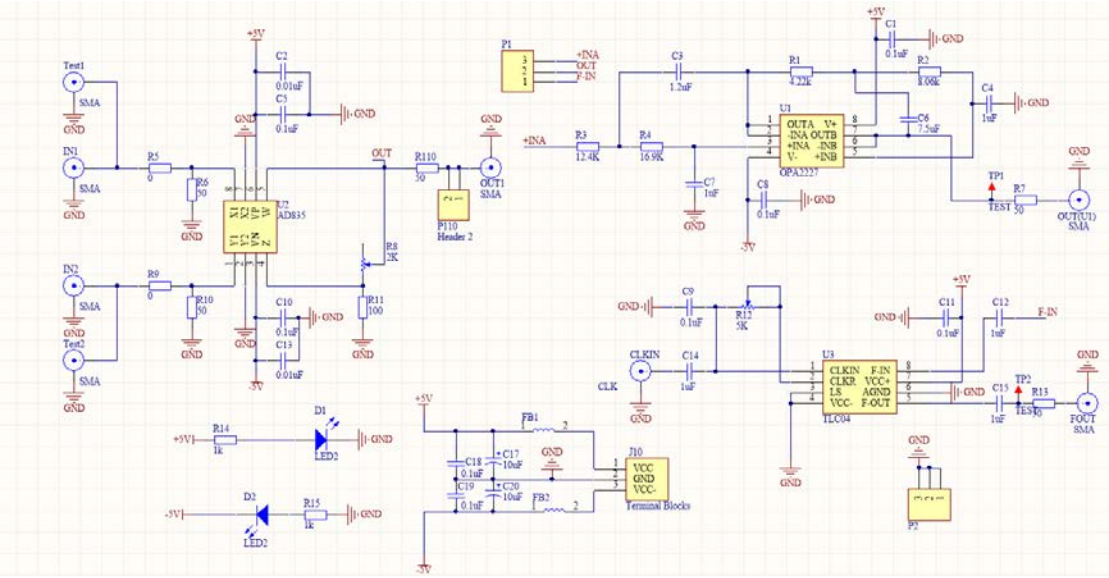


Figure 3-13 Circuit design of PSD and LPF

3.3.1 Mixer (PSD)

Analog multiplier can work as a mixer easily, which has more simple circuit structure than chopper at high frequency (10MHz or even higher).

The suitable analog multiplier for this design is AD835, and it is a complete four-quadrant, voltage output analog multiplier. The linear product of its X and Y voltage inputs can be generated with a −3 dB output bandwidth of 250MHz. The slew rate is 1000 V/us and it costs 20 ns for the settling time to 0.1% typically. However, it has a large output offset (up to ±75mV), this offset is eliminated by software in this design (see Chapter 4 for details.).

The on-resistance (RON), charge injection and leakage current are main parameters of the analog switch should be considered for the chopper. In theory, they should be as small as possible. The low RON reduces the input signal losses and minimizing RON and the parasitic capacitor can also improve the linearity of RON versus VIN over temperature and voltages. The small charge injection can decrease output voltage change by ±ΔVOUT (a few millivolts). The low leakage current and a high on-off current ratio (This ratio describes the ability of a device to switch from the on state to the off-state) provide more accuracy VOUT and almost ideal floating ±1 switch.

After consideration of transition time (turn-on and turn-off time should be less because of the high frequency), there are some suitable switches, such as ADG711, 74HC4066 and TS3A4751. The first reason is transition time, for 10MHz sinewave signal, one cycle needs 100ns, but transition time of most of the switches are longer than 10ns, it can cause signal distortion. The other main reason is the power supply, most of the switches with a small transition time are the single power supply, some of them are smaller than 5V, if these kinds of switches are used in this design, the circuit would be more complicated.

In conclusion, AD835 can be the best choice as a multiplexer.

3.3.2 Lowpass filter

In this part, it has two different lowpass filter, which provides the final output signal (DC signal). The

first is 2 cascaded 2nd order Sallen-Key lowpass filter (OPA2227), which can be built by cascading two building blocks made of the second-order low-pass filter. Its topology is shown in Figure 3-14.

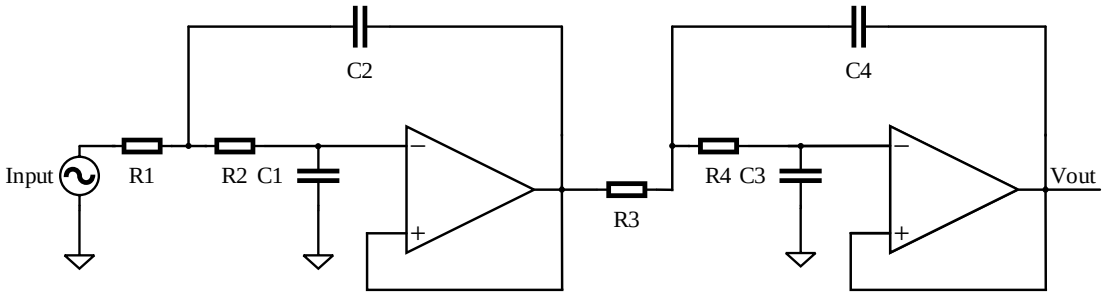


Figure 3-14 2 cascaded 2nd-order Sallen-Key topology

The transfer function H(s) and its cut-off frequency f_c are expressed as:

$$H(s) = \frac{1}{1 + C_1(R_1 + R_2)s + C_1C_2R_1R_2s^2} \times \frac{1}{1 + C_3(R_3 + R_4)s + C_3C_4R_3R_4s^2} \qquad (3-14)$$

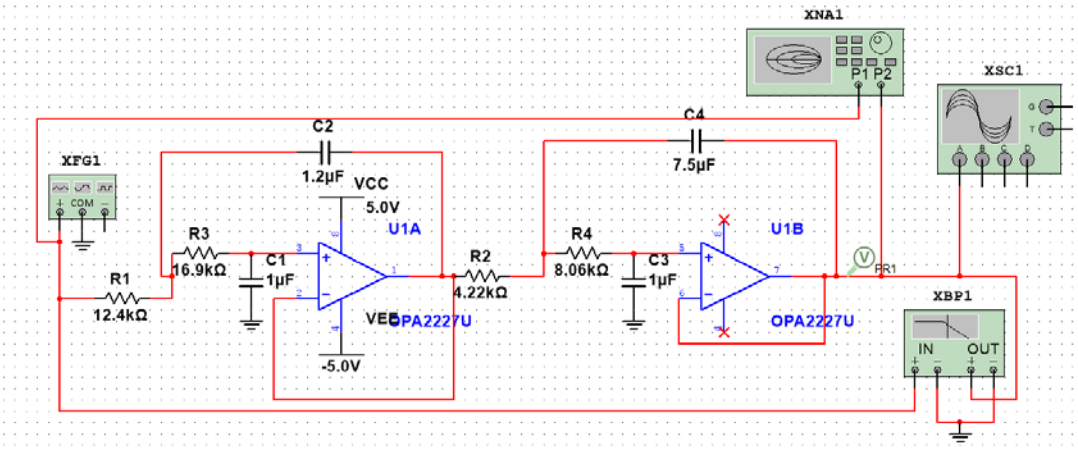
$$f_{c1} = \frac{1}{2\pi\sqrt{C_1C_2R_1R_2}} \; and \; f_{c2} = \frac{1}{2\pi\sqrt{C_3C_4R_3R_4}} \qquad (3-15)$$

Because the output is DC signal, f_{c1} and f_{c2} can be 10Hz (it can also filter the frequency of the network 50 Hz from the power supply), which can filter high-frequency signal from the mixer. The values of the resistors and capacitors are chosen by software from TI shows in Figure 3-13.

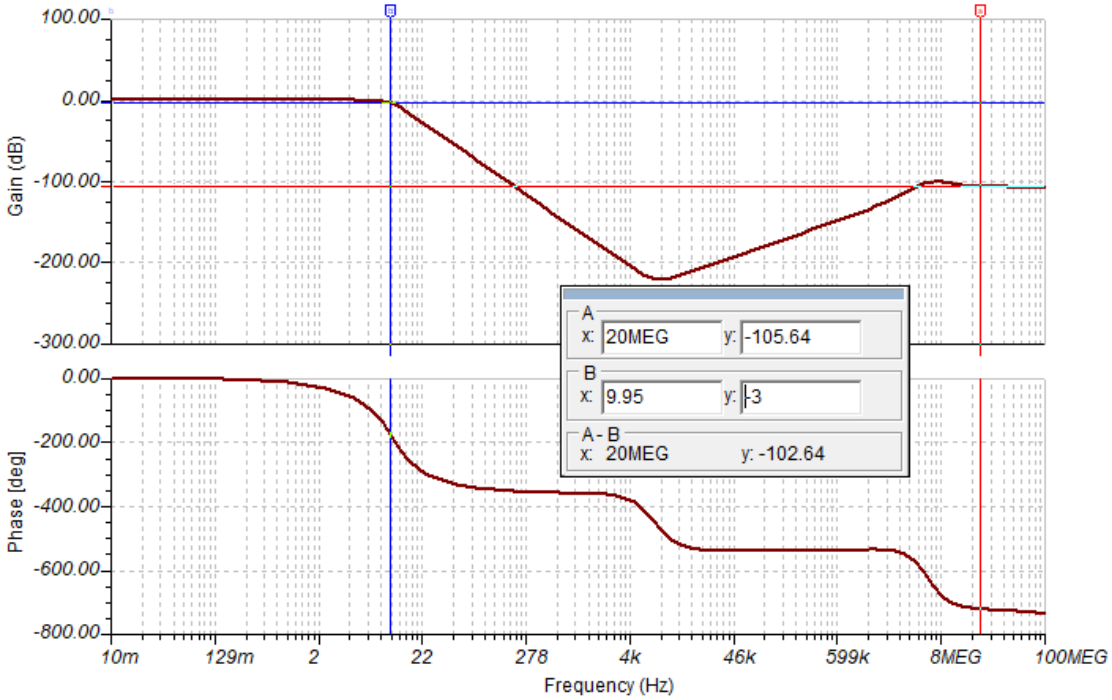
The other one is Butterworth 4th order low-pass switched-capacitor filter (TLC04), which can adjust the cut-off frequency by a potentiometer, the Clock to the cutoff-frequency ratio (f_{clock}/f_{co}) is 50.07 and the formula of f_{clock} is 1/1.69*RC.

3.3.3 Simulation

For 2 cascaded 2nd order Sallen-Key lowpass filter, it is input parameters are set as a sinewave with 20MHz frequency, 1V amplitude and 0.5V DC offset. Simulation circuit and results show in Figure 3-15, where the gain is 1V/V, -3dB bandwidth is 9.95Hz.



(a) Simulation circuit



(b) -3dB bandwidth and gain in 20MHz frequency

Figure 3-15 Simulation circuit and results of the lowpass filter

3.4 MCU and its subsidiary system

This part consists of a 24-bit ADC (ADS1256), a 32-bit flash microcontroller (STM32F103RB) and some chips for connection. The specific control method will be shown in section 4.2. The topology of this part is shown in Figure 3-19.

3.4.1 MCU

There are many factors to consider when choosing an MCU, for example, the speed of data processing, price cost, and internal resources.

So, there are three aspects to take into account when choosing an MCU:

(1). The role of MCU in the entire design and the complexity of the task: In this design, MCU is mainly responsible for AD9959 control, signal acquisition, signal data processing and communication with PC.

(2). Simplify the design of the entire system: The more integrated functional unit of MCU, the better. In this case, not only simplifies the system design but also increase the reliability of the system.

(3). System production costs: Replacement cost should be low, which can reduce the cost of MCU and system.

Based on the above four factors, STM32F103RB is chosen, which is the first 32-bit RISC (Reduced Instruction Set) processor based on the ARM®Cortex®-M3 architecture, which provides high code efficiency and shows the high performance of the ARM core on storage of typically 8- and 16-bit systems. This series microprocessor operating frequency is 72MHz, and the built-in Flash memory up to 128Kbytes. The chip has many advantages, rich internal resources, its excellent performance as shown in Table 3-4:

Table 3-4 STM32F103RB characteristic

7-channel DMA controller	Low-power (2.0-3.6V)
2 x 12-bit, 1 μs A/D converters (up to 16 channels)	Up to 51 fast I/O ports
Seven timers	Up to 2 SPIs (18 Mbit/s)
20 Kbytes of SRAM	Up to 3 USARTs

3.4.2 ADC

After the low-pass filter, continuous DC signals are generated, by using the analog-to-digital converter to collect and transmit them to MCU for processing.

There are generally two options for ADC: an integrated analog-to-digital converter in the microcontroller and an external analog-to-digital converter, which depend on the accuracy of the system. In order to obtain an accurate output, external high-precision ADC is used. According to the system performance requirements (10-bit amplitude and 14-bit phase resolution), considering the technical information provided, price and other factors, this design uses the ADS1256 as ADC of the circuit.

The ADS1256 is a high-speed, low-noise, 24-bit ADC that provides a complete high-resolution measurement solution for analog signal. Its data rate of up to 30kSPS, the analog signal input voltage is 0-5V, and the digital signal output voltage is 1.8V to 3.6V. Standard operating mode power consumption is only 38mW [8]. Communication is handled over an SPI-compatible serial interface.

As shown in Figure 3-16, the whole structure of the ADC circuit is presented. Because this chip does not have an internal reference, the external voltage reference (REF3225) is needed, which provides a 2.5V with 0.01% error and very low-temperature drift.

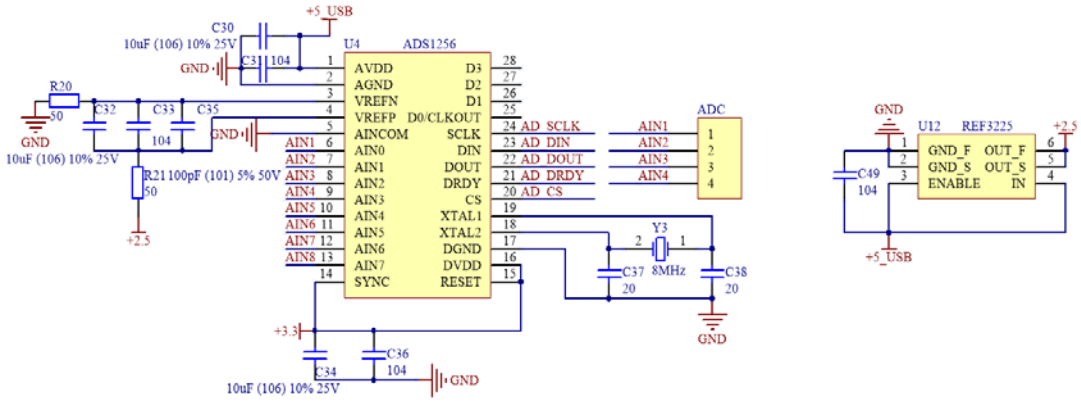


Figure 3-16 Schematic of ADC

Because the software portion of the analog-to-digital converter has not been debugged, the 16-bit ADC in the data acquisition box (DAQ, NI6363) is used to collect the data.

3.4.3 Remaining circuits

There are also some parts show in Figure 3-19, the USB interface provides 5V from PC and the voltage supply circuit generates 3.3V for MCU, serial communication module (MAX3232 and serial port interface), JTAG interface and AD9959 interface are used as a communication interface with PC or chips.

3.5 Overall noise analysis

For further stage, in order to determine which part is the main noise source, noise analysis of the entire system is still needed. The noise model of the whole system is shown in Figure 3-17, where the reference capacitance is 32pF and the unknown capacitance is 47pF.

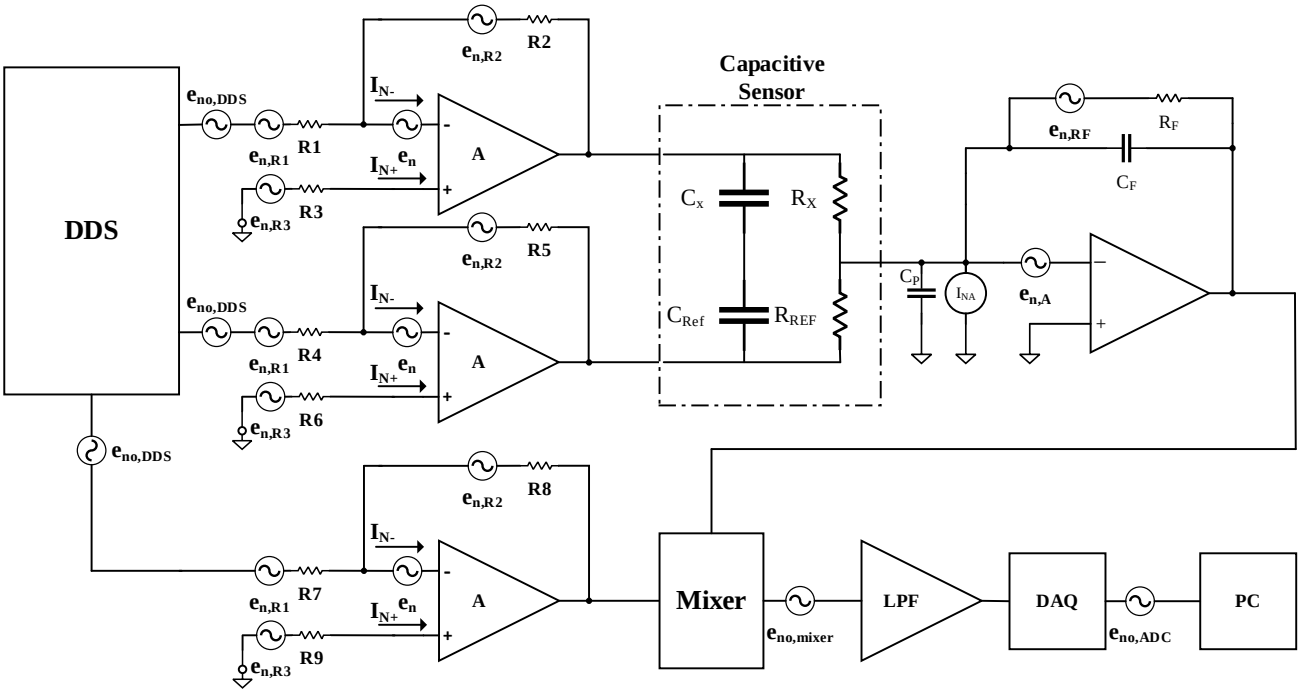


Figure 3-17 The noise model of the entire system

For DDS, its noise sources are mainly composed of truncation error, phase noise, quantization noise and harmonic noise [9]. The truncation error is generated by the truncation of the phase register, the phase noise is generated by the reference clock jitter, the quantization noise and harmonic noise are caused by the resolution and nonlinearity of the DAC (core part of DDS). The output noise is mainly caused by the quantization noise, it produces some remaining bits (voltage) for the self-balanced bridge; however, this noise is eliminated in the software by interpolation method, which increases the resolution of the system. For specific value of the remaining noise sources, they can be measured with the spectrum analyzer.

Moreover, the amplitude and phase noise from both DDS channels are subtracted in the bridge circuit as shown in the figure below.

For the uncorrelated noise the output noise of the charge amplifier can be written as:

$$\overline{V_{out}^2} = \frac{C_F^2}{C_x^2}\overline{V_{n1u}^2} + \frac{C_F^2}{C_{ref}^2}\overline{V_{n2u}^2} \qquad (3-16)$$

Since both channels are derived from the same DDS RF signal some correlation between the channels is expected. Since both channels are in opposite phase the noise will be reduced the correlated noise sources can be subtracted written as:

$$\sqrt{\overline{V_{out}^2}} = \frac{C_F}{C_x}\sqrt{\overline{V_{n1c}^2}} - \frac{C_F}{C_{ref}}\sqrt{\overline{V_{n2c}^2}} \qquad (3-17)$$

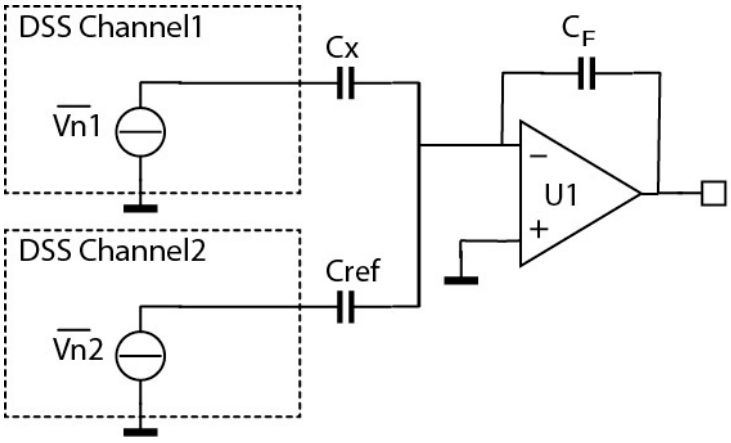


Figure 3-18 Simplified schematic for noise analysis from the DDS.

For pre-amplifier, there are six separate noise sources: the thermal noise of the 3 resistances ($e_{n,R1}$, $e_{n,R2}$ and $e_{n,R3}$), the current noise in each input of amplifier (I_{N+} and I_{N-}) and the amplifier internal voltage noise (e_n). Assuming each noise is uncorrelated, the total equivalent input noise density (e_{ni}) is calculated by using the following equation:

$$e_{ni} = \sqrt{(e_n)^2 + (I_{N+} \times R_3)^2 + (I_{N-} \times (R_1||R_2))^2 + 4kTR_3 + 4kT(R_1||R_2)} \qquad (3-18)$$

Where k is Boltzmann's constant = 1.380658×10^{-23}, T is Temperature in degrees Kelvin, $R_1 \parallel R_2$ is parallel resistance of R_1 and R_2, $4kT*(R_1 \parallel R_2)$ is equal to $4kTR_1*(R_1 \parallel R_2)^2$ plus $4kTR_2*(R_1 \parallel R_2)^2$.

To obtain the equivalent output noise density of the op amp, just multiply e_{ni} by the amplifier gain.

$$e_{no} = e_{ni}A_v = e_{ni}\left|\frac{R_2}{R_1}\right| \qquad (3-19)$$

Where A_v is the amplifier gain in inverting mode.

According to the datasheet of THS3001, the equivalent output noise density of pre-amplifier is 12.0nV/√Hz.

For charge amplifier, the specific analysis has been introduced in section 3.1.3, the equivalent output noise density of charge amplifier is 67.2nV/√Hz.

For high-gain amplifier, it has the same structure as the pre-amplifier, so the equivalent output noise density of high-gain amplifier can be easily calculated with the datasheet of OPA846, which is 31.49 nV/√Hz.

For mixer (AD835), noise sources consist of the product noise, phase noise and low frequency drift. The product noise can be regarded as the main noise source, which is 50nV/√Hz.

Except the noise sources of the DDS, the above output noise density can be put together to calculate the overall output noise density, it can be expressed as:

$$12.0 \times NG_C \times NG_H \times NG_M + 67.2 \times NG_H \times NG_M + 31.5 \times NG_M + 12.0 \times NG_M + 50 \qquad (3-18)$$

where NG_C is the noise gain (15.8V/V) of the charge amplifier, NG_H is the noise gain (20V/V) of the high-gain amplifier and NG_M is the noise gain (1V/V) of the mixer.

So, the overall output noise density is 5229.5nV/√Hz, and output noise is 16.5μV (5229.5 * √10, 10Hz is the bandwidth of the low-pass filter).

Also, as can be seen from the data sheet of the DAQ, the random noise is 17μV when the full scale of analog input is 100mV. Finally, the output noise of the whole system (without DDS) is 33.5μV, where the noise of the ADC (17μV) and the noise of the pre-amplifier (12μV) are the main noise sources, and refer the output noise back into the input of the charge amplifier, the equivalent input-referred noise of the entire system can be obtained, which is 0.11μV.

According to the previous analysis in section 3.1.2, the step value of input signal at charge amplifier stage is 553.13μV, then based on the equation 1-11, the equivalent capacitance change (without the equivalent noise from the DDS) at input of charge amplifier is 6.2aF.

3.6 Conclusion

In this chapter, the circuit-level analysis, design and simulation of the lock-in amplifier are introduced. In the front-end circuit design, principle and communication structure of DDS are discussed, then noise analysis, circuit structure and simulation of pre-amplifier and charge amplifier are presented. For a second-order amplifier, it is a high-gain amplifier, the main function of this amplifier is amplified the weak signal for the further stage with a high gain. Furthermore, a comparison between mixer and chopper is presented, mixer is chosen as PSD device; there are two low-pass filters: 4^{th} order sallen-key low-pass filter with fixed cut-off frequency and Butterworth 4^{th} order low-pass switched-capacitor filter with changeable cut-off frequency, which can help ADC obtain an excellent DC signal, the simulation of mixer and low-pass filters are given also. After that, the noise analysis of the whole system is given. Finally, the MCU design with ADC and some interface have been briefly introduced. The detailed connection between each hardware is described in Appendix I.

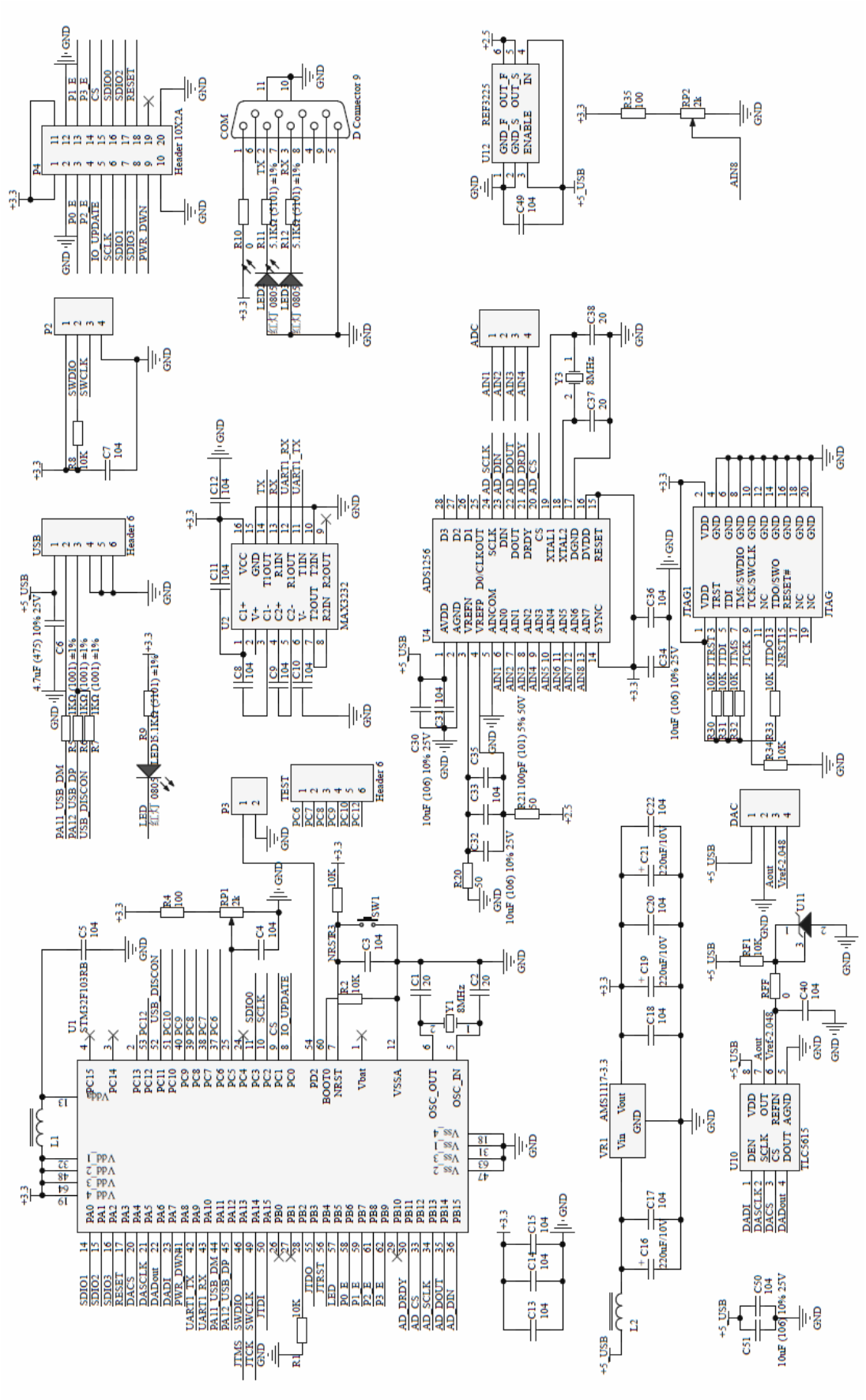
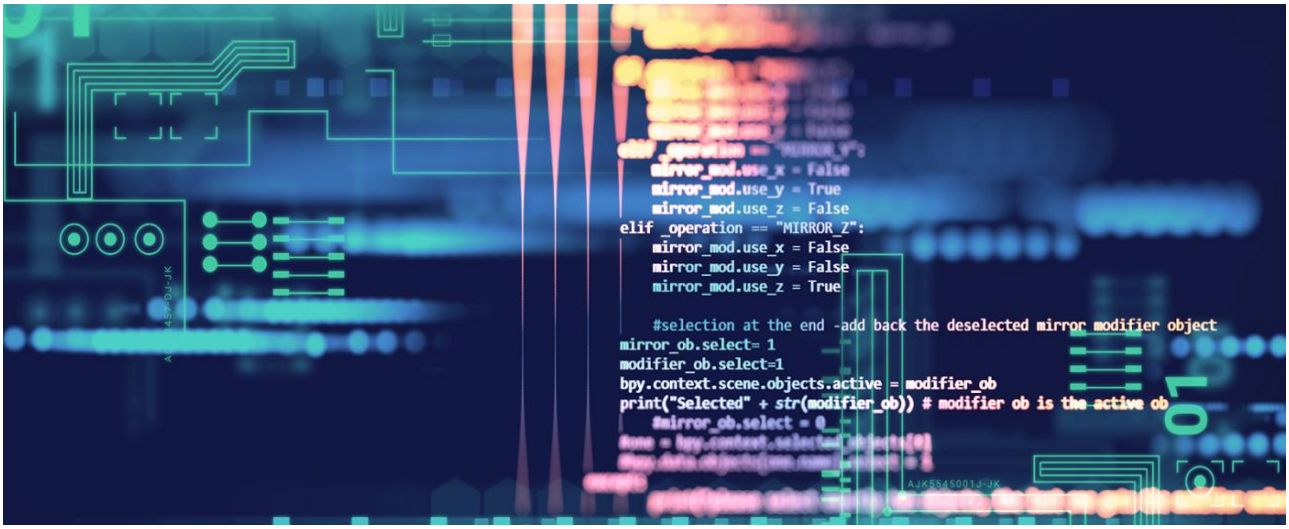


Figure 3-19 Schematic of Master control board

Reference

[1] https://www.ieee.li/pdf/essay/dds.pdf
[2] https://en.wikipedia.org/wiki/Direct_digital_synthesizer
[3] http://www.analog.com/media/en/training-seminars/tutorials/MT-049.pdf
[4] https://www.k-state.edu/edl/docs/pubs/technical-resources/Technote3.pdf
[5] http://www.ti.com/lit/an/slyt369/slyt369.pdf
[6] http://www.ti.com/lit/an/slyt369/slyt369.pdf
[7] http://cds.linear.com/docs/en/datasheet/626810f.pdf
[8] http://www.ti.com/lit/ds/symlink/ads1256.pdf
[9]http://www.analog.com/en/analog-dialogue/articles/dds-generates-high-quality-waveforms-efficiently.html



Chapter 4

Software design

In this chapter, the software design of lock-in amplifier is presented. The flow chart of system design is given at first. Then, the modular programming is divided into six parts: Initialization, interrupt DDS interface, A/D interface, D/A interface and MATLAB GUI. Finally, a summary of this section will be provided.

4.1 System software design

The work of the lock-in amplifier not only requires the normal operation of the hardware circuit, but also the design of the system software directly affects the final performance of the lock-in amplifier. In this paper, the Keil5.0 software as a main software development platform of the microcontroller (STM32), all of the system control parts of the program are completed on this development platform, which is simple and easy to complete the ARM system software development. Then the data is sent, collected and processed through the host computer (MATLAB GUI). The overall design flow chart for the lower computer (STM32) is shown in Figure 5-1.

The working process of the main program software is: After the initialization of the system program, waiting for the command word generation, then select the module according to different control words. For example, if the AD9959 is selected, its system frequency and VCO status can be set at first, and then frequency, amplitude, and phase of each channel can be adjusted by MATLAB GUI.

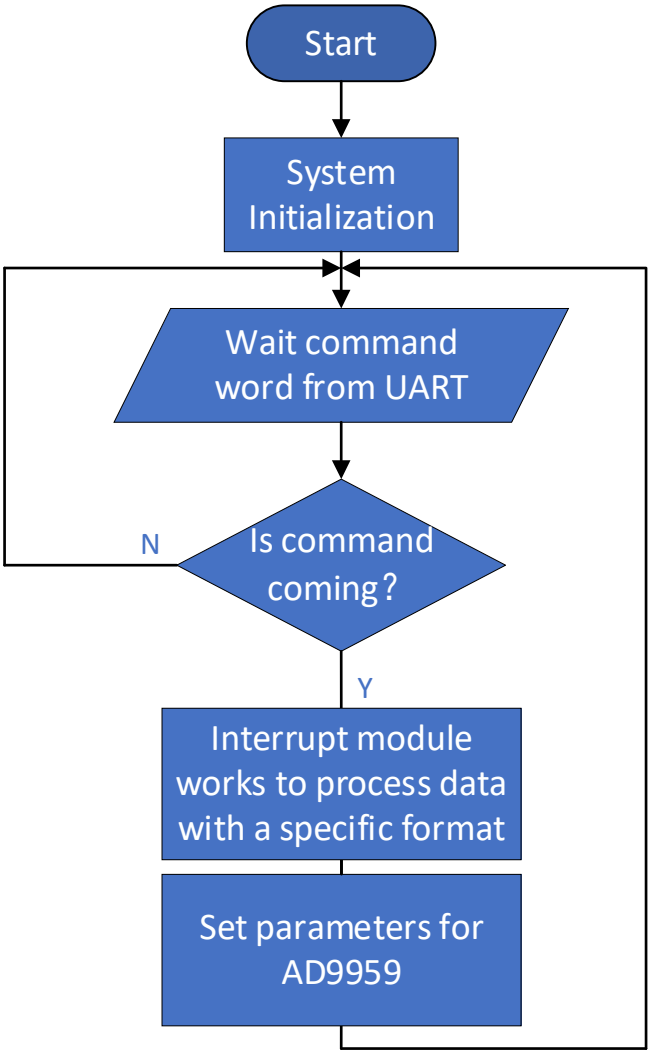


Figure 4-1 Overall software flow chart of the system

4.2 Modular programming

For the complex program, separating program modules are easier to write and handle.

4.2.1 Initialization module

The initialization process is mainly a set of the initial state of the entire software, including pin configuration and initialization of the relevant GPIO port, the system clock initialization, serial port initialization, DDS initialization, A/D initialization, interrupt program initialization and so on. The process of initializing the program is correct or not directly related to the correctness of the next functional program.

4.2.2 Interrupt module

For this system, serial port transfers data from MATLAB to MCU, so USART interrupt is needed, by writing a receive interrupt program in specific interrupt function (USART1_IRQHandler) with a specific data format ('&' '@' 'channel number' '@' 'frequency' '*' 'phase' '*' 'amplitude' '#'), when MATLAB send any data, MCU will pause the current work and enter the interrupt program, then

process the data transferred by MATLAB and complete the corresponding instructions according to the specific data format, after MCU finishes the work from an interrupt program, it will continue to complete previous work.

4.2.3 A/D interface module

The driving process of A/D converter mainly depends on STM32 SPI (Serial Peripheral Interface) interface, it is a synchronous, full duplex serial interface, there are two completely independent SPI controllers, the maximum data transfer rate is 1/8 of the clock rate, which can be configured as master or slave according to the direction of data transfer. The SPI interface timing diagram is shown in Figure 4-2, where CPOL is the polarity of the clock signal SS is the slave select and CPHA is the clock phase. For example, if CPOL=0, the base value of the clock is zero. For CPHA=0, data are captured on the clock's rising edge and data are propagated on a falling edge.

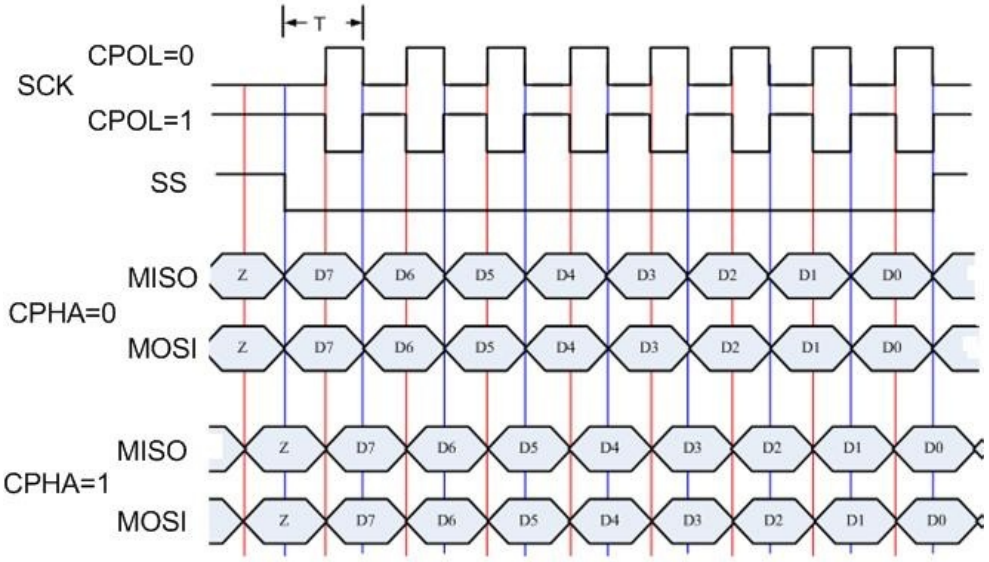


Figure 4-2 SPI timing diagram

In the preparation of the driver ADC, it requires a combination of the chip operation timing and the SPI operation timing of MCU to complete. The main work process is as follows: connect the pin to the corresponding SPI interface and initialize it, then set the master (STM32) and slave (ADS1256), write a function to send and receive one byte of data through the SPI and call that function in the main function of AD sampling. When making a call, the chip select is required firstly, then send the appropriate SPI commands to set the register to complete the appropriate function, and finally, start the AD sampling process.

4.2.4 DDS interface module

Similar to ADC, AD9959 register configuration is written through the SPI port, including channel selection, channel control, frequency control word, phase control word, and the amplitude control word, after writing these control words, a rising edge of UPDATE is given to make AD9959 work and get the output signal.

The four channels of the AD9959 share a single set of register addresses. This address sharing mechanism allows them to write the same data to all configuration registers of four channels

simultaneously. When it is necessary to set four channels differently, the data for each channel can be written independently by setting the channel to enable bits.
So, the main work process is:
1) Send a DDS reset signal, so that the internal registers of AD9959 could be the initial state.
2) Set system frequency (reference frequency times PLL multiplication factor from 4 to 20) as 500MHz.
3) Channel 0 enable bit is set, all other channels enable bits are set to 0.
4) The serial I/O port is used to send the frequency control word, phase control word and amplitude word required by channel 0.
5) The channel 1 enable bit set to 1, and other channels enable bits are set to 0.
6) The serial I/O port is used to send the frequency control word, phase control word and amplitude word required by channel.
7) Similarly, send channel 2 and channel 3 frequency control word, phase control word, and amplitude word.
8) Send I/O UPDATE signal, enable AD9959 output.

4.2.5 MATLAB GUI

There are many ways to receive data on PC. VC ++ should be typical but considering that MATLAB is more suitable for signal processing, MATLAB serial communication is used. For MCU, the work of STM32 includes clock, interrupt, IO port, serial port, AD, DA and DDS initialization, then start the conversion and send data to serial port. It is important to note that when sends two consecutive data to the serial port (such as high 8bits and low 8bits of 16-bit data), the transmission completion flag (TC) cannot be queried when the first data is completely sent. When checking the transmission completion flag, the second data will overwrite the first one. The solution is to check the flag of the transmit data register empty (TXE).
When using MATLAB to receive serial data, serial port objects (includes the baud rate, stop bits, parity way, input and output buffer size) need to create and initialize, then open the serial port, call the callback function when a specific serial communication event occurs and close the serial port in the main function at last.
The main work process is shown in Figure 4-3, which aims at calculating the capacitor value of sensors, it has five steps:

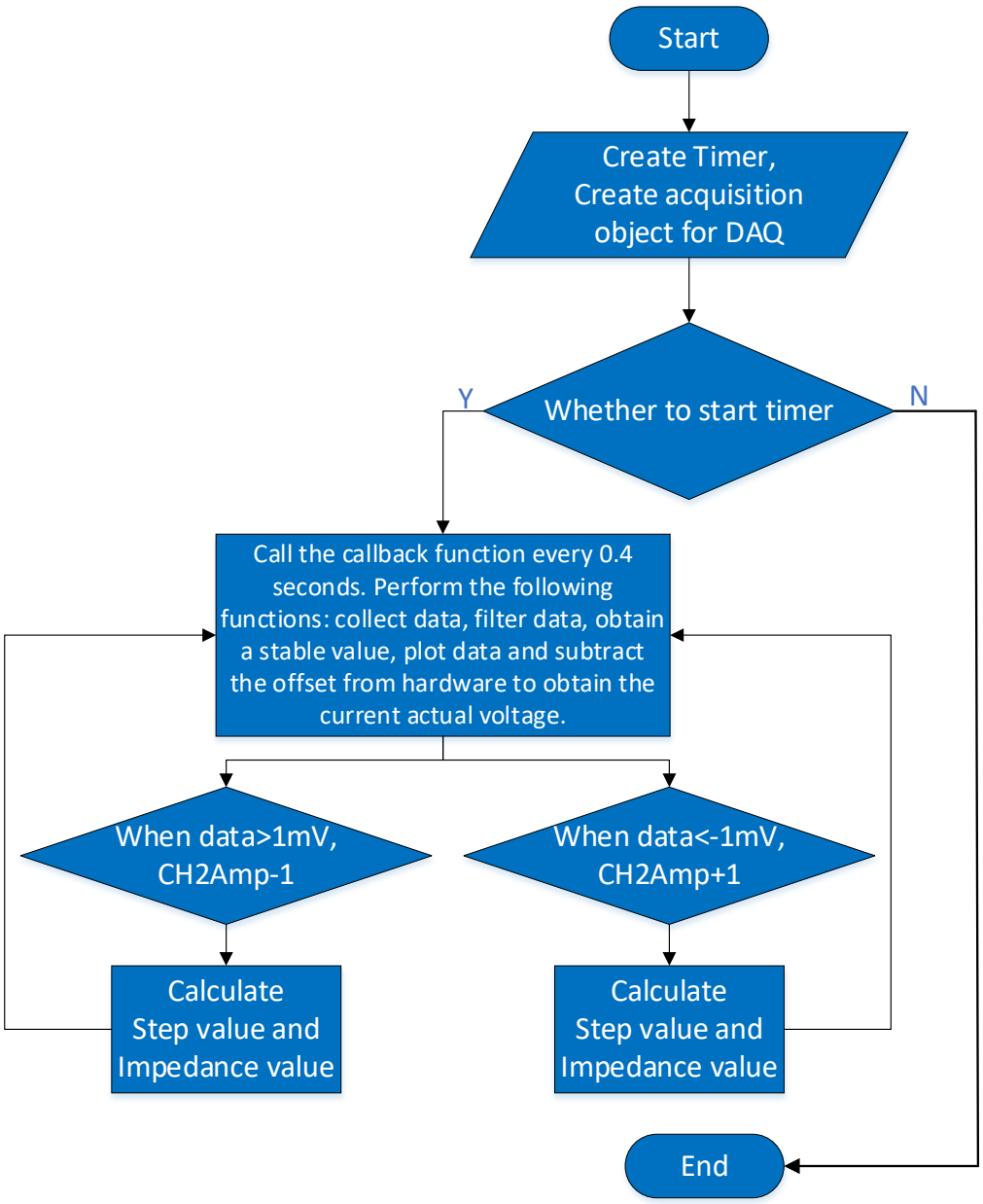


Figure 4-3 Flow chart of MATLAB GUI

Step 1. Receive OUTPUT data, preprocess the data with smooth and average function in MATLAB.

Step 2. Plot the output data in the coordinate system in real time and determines whether the output data satisfies -1mV ≤ output data ≤ 1mV (in order to judge the system is stable or not). If the system is stable, using the following formula to calculate Step Value (Changes of output when input changes 1bit), Capacitor Value (using the interpolation way) and Resistor Value, then display their value in the GUI interface (C_x is shown as C and R_x is shown as R in Figure 4-4), otherwise go to Step 3.

$$C_x = -\frac{CH0Amp}{CH2Amp} \times C_{REF} \times cos[phase(CH2) - phase(CH0)] \qquad (4-1)$$

$$-\frac{OUTPUT(Stable)}{Step\ Value} \times \frac{1}{1024} \times C_{REF}$$

$$Step\ Value = \frac{(OUTPUT(CH2Amp+4) - OUTPUT(CH2Amp+1))}{3} \qquad (4-2)$$

$$R_x = \frac{CH0Amp}{CH2Amp} \times \frac{1}{2 \times \pi \times f \times C_{REF} \times sin[phase(CH2) - phase(CH0)]} \qquad (4-3)$$

Where CH0Amp and CH2Amp are amplitudes of channel 0 and channel 2 of DDS respectively, C_{REF} is the reference capacitor, C_{REF} *(output/step value)/1024 is interpolated to achieve the required very high resolution of capacitor value.

Step 3. If the output data is bigger than 1mV, set CH2Amp = CH2Amp – 1 then save CH2Amp value, calculate Capacitor Value with the following formula and show it in the GUI interface (C_x is shown as C_{fix} in Figure 4-4), otherwise enter Step 4.

$$C_x = -\frac{CH0Amp}{CH2Amp} \times C_{REF} \times cos[phase(CH2) - phase(CH0)] \qquad (4-4)$$

Step 4. If the output data is smaller than -1mV, set CH2Amp = CH2Amp + 1 and do the same thing as Step 3.

Step 5. Repeat the above steps in the Timer.

As shown in Figure 4-4, the MATLAB GUI is presented, which consists of 6 parts: Communication, Work clock, Channel, OUTPUT, Step Value and Impedance Value. Communication part has the selection of cluster communication port (COM), the baud rate, open and close serial port; reference clock, PLL and VCO of DDS are included in work clock part, which can set system frequency; 4 channels of DDS with frequency-tunable, phase-tunable and amplitude-tunable are shown in Channel part; data of the lock-in amplifier output sends to OUTPUT part; and Step Value part shows the 1 bit output value in real-time, which is ready for calculation of interpolation; final values of capacitor and resistor is shown in Impedance part.

Click the Open button to connect STM32, then click Load CH0, Load CH1, Load CH2 and Load CH3 to send information of frequency, phase and amplitude to STM32 and AD9959 respectively, after Start button, data processing program is working and Impedance value can be obtained from GUI.

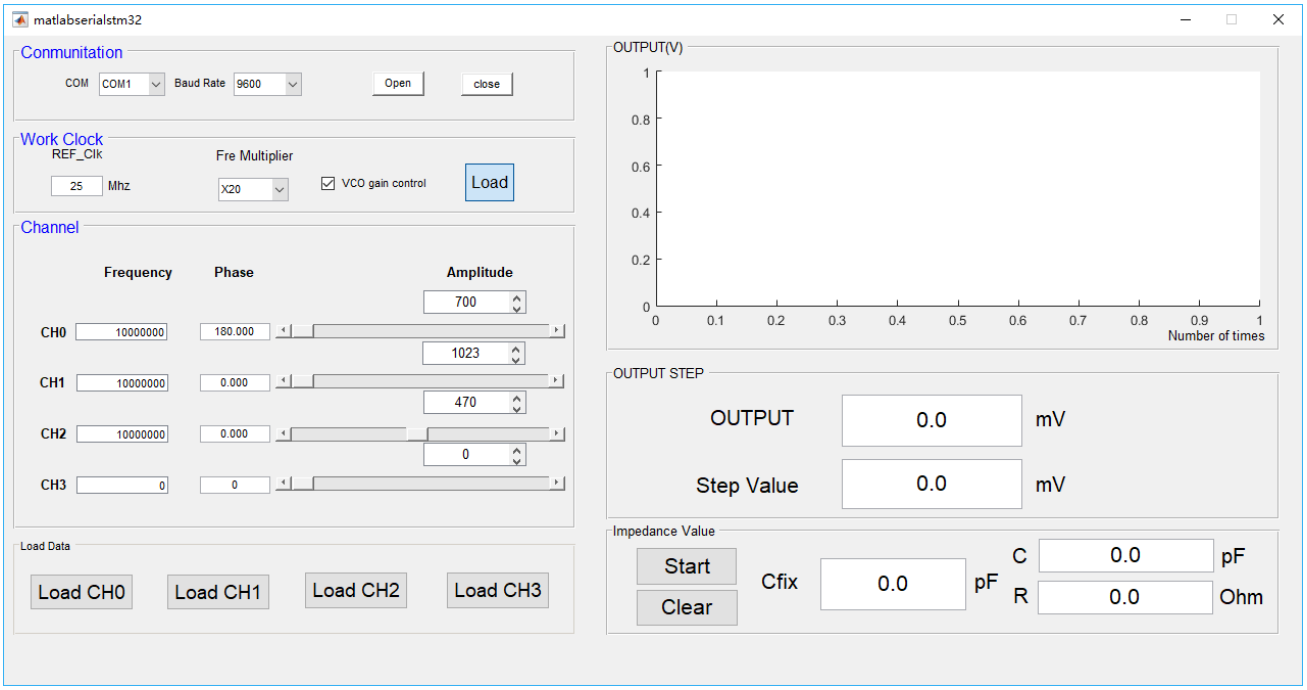


Figure 4-4 Structure of MATLAB GUI

4.3 Conclusion

In this chapter, the whole structure of the program is presented, a flow chart of the overall software system is shown at first, then it is divided into six sections: initialization, interrupt, A/D interface, DDS and MATLAB GUI. Each section has a briefly introduce with working principle, a basic idea and operation of MATLAB GUI are introduced as well.

Reference

[1] http://www.ti.com/lit/ds/symlink/tlc5615.pdf

Chapter 5
Measurement Results

In this chapter, measurement results of the whole system are presented. The measurement setup, including all PCBs, power and measurement equipment are presented first. Then, a hardware test has been done for each part and the characteristics of them are shown. After that, the measurement results obtained using both AH2700 and the realized system are presented. Finally, a summary of this chapter is provided.

5.1 Measurement Setup

In order to start measurement, building the measurement setup is necessary. An overview of the equipment setup is depicted in Figure 5-1.

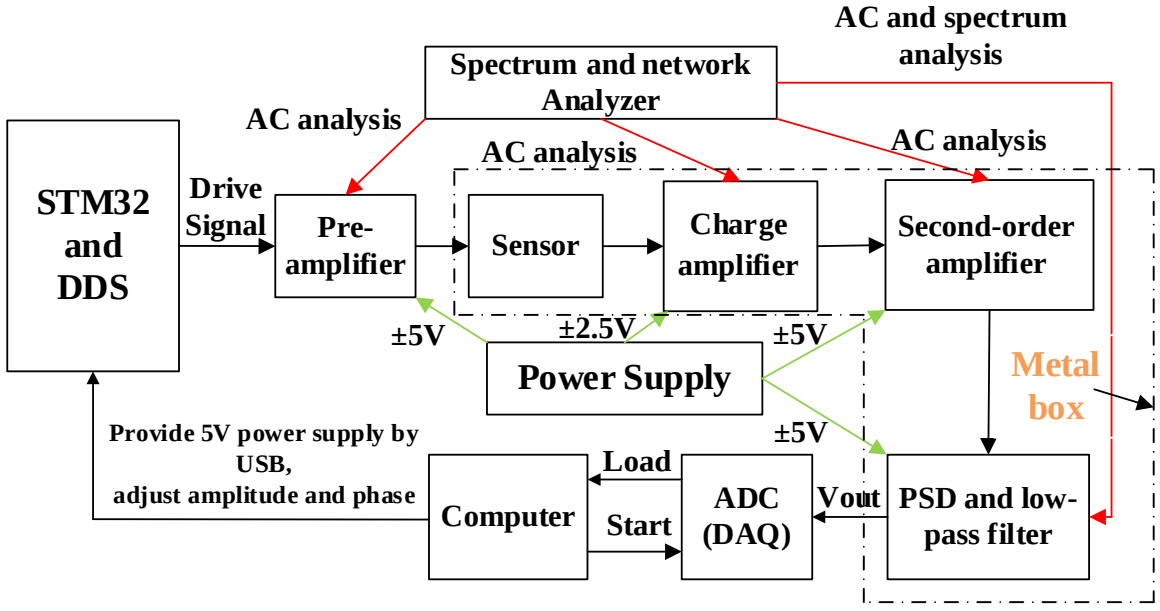


Figure 5-1 Overview of the measurement equipment setup

In this measurement, some equipment has been used:

1. Power Supply

It provides ±5V and Ground (GND) for pre-amplifier, Second-order amplifier, PSD and low-pass filter, ±2.5V and Ground (GND) for charge amplifier as well.

2. Spectrum/Network Analyzer

The Spectrum/Network analyzer measures both amplitude and phase properties, which can provide useful information about transfer function, gain, -3dB frequency and phase margin of each amplifier. It can also detect the magnitude of an input signal versus frequency within the full frequency range of the instrument [1], which can check the output signal of the mixer.

3. Metal box

A metal box is used for shielding of the PCB during measurement, in order to eliminate the external environment interference.

4. STM32 (MCU) and DDS

The MCU provides the control signal for DDS, and DDS can generate a sine wave signal to drive the whole hardware system.

5. Data acquisition board (DAQ)

The data acquisition board reads out the results from the hardware (the final output signal comes out from the low-pass filter) and transfers them to the PC as feedback.

6. PC

The PC controls the DAQ and processes the measurement results to calculate the capacitor and resistor values.

A photo including all the measurement equipment is shown in Figure 6-2 and Figure 6-3.

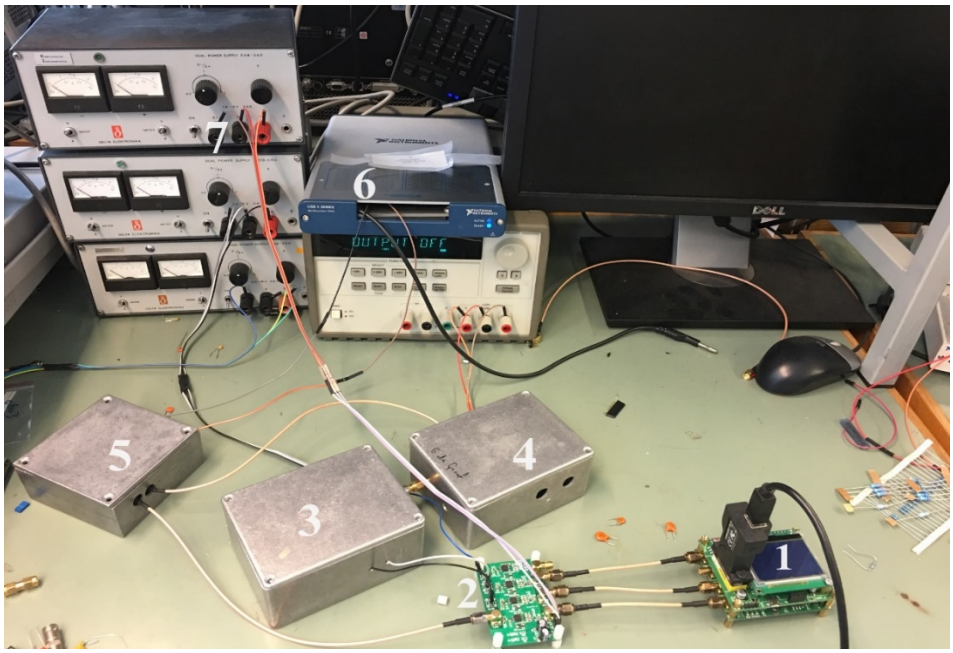
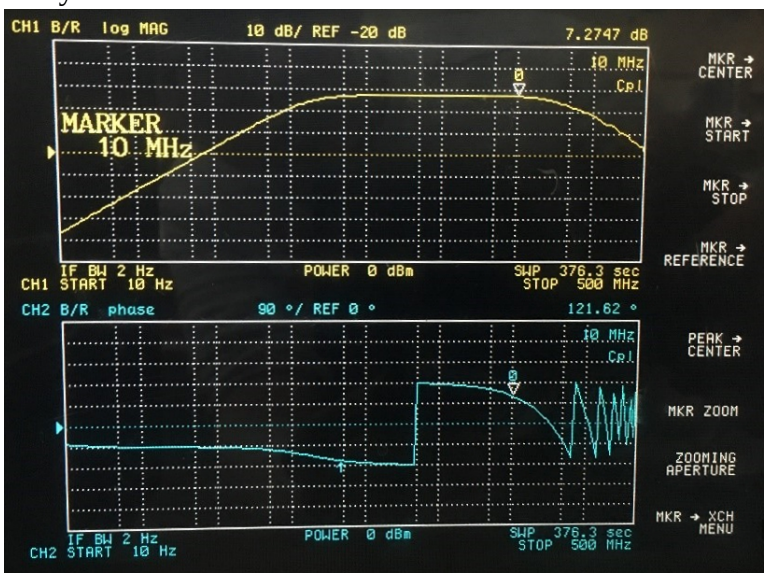


Figure 5-2 A photo of the measurement equipment. 1: MCU and DDS; 2: pre-amplifier; 3: capacitor bridge and charge amplifier in a metal box; 4: second-order amplifier in a metal box; 5: PSD and LPF in a metal box; 6: data acquisition board; 7: power supply.

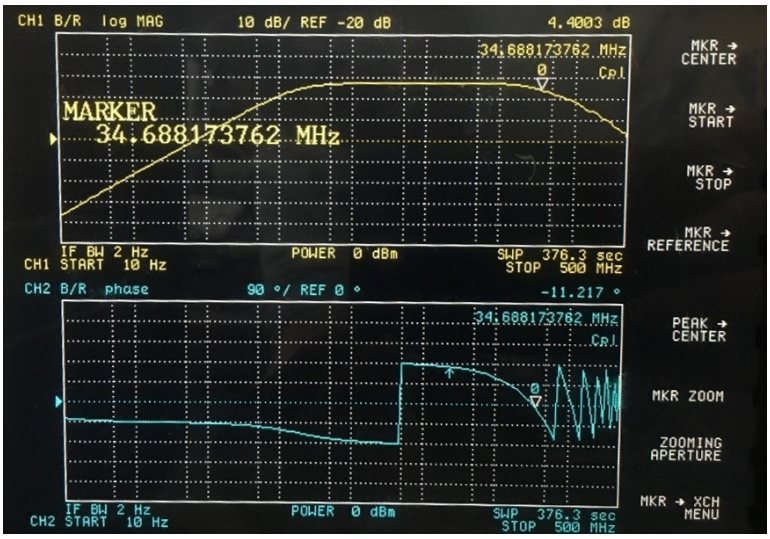
Equipment number 1 to 7 is used for the closed loop impedance measurement (in Figure 5-2), which form a complete detect system; while network analyzer is applied for the spectrum measurement and AC analysis (gain and phase) (section 5.2).

5.2 Hardware Test

Four circuits need to be tested: pre-amplifier, charge amplifier, low-noise amplifier, and mixer.
As shown in Figure 5-3, it is the analysis of pre-amplifier, because of the power splitter, this signal has some loss. When the frequency is 10MHz, input resistor is 1K Ohm and feedback resistor is 2.4K Ohm, the ideal gain can be calculated, which is 7.60dB, the real gain can be obtained from Figure 5-3(a) is 7.27dB. Also, the cut-off frequency is about 34.69MHz. Because there is a 10nF input capacitor, it has some attenuation at low frequencies. Compared with the simulation results, due to the ideal components in simulation software, the loss and some tolerance from resistors in a real situation, this gain and cut-off frequency are reasonable.



(a) Gain @10MHz

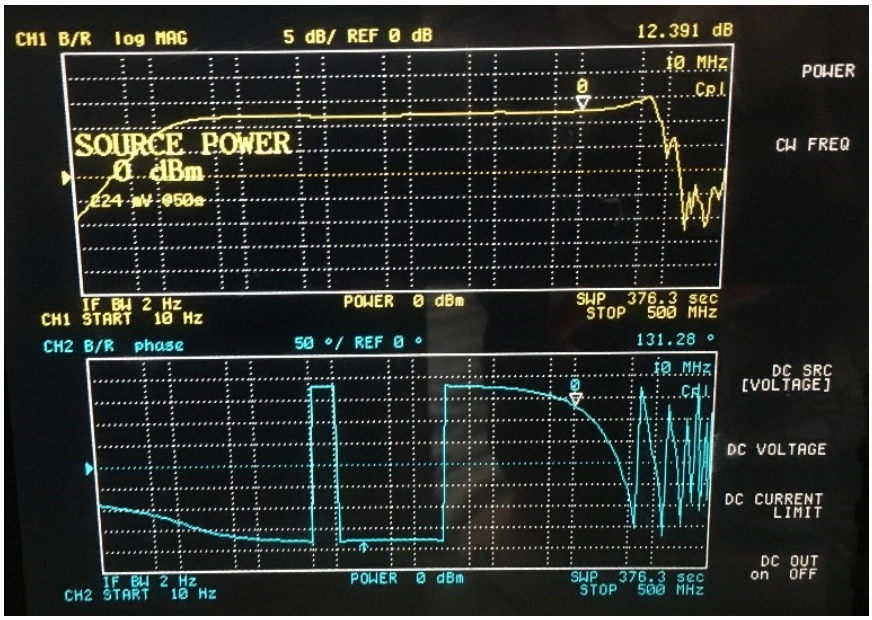


(d) -3dB bandwidth

Figure 5-3 Bode plot of pre-amplifier

For charge amplifier, 10pF and 22pF are chosen as input capacitances to do the test, as shown in Figure 5-4, when the frequency is 10MHz, input capacitor is 22pF, feedback capacitor is 5pF and feedback resistor is 250MΩ, the ideal gain is 12.87dB. Because of loss and some tolerance from the capacitor, the real gain is 12.39dB; there is a peak at 59.81MHz due to the characteristic of the chip itself (LTC6268-10). The same situation, when the input capacitor is 10pF, the ideal gain is 6.02dB, the real gain can be obtained from Figure 5-4(c) is 5.714dB, and the peak at about 61.36MHz.

Compared with simulation results, there is a considerable difference in the bandwidth and peak frequency, after using the new board and replacing a new chip, the bandwidth is the same as shown in Figure 5-4. This might be due to the model of the opamp is not updated, or some components are ideal in the software. However, this is not a problem for our system because the maximum input frequency is 10MHz.



(a) Gain @10MHz when input capacitor is 22pF

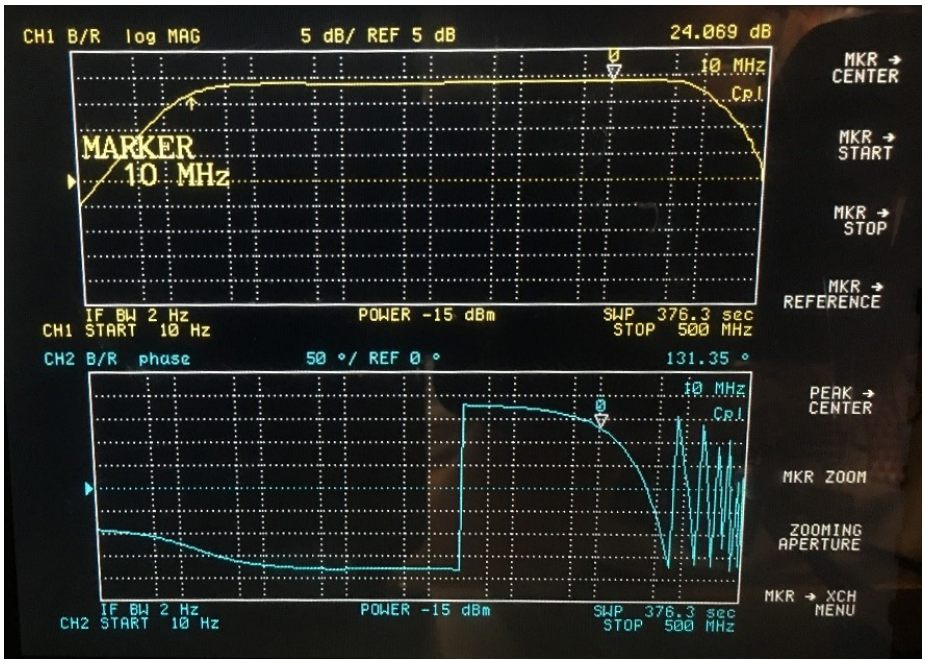
(b) Frequency and gain of peak when input capacitor is 22pF

(c) Gain @10MHz when input capacitor is 10pF

(d) Frequency and gain of the peak when the input capacitor is 10pF

Figure 5-4 Bode plot of the charge amplifier

For a high-gain amplifier, the ideal gain is 26.02dB when the frequency is 10MHz, input resistor is 50 Ohm and the feedback resistor is 1K Ohm. From Figure 5-5, because of the power splitter, the input signal has some loss, it can be seen that the real gain is 24.07dB and -3dB frequency is about 113MHz. Compared with the simulation results, there is a considerable difference in the bandwidth, after using the new board and replacing a new chip, the bandwidth is the same as shown in Figure 5-5. This might be due to the model of the opamp is not updated in the software. However, this is not a problem for our system because the maximum input frequency is 10MHz.

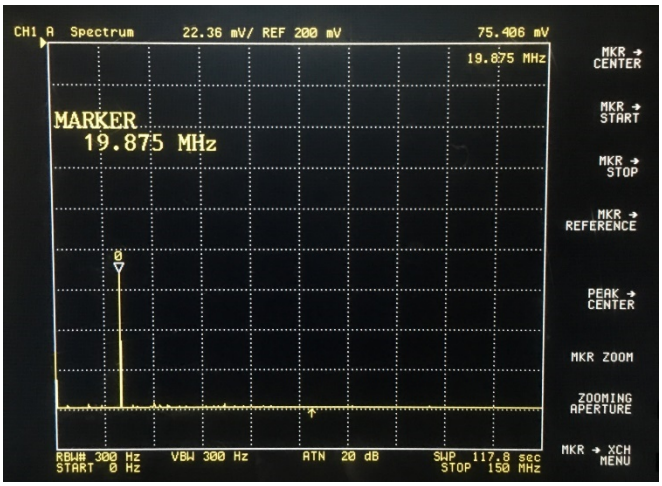


(a) Gain @10MHz

(b) -3dB bandwidth

Figure 5-5 Bode plot of high-gain amplifier

The spectrum analysis of mixer is shown in Figure 5-6 when the frequency of input signal and the reference signal are 10MHz. It has a quite nice output (75.41mV) at 20MHz, while the remaining harmonics have normal attenuation. Meanwhile, after measurement, since the mixer has an offset (about 24mV), this offset is removed by software method, the DC output (-80.29mV) of the mixer can be obtained on GUI, it can be seen that the mixer is working properly and meets the expectations of the design.



(a) Spectrum analysis of mixer

Figure 5-6 Spectrum analysis of mixer when input frequency is 10MHz

From the above measurement, it can be seen that the whole hardware system is working properly up to 10MHz, but the input voltage range needs to be considered to avoid harmonic distortion. Meanwhile, the peak of the charge amplifier also needs to be considered in the future; this peak may affect the stability of the system when it works at 10MHz frequency, replacing this chip with higher bandwidth chips or picking similar chips without a peak.

5.3 System Result

As shown in Figure 5-7, in order to get resolution and stability of this closed-loop system, a simple test impedance bridge is needed. Simulate a capacitive sensor by using two ultra-stable over temperature and voltage capacitors.

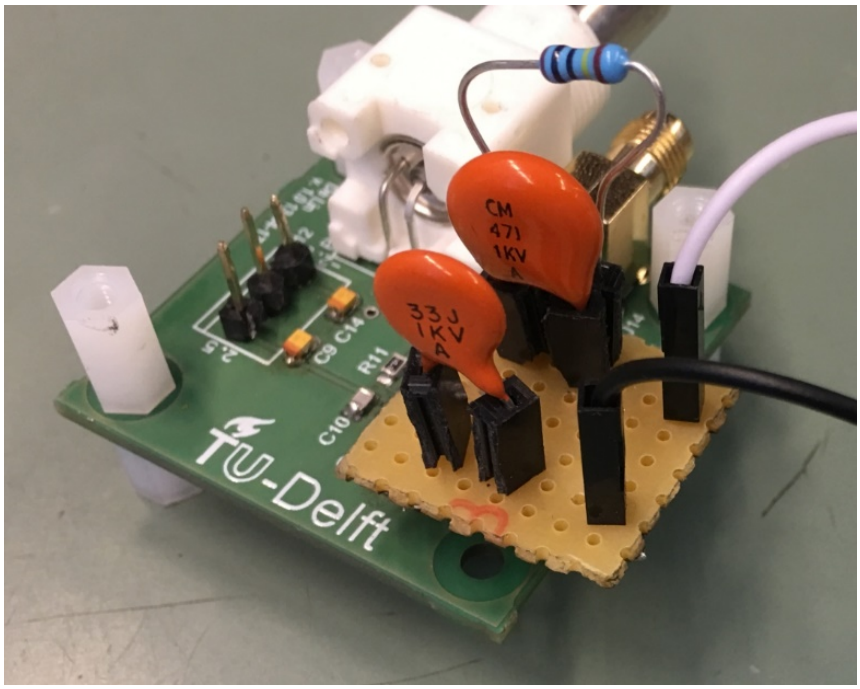


Figure 5-7 Test impedance bridge

In this test, five different value capacitors are selected, and 33pF is used as a reference capacitor. Because these capacitors have ±5% tolerance, get relative accurate capacitance values are essential,

which can reduce certain value errors for data processing. Table 5-1 shows values (different frequency, external environment, and pin distance can cause a few changes of value) of the capacitor by using the Andeen-Hagerling 2700A Ultra-precision Capacitance Bridge at 1000Hz frequency.

Table 5-1 Capacitor value

Value (pF)
22.15168
28.78325
32.25574 (Reference)
33.27435
46.41213
48.23594

When starting the system first, detect the output and subtract output offset (minus an average of a thousand data of output without input connection) is needed, a 48.23594pF test capacitor is used, then click the start button on GUI, the specific DDS parameters and capacitor value are shown in Figure 5-8.

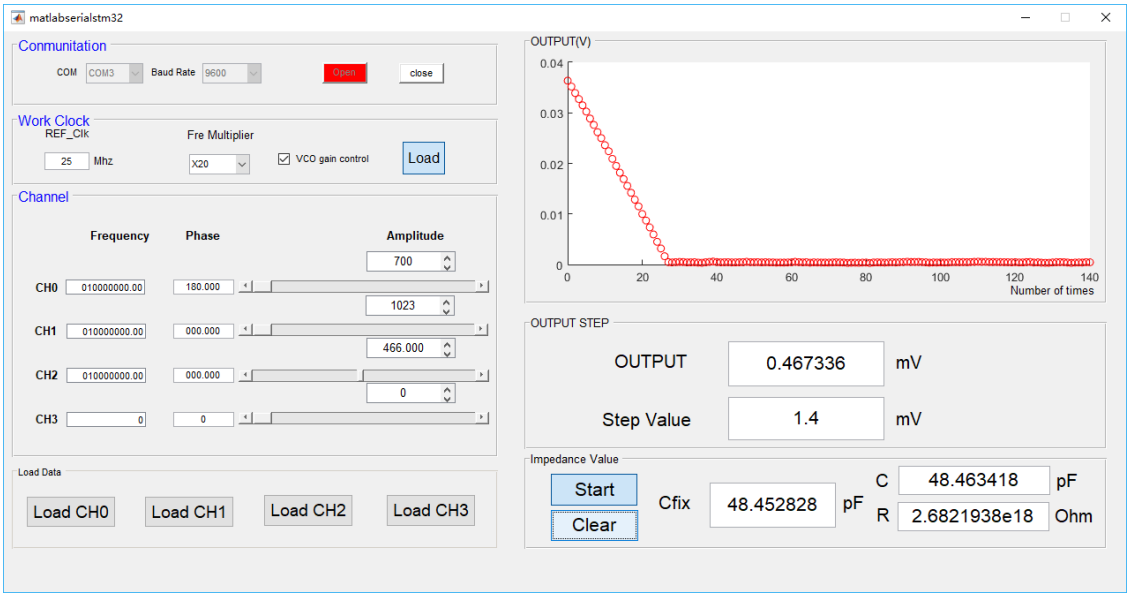
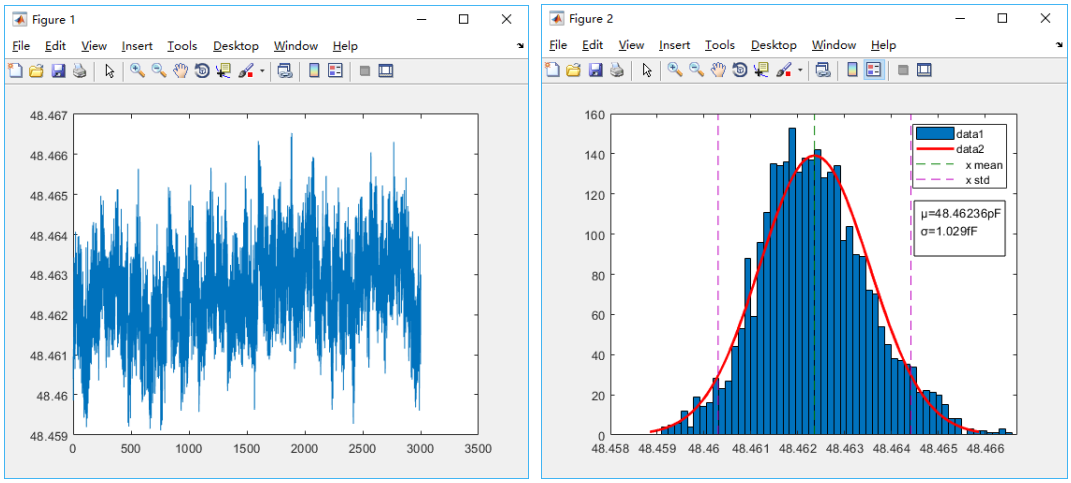


Figure 5-8 Result of the system when C_{REF} is 32.25574pF

It can be seen that signal frequency is 10MHz, CH0 provides a drive signal for reference capacitor, CH2 provides a drive signal for the unknown capacitor and CH1 provide a drive signal for PSD (mixer) as a reference signal. From section 3.1.3, in order to make I_{IN} = 0, they need a 180-degree phase shift. However, in real situation, wire has a very small resistance, it can cause a phase shift between two different capacitor value, by using the following formula, when R_{Wire} = 0.1Ohm, C_{REF} = 32.25774pF and Cx = 48.23594pF, it can be calculated the phase shift of them, where φ_{REF} = -89.988˚ and φ_X = -89.983˚, because AD9959 only has 14bit resolution (0.022˚) of phase, so this phase shift between capacitors can be ignored here.

$$\varphi = \arctan\left(-\frac{1}{\omega RC}\right) \qquad (6-1)$$

With more output data, some noise might be averaged out, so for each unknown capacitor, 3000 capacitor values have been recorded at least. From Figure 5-9, it can be seen that the distribution of capacitance values shows a Gaussian distribution, which means that the average of these values is a fixed value (The average of thermal noise is 0). Figure 5-9(b) shows the Gaussian fit curve (red curve), μ and σ are mean value and standard deviation value respectively, they can be considered the ideal result of this measurement. Therefore, it is possible to calculate the relative real capacitance value and effective noise (RMS noise, it serves as a standard to assess the resolution of this measurement process) of the capacitor.



(a) (b)

Figure 5-9 (a). 3000 data of 48.23594pF; (b). Distribution of 48.23594pF

Table 5-2 shows the characteristic of the system when C_{REF} = 32.25574pF and C_X = 48.23594pF, according to interpolate the remaining output signal and change it into capacitor value, which can improve the resolution of the system (Better than 10-bit input). So, with some calculations, the RMS of noise is 1.17fF and capacitor value is 48.46236pF. Moreover, there are two different relative capacitor values between two devices, the main reasons are:

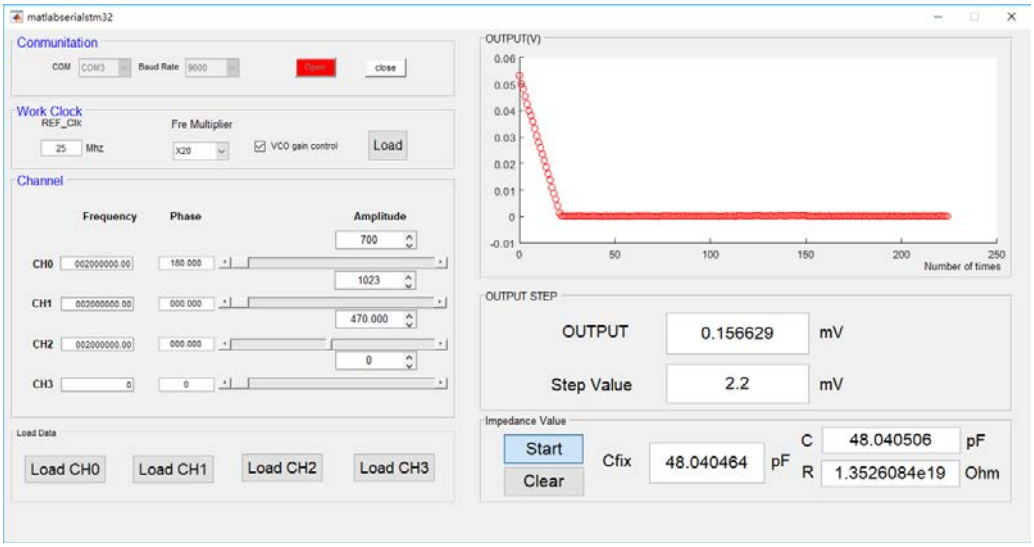
1. The simple test impedance bridge on breadboard provides extra parasitic capacitance.
2. The SMA connectors could provide additional interference.
3. Metal box (Shielding box) connects ground with the circuit or not also affects the measurement of the capacitance.

In order to decrease these effects and improve accuracy, next step can be built a new PCB with all these circuits and a test impedance bridge, try to use shielding cables as well, which can eliminate parasitic capacitance from cable and breadboard, then put it in a larger metal box to obtain a better shielding.

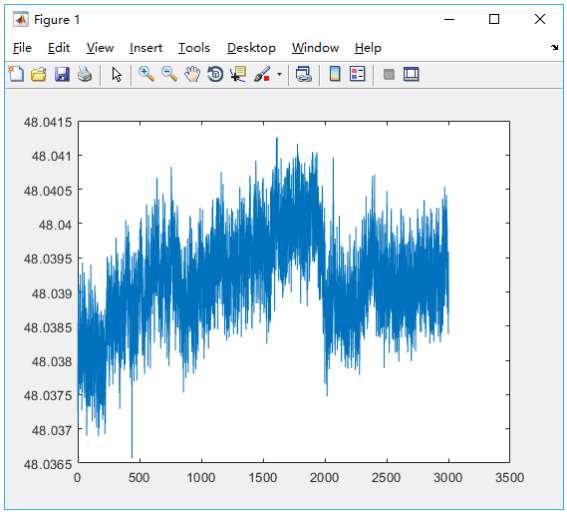
Table 5-2 Characteristic of the system when C_{REF} = 32.25574pF and C_X = 48.23594pF

C_X (AH2700)	Average of C_X	Error	Step value	RMSE of C_X	Resolution
48.23594pF ±0.25fF	48.46236pF ±1.17fF	0.47%	1.4mV	±1.17fF	±24.14ppm

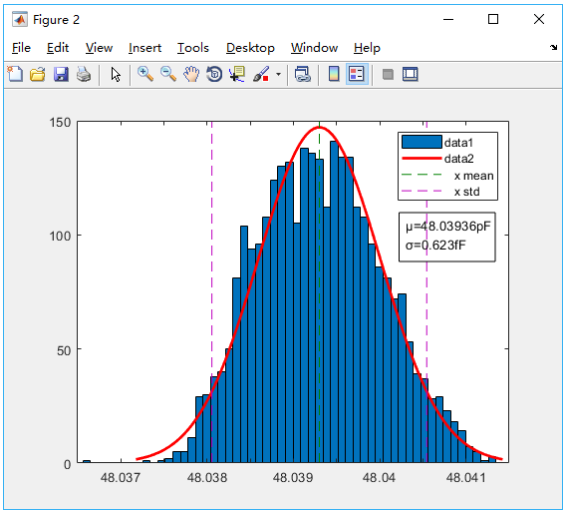
Furthermore, as a comparison, the 48.23594pF is tested again in the same situation when frequencies are 2MHz and 5MHz respectively, the basic set and capacitor value are shown in Figure 5-10.



(a) The result of 48.23594pF on GUI (2MHz)

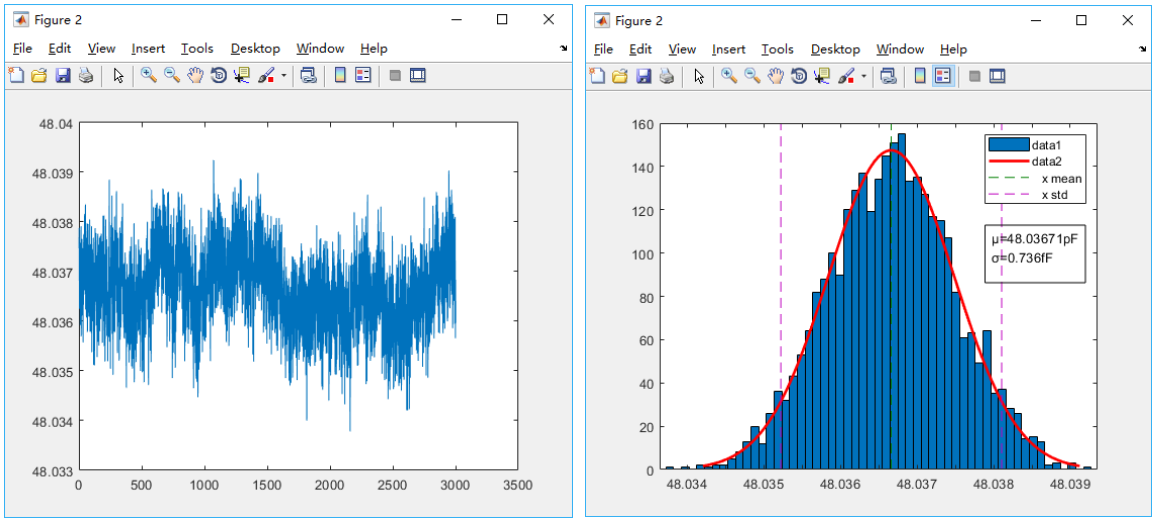


(b) 3000data of 48.23594pF @2MHz



(c) Distribution of 48.23594pF (2MHz)

(d) The result of 48.23594pF on GUI (5MHz)



(e) 3000 data of 48.23594pF @5MHz (f) Distribution of 48.23594pF (5MHz)

Figure 5-10 Results of 48.23594pF @2MHz and 5MHz

And the characteristics of the system when frequencies are 2MHz and 5MHz is depicted in Table 5-3.

Table 5-3 Characteristic of the system when Cx = 468.23594pF @2MHz and 5MHz

Frequency	Cx (AH2700)	Average of Cx	Error	Step value	RMSE of Cx	Resolution
2MHz	48.23594pF ±0.25fF	48.03936pF ±0.71fF	0.40%	2.2mV	±0.71fF	±14.74ppm
5MHz	48.23594pF ±0.25fF	48.03671pF ±0.81fF	0.41%	1.8mV	±0.81fF	±16.80ppm

From Table 5-2, Table 5-3 and Figure 5-11, it can be seen that when the frequency is decreased, the resolution has a certain increase, this is due to the characteristics of the DAC (the core component of DDS) [2]. In addition, due to the phase shift caused by the input resistance, the error of the capacitor has some changes at different frequencies, and the phase shift needs to be considered in the later work.

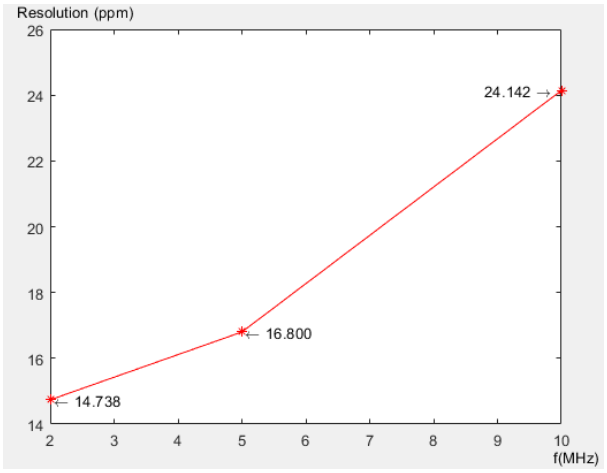
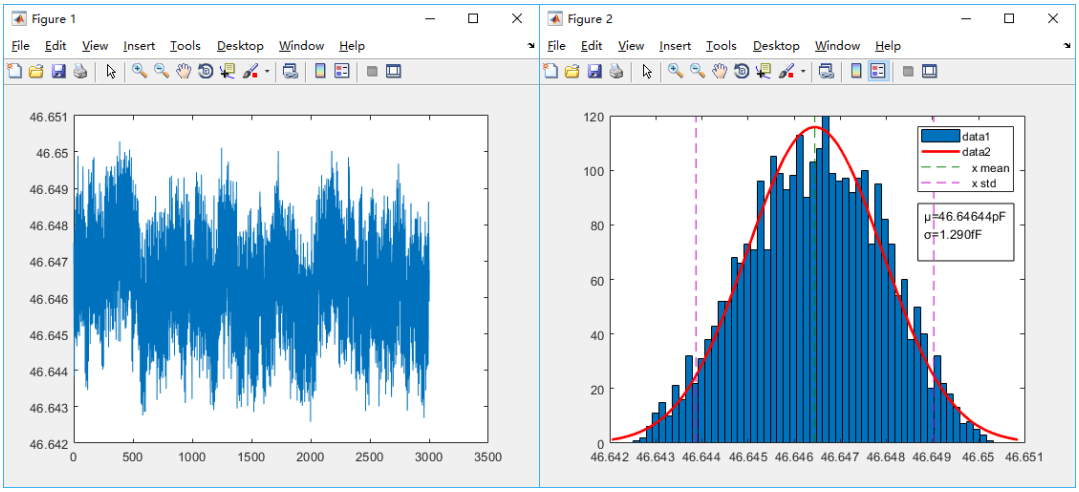


Figure 5-11 Resolution of the system with different frequency when Cx = 48.23594pF

Similarly, the distribution, relative real capacitor value and RMS of noise can be obtained for each capacitance at 10MHz.

For 46.41213 pF, the GUI, output data and distribution are shown in Figure 5-12, characteristic of the system when Cx = 46.41213 pF is depicted in Table 5-4.

(a)



(b) (c)

Figure 5-12 (a). The result of 46.41213pF on GUI; (b).3000 data of 46.41213pF; (c). Distribution of 46.41213pF

Table 5-4 Characteristic of the system when C_X = 46.41213pF

C_X (AH2700)	Average of C_X	Error	Step value	RMSE of C_X	Resolution
46.41213pF ±0.24fF	46.64645pF ±1.47fF	0.50%	1.3mV	±1.47fF	±31.45ppm

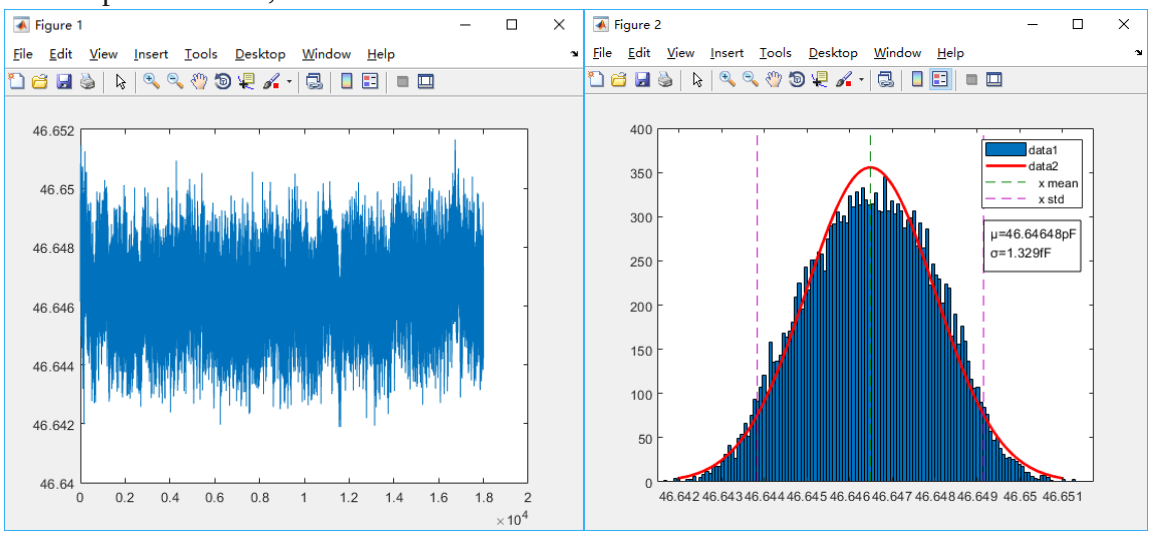
As shown in Figure 5-13, the stability of the whole circuits can be tested, let them run about 2 hours (18000 output data), then check the data on GUI and workspace, calculate RMS of noise and capacitor value and compare them with previous data.

Table 5-5 The results from two time periods

Average of C_X (18000)	Average of C_X (3000)	RMSE of C_X (18000)	RMSE of C_X (3000)	Resolution (18000)	Resolution (3000)
46.64648pF ±1.50fF	46.64645pF ±1.47fF	1.50fF	1.47fF	±32.11ppm	±31.45ppm

As you can see from Table 5-5, the output relative capacitor value and RMS of noise do not have too much error between two different time periods for 10-bit input amplitude. The stability of the whole

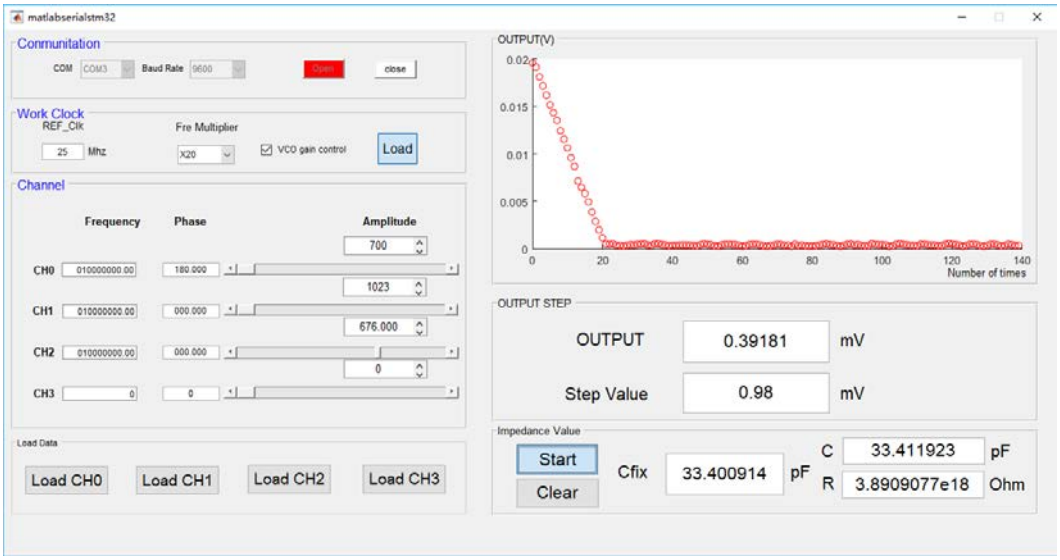
circuit is fine, but the external environment still provides some interference, which causes a little bit shift of the capacitor value, and the resolution has become even worse.



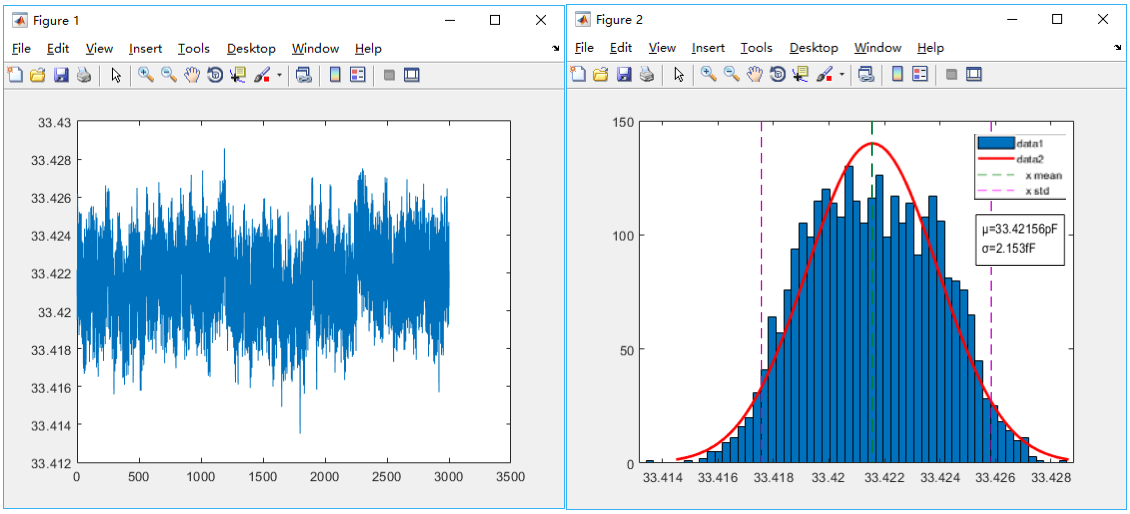
(a) (b)

Figure 5-13 (a).18000 data of 46.41213pF; (b). Distribution of 46.41213pF

For 33.27435 pF, the GUI, output data and distribution are shown in Figure 5-14, characteristic of the system when C_X = 33.27435 pF is shown in Table 5-6.



(a)



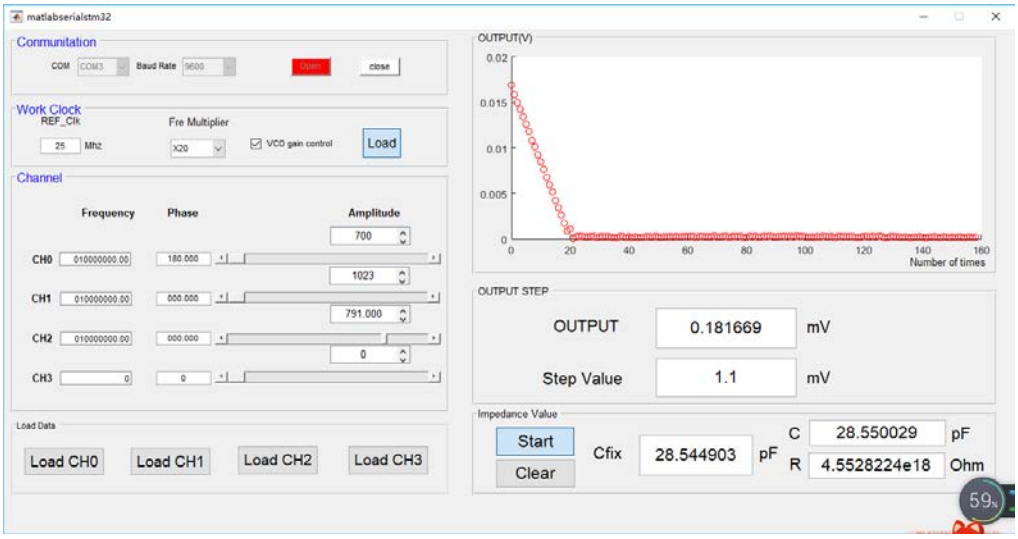
(b) (c)

Figure 5-14 (a). The result of 33.27435pF on GUI; (b).3000 data of 33.27435pF; (c). Distribution of 33.27435pF

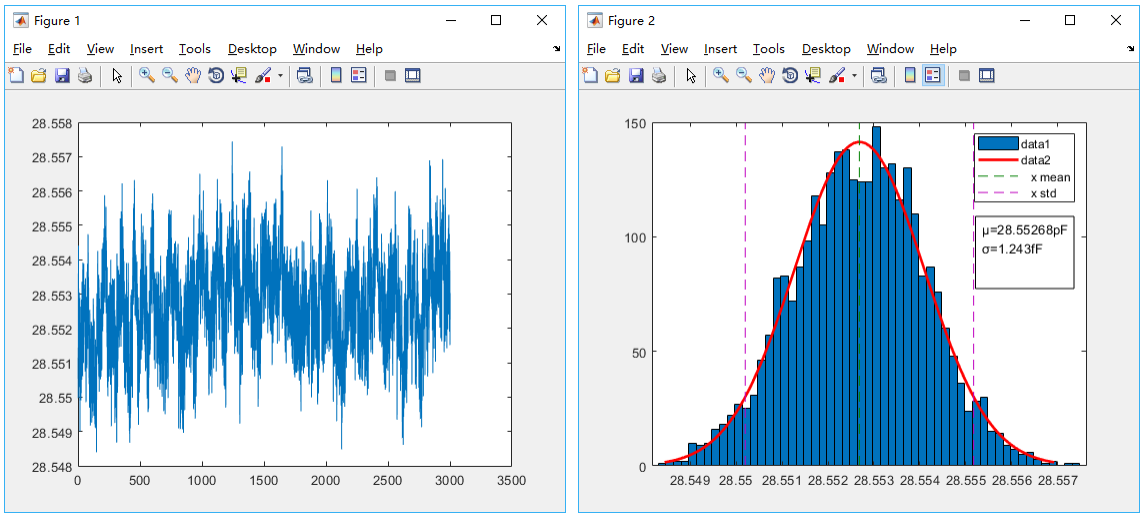
Table 5-6 Characteristic of the system when C_X = 33.27435pF

C_X (AH2700)	Average of C_X	Error	Step value	RMSE of C_X	Resolution
33.27435pF ±0.17fF	33.42156pF ±2.36fF	0.44%	0.98mV	±2.36fF	±70.55ppm

For 28.78325 pF, the GUI, output data and distribution are shown in Figure 5-15, characteristic of the system when C_X = 28.78325 pF is shown in Table 5-7.



(a)



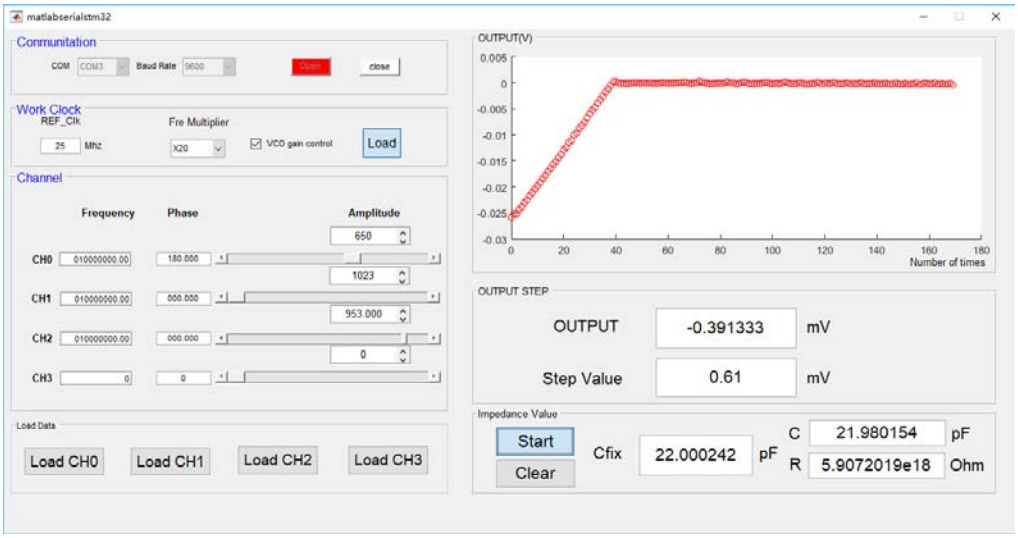
(b) (c)

Figure 5-15 (a). The result of 28.78325pF on GUI; (b).3000 data of 28.78325pF; (c). Distribution of 28.78325pF

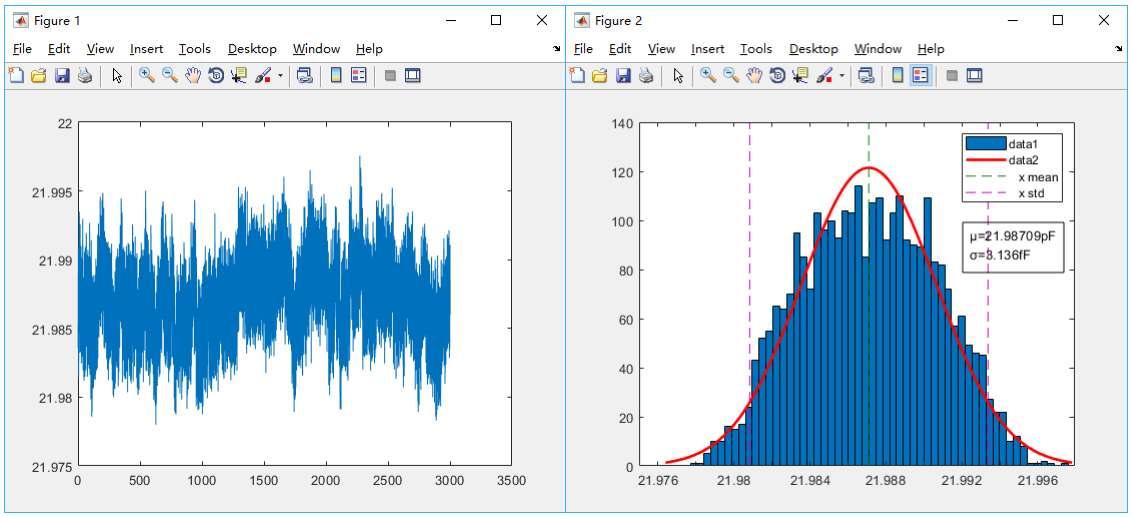
Table 5-7 Characteristic of the system when C_X = 28.78325pF

C_X (AH2700)	Average of C_X	Error	Step value	RMSE of C_X	Resolution
28.78325pF ±0.15fF	28.55268pF ±1.41fF	0.80%	1.1mV	±1.41fF	±49.52ppm

For 21.57155 pF, the GUI, output data and distribution are shown in Figure 5-16, characteristic of the system when C_X = 21.57155 pF is shown in Table 5-8.



(a)



(b) (c)

Figure 5-16 (a). The result of 22.15168pF on GUI; (b).3000 data of 22.15168pF; (c). Distribution of 22.15168pF

Table 5-8 Characteristic of the system when C_X = 22.15168pF

C_X (AH2700)	Average of C_X	Error	Step value	RMSE of C_X	Resolution
22.15168pF ±0.11fF	21.98710pF ±3.57fF	0.74%	0.61mV	±3.57fF	±162.28ppm

From Table 5-2 to Table 5-8, five capacitor values can be obtained; there are 0.2pF to 0.3pF error between this system measurement and AH2700A, the resolution of the final value between different capacitances is shown in Figure 5-17.

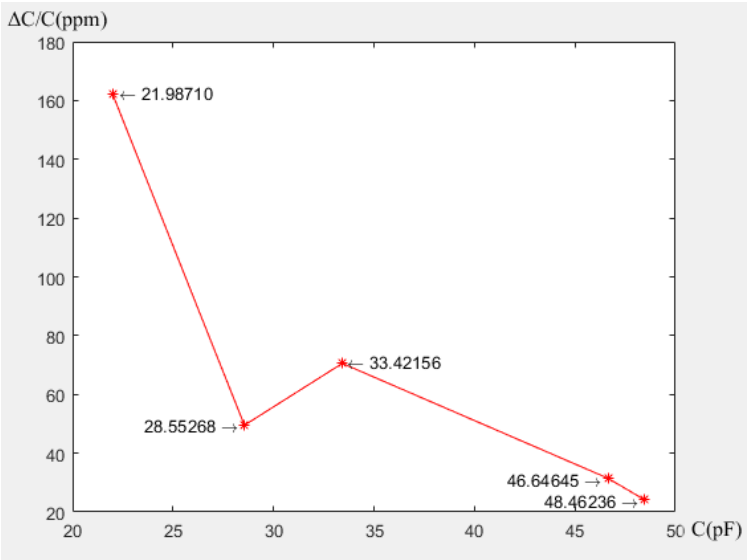


Figure 5-17 The resolution of the final value between different capacitances

Meanwhile, the resolution of this system can be calculated when 48.23594pF is used. From Table 5-2, the resolution is equal to 0.00117pF/48.46236pF when input frequency is 10MHz, which is ±24.14ppm @10Hz (parts per million). Compared with expectations (the impedance of capacitive sensor devices in the range of 22pF to 47pF with a 24ppm resolution @10Hz), the actual resolution is very close to this goal. However, it can be seen that the capacitance value becomes larger and the resolution is higher, but the resolution of 28.55268pF is better than that of 33.42156pF. This is because

the output signal of the impedance bridge is weak after the balance and the shielding method is not good enough, so it is easily interfered by the external environment (near people, mobile phones, etc.), the strength of the interference may have a certain impact on the resolution, which causes the above phenomenon. If the system can be tested in a more stable and less interference environment and improve the shielding way, the resolution might be better than 24ppm.

When parasitic resistance needs to be considered, we added a test impedance bridge could add some different resistors to simulate parasitic resistance and see what has changed in the output.

So, put 1M, 3.3M, 10M and 33M Ohm resistors in parallel with the unknown capacitance respectively (pick 48.23594pF as the unknown capacitor), the phase shift between parasitic resistance and the unknown capacitor can be obtained by calculation (Equation 4-3). However, because of the 14-bit phase resolution, 1-bit step has about 0.022˚, the phase of the GUI will be approximated to the corresponding 14-bit phase in AD9959, it can cause certain errors for resistance and capacitance values, the value of each resistance, phase shift and resolution of capacitance are tabulated in Table 5-9.

Table 5-9 Accuracy of resistance and resolution of the system with different

Actual value / Real value	1M Ohm	3.3M Ohm	10M Ohm	33M Ohm
Phase (Actual)	180.0189˚	180.0057˚	180.0019˚	180.0006˚
Phase (Real)	180.019˚	180.006˚	180.002˚	180.001˚
Resistance value	0.982M Ohm	3.116M Ohm	9.370M Ohm	18.820M Ohm
Accuracy	1.8%	5.6%	6.3%	43.0%
Resolution of capacitance	24.44	24.26	24.34	24.15

With the higher phase resolution, the higher accurate resistor can be obtained. Also, it can be seen that the resolution of capacitance has some shift, however, based on the noise analysis and simulation results of the charge amplifier, the noise provided by the parasitic resistance is very small, and the impact on the resolution of the system is minimal. When the resolution of the system is increased to aF or higher, the noise provided by the parasitic resistance limits the system resolution. The difference in resolution shows in Table 5-9 may be mainly caused by external interference.

5.4 Conclusion

In this chapter, the different capacitors are measured by two methods: self-balanced bridge based on LIA measurement and, for comparison, AH2700 capacitance bridge measurement. The results are summarized in Table 5-10.

Table 5-10 Summary of results

C value form AH2700	C value from this system
22.15168pF±0.11fF	21.98710pF±3.57fF
28.78325pF±0.15fF	28.55268pF±1.41fF
33.27435pF±0.17fF	33.42156pF±2.35fF
46.41213pF±0.24fF	46.64645pF±1.47fF
48.23594pF±0.25fF	48.46236pF±1.17fF

The resolution of this system still needs to be improved by better shielding and better test bridge, the resolution is about 24.14ppm @10Hz when input amplitude is only 10-bit as well as 10MHz input

frequency, the system also has ±14.74ppm and ±16.80ppm resolution @10Hz when input frequencies are 2MHz and 5MHz, and AH2700 has 0.16ppm resolution @1000Hz from 10pF to 100pF.
Also, the different parasitic resistances are measured, the results show that the phase resolution and resistance value can cause significant errors and effects on the system, which makes the accuracy of the capacitance value worse.

Reference

[1] https://en.wikipedia.org/wiki/Spectrum_analyzer
[2]http://www.analog.com/en/analog-dialogue/articles/analyzing-and-managing-the-impact-of-supply-noise-and-clock-jitter-on-high-speed-dac-phase-noise.html

Chapter 6
Conclusions and Future Work

6.1 Conclusions

Due to an increasing demand for the measurement of small changes in capacitances, it is clear that high resolution, high accuracy, high speed and very stable new readout circuits are required.

There are three main approaches for the measurement of the impedance value of capacitive sensors: switched-capacitor readout [1-4], ac-bridge with voltage amplifier [5,6,7] and transimpedance amplifier [8-11] based on lock-in amplifier measurement. In this work, because the system needs to work at higher frequencies and eliminate the effects of parasitic capacitance while increasing the resolution of the system, we apply the closed-loop transimpedance amplifier based on LIA measurement technique (self-balanced bridge). Several studies, PCB implementations and instruments of the self-balanced bridge readout have been reported in the literature [12-15]. Compared with these implementations (Table 6-1), this implementation shows a nice resolution with a simple structure, but the accuracy still needs to be improved. After comparing the characteristics of analog lock-in amplifier and digital lock-in amplifier from literature, a simple and portable self-balanced bridge system is proposed.

Table 6-1 Performance comparison

Reference	Frequency range [Hz]	C range [F]	C accuracy (err%)	Resolution (ppm)
[13]	n.a.	200p-400p	0.03%	75
[15]	50-20K	-0.165μ-+1.65μ	0.005%	0.16@1KHz
This work	2M-10M (tested)	22p-47p	n.a.	24.14@10MHz

In the circuit level design, it is composed of four main parts: an MCU board with serial port provides control commands for DDS and communicates with the computer. A four channels DDS is used as a signal generator, which has 10-bit amplitude resolution, 14-bit phase resolution and 0.12Hz frequency tuning resolution. A four channels pre-amplifier can increase the amplitude of the signal from DDS with a certain gain. A Charge amplifier with femto-ampere bias current is required whenever the difference of currents or voltage is small and needs to be accurately measured, which provide a certain gain by $-C_{IN}/C_F$. A low-noise amplifier with a high gain can increase the signal from the charge amplifier with less noise. A mixer and low-pass filter convert the input signal into a DC signal finally. Prepare for further increases in frequency while reducing the effects of parasitic resistance and noise, leave some margin for each amplifier is necessary. Simulation results of the circuit-level design have been analyzed in terms of signal amplitude, noise and offset.

In the software design, by using the MATLAB GUI as a control center, it can realize the communication to the MCU, the control of the signal generator and the DAQ, which provides a simple interface for the whole system.

For measurement, together with a simple test impedance bridge and a DAQ board on which the digital part of the readout circuit has been implemented, a fully-functional interface-based readout approach has been realized.

According to the noise analysis of the whole system and comparison with measurement results, the output noise is 33.5μV and 52.4μV respectively. As you can see, there is a significant difference in output noise because the noise of the DDS is not obtained, and it may also involve errors caused by temperature drift and other external interference. Therefore, combined with the previous analysis in section 3.5, the DDS, preamplifier and ADC are the three main noise sources, for further improvement of the system resolution, it is a good choice to start with these three components.

The same capacitors have been measured by using two different methods: the proposed closed-loop impedance bridge with LIA measurement using this system, and, for comparison, a high-resolution capacitive bridge (AH2700A) is chosen. The measured capacitors show good resolution with only 10-bit amplitude resolution (about 31.52fF with 32.25574pF reference capacitor). The capacitor values have a little bit of change with different parasitic resistances due to phase shift and lower resolution phase compensation. Since the system does not achieve a proper shielding and calibration, the accuracy of the system still is worse than AH2700A and the capacitance value shows a certain error. Furthermore, because of the phase resolution of DDS, the accuracy of resistance is not so good when resistance becomes larger. The total power consumption of these designed PCBs is 1.85W. Finally, the specifications of the whole system are shown in Table 6-2.

Table 6-2 Specifications of the system

Range of capacitance	Resolution without interpolation	Resolution of capacitor(2MHz)	Resolution of capacitor(5MHz)	Resolution of capacitor(10MHz)
22-47pF	31.52fF	±14.74ppm @10Hz	±16.80ppm @10Hz	±24.14ppm @10Hz

6.2 Future work

In order to obtain a higher resolution and accuracy, the proposed sensor interface could be further improved; there are various things can be considered in future work to improve the performance:

1. Use a better DDS with a higher amplitude resolution, which can improve the resolution and accuracy of the capacitive sensor directly. There are also DDS chips with 12-bit amplitude and 16-bit phase resolution so that these chips can be used in future work.
2. Decrease the resistance values of pre-amplifier, change a new op-amp of pre-amplifier with lower input current noise density and use a higher resolution ADC can also improve the system resolution significantly.
3. Build all functions of PCBs in a new PCB entirely with the better circuit layout, build a new PCB for the test impedance bridge as well, they can eliminate more errors from cables and parasitic capacitance from PCB.
4. Increase the upper limit of the input signal frequency so that the influence of parasitic resistances can be further reduced, which means the accuracy of the system can be improved.
5. Make an integrated version of the self-balanced bridge system with a standard bus interface as a commercial product.
6. To reduce the cost of the system, all components except the microcontroller and DDS can be considered to be fabricated into an integrated chip using CMOS technology. However, the noise,

offset, bias current/voltage and bandwidth of the amplifier and multiplier (especially the charge amplifier part) are bottlenecks. At the same time, according to the accuracy and resolution requirements of the system and the capacitance range of the sensor, the amplifier, multiplier and its feedback circuit need different design choices. Finally, by simplifying the software module, the entire feedback system can be controlled by the microcontroller only, using a touchscreen or wireless (personal computer) to operate circuit, so that the system can be made into a handheld device or even simpler.

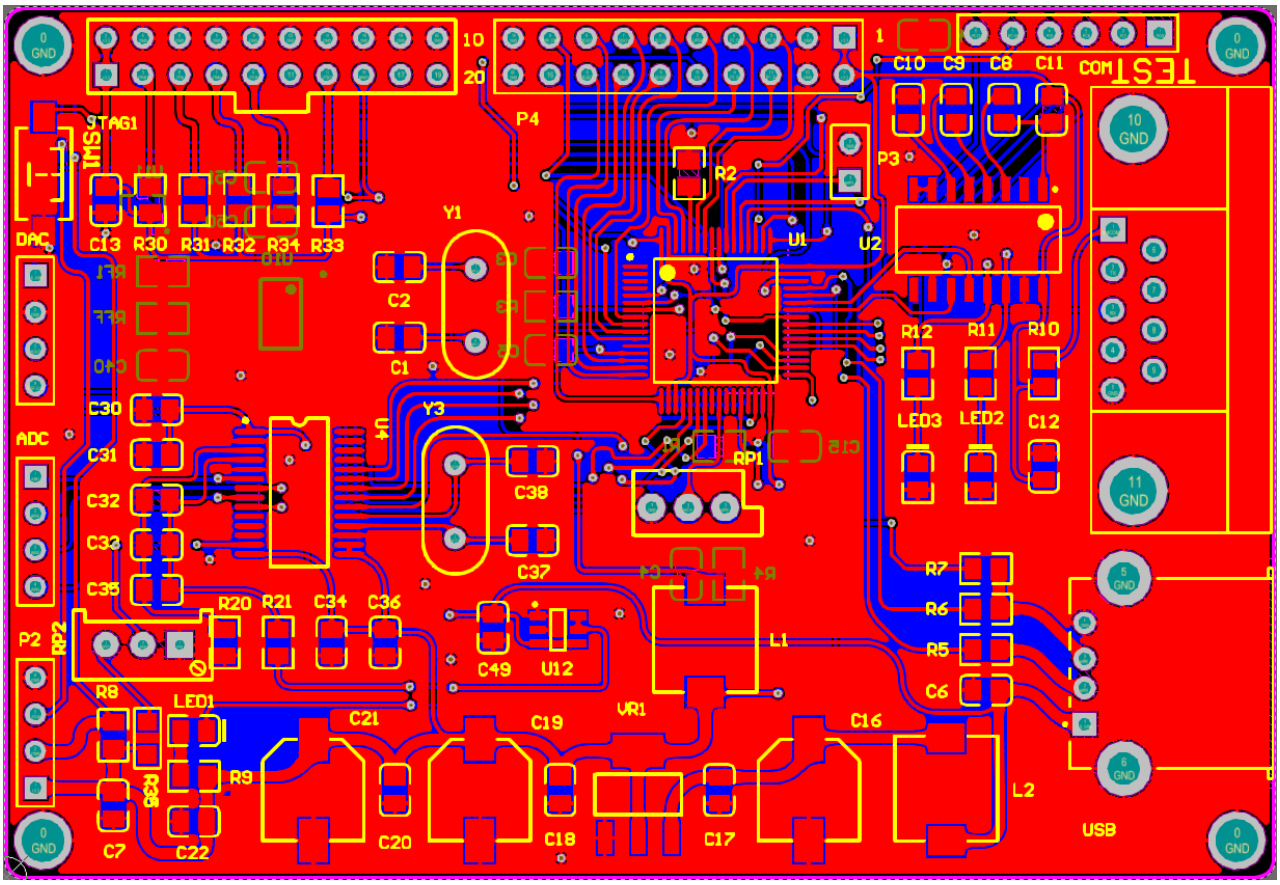
7. Do measurement with real sensors, in this way, the working performance of the circuit can be further verified.
8. Consider the phase shift from resistance in the wire and capacitance, for high accuracy, 0.01Ohm resistance still causes an unignored phase shift. Also, the parasitic resistance (Mega Ohm to Giga Ohm) and capacitance in parallel cause a phase shift, so for high resolution and accuracy systems, phase compensation is necessary.
9. Improve the software, which can decrease the measurement time significantly.
10. Do temperature test, use a better reference capacitor with temperature stabilization in an oven, may be needed to achieve better accuracy.

Reference

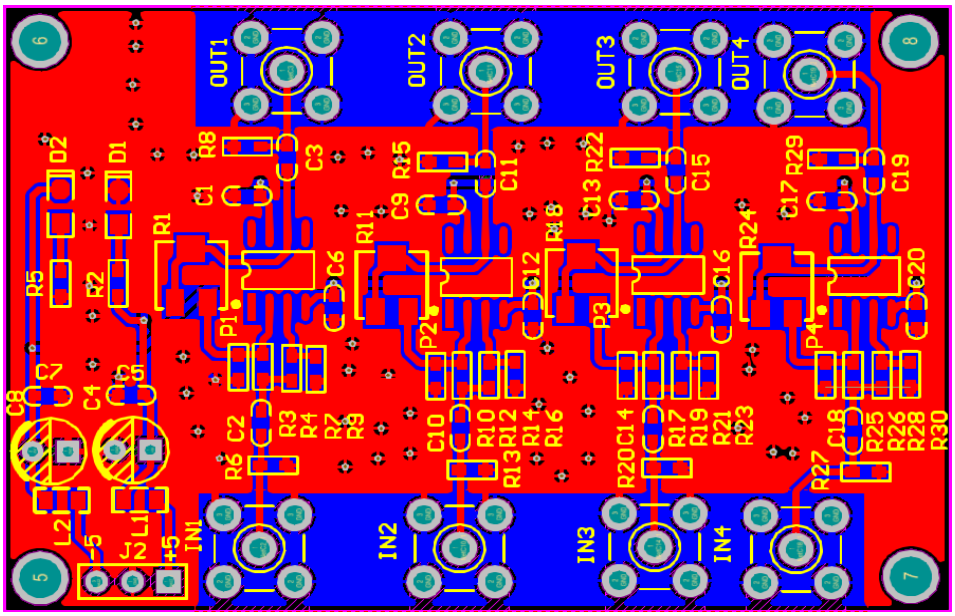
[1] J. Shiah, H. Rashtian, and S. Mirabbasi, "A low-noise high-sensitivity readout circuit for MEMS capacitive sensors," in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), pp.3280-3283, 2010.
[2] X. Li and G. C. M. Meijer, “An accurate interface for capacitive sensors,” IEEE Trans. Instrum. Meas., vol. 51, no. 5, pp. 935–939, Oct. 2002.
[3] B. George and V. J. Kumar, “Switched capacitor signal conditioning for differential capacitive sensors,” IEEE Trans. Instr. and Meas., vol. 56, No. 3, pp. 913-917, 2007.
[4] U. Schoneberg, H.G. Dura, B.J. Hosticka, W. Mokwa, Low- drift gas sensor with on-chip instrumentation, Proceedings of the 1991 International Conference on Solid-State Sensors and Actuators, San Francisco, CA, 1991, pp. 1006-1007.
[5] S.J. Sherman, et.al., “Low cost monolithic accelerometer”, Dig. VLSI Circuits Symp., June 1992, pp. 34-35.
[6] K. Chau, S.R. Lewis, Y, Zhao, R. T. Howe, S.F. Bart, and R.G. Marcheselli, “An integrated force-balanced capacitive accelerometer for low-g applications,” 1995 IEEE Conf. on Solid-State Sensors & Actuators, June 1995, pp. 593-596.
[7] J. Wu, G.K. Fedder, L.R. Carley, “A low-noise low-offset capacitive sensing amplifier for a 50-Pg/Hz monolithic CMOS MEMS accelerometer”, IEEE J. of Solid-State Circuits, vol. 39, May 2004, pp. 722-730.
[8] J. A. Geen, S. J. Sherman, J. F. Chang, S. R. Lewis, “Single chip surface micromachined integrated gyroscope with 50°/h Allan deviation,” IEEE J. of Solid-State Circuits, vol. 37, Dec. 2002, pp.1860-1866.
[9] Da Silva, M.J. Impedance Sensors for Fast Multiphase Flow Measurement and Imaging. Ph.D. Thesis, Technische Universität Dresden: Dresden, Germany, 08 November 2008.
[10] J-K Woo, C. Boyd, J. Cho, and K. Najafi, “Ultra-Low Noise Transimpedance Amplifier for High Performance MEMS Resonant Gyroscopes,” under review, Transducers 2017, June 2017.
[11] G. Royo, C. Sánchez-Azqueta, C. Gimeno, C. Aldea, S. Celma, “Programmable low-power low-noise capacitance to voltage converter for MEMS accelerometers”, Sensors, vol. 17, no. 1, p. 67, 2017.
[12] P. Holmberg, "Automatic balancing of linear AC bridge circuits for capacitive sensor elements", IEEE Trans. Instrum. Meas., vol. 44, no. 3, pp. 803-805, Jun. 1995.
[13] P. Mantenuto, A. De Marcellis, G. Ferri, "Novel modified De-Sauty autobalancing bridge-based analog interfaces for wide-range capacitive sensor applications", IEEE Sensors J., vol. 14, no. 5, pp. 1664-1672, May 2014.
[14] B. Hu, J. Wang, G. Song and F. Zhang, "A compact wideband precision impedance measurement system based on digital auto-balancing bridge", Measurement Science and Technology, vol. 27, no. 5, p. 055902, 2016.
[15] http://www.andeen-hagerling.com/ah2700a.htm

Appendix I PCB layout

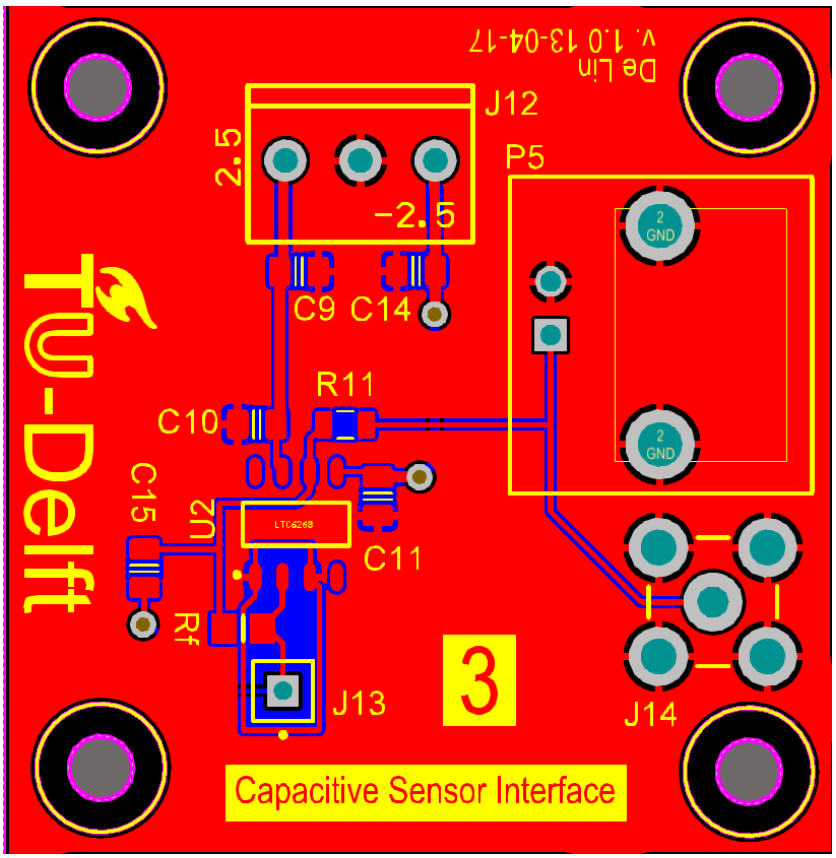
For Figure I-1 (a), it is an MCU board with STM32 (U1), ADC (U4), DAC (U10) and some interface, where JTAG1 is the interface to download the program, P4 is the interface to connect the DDS (AD9959) and COM is the interface to communicate with PC. For Figure I-1 (b), it is a Pre-amplifier board, where IN1 is connected with CH1 of DDS, OUT1 is connected with IN1 of mixer, IN2 is connected with CH0 of DDS, IN3 is connected with CH2 of DDS, OUT2 and OUT3 is connected with the reference capacitor and the unknown capacitors (the test impedance bridge) respectively. For Figure I-1 (c), it is a charge amplifier board, where J13 is connected with the output of the impedance bridge, and P4 or J14 is connected with J3 of LNA board. For Figure I-1 (d) it is a low-noise amplifier board, where OUT1 is connected with IN2 of the mixer. For Figure I-1 (e) it is a PSD and LPF board, using a jumper to connect two pins at the bottom of P1, OUT1 can be a connector for network analyzer, J2 is the output of the system and P2 is the ground of board.



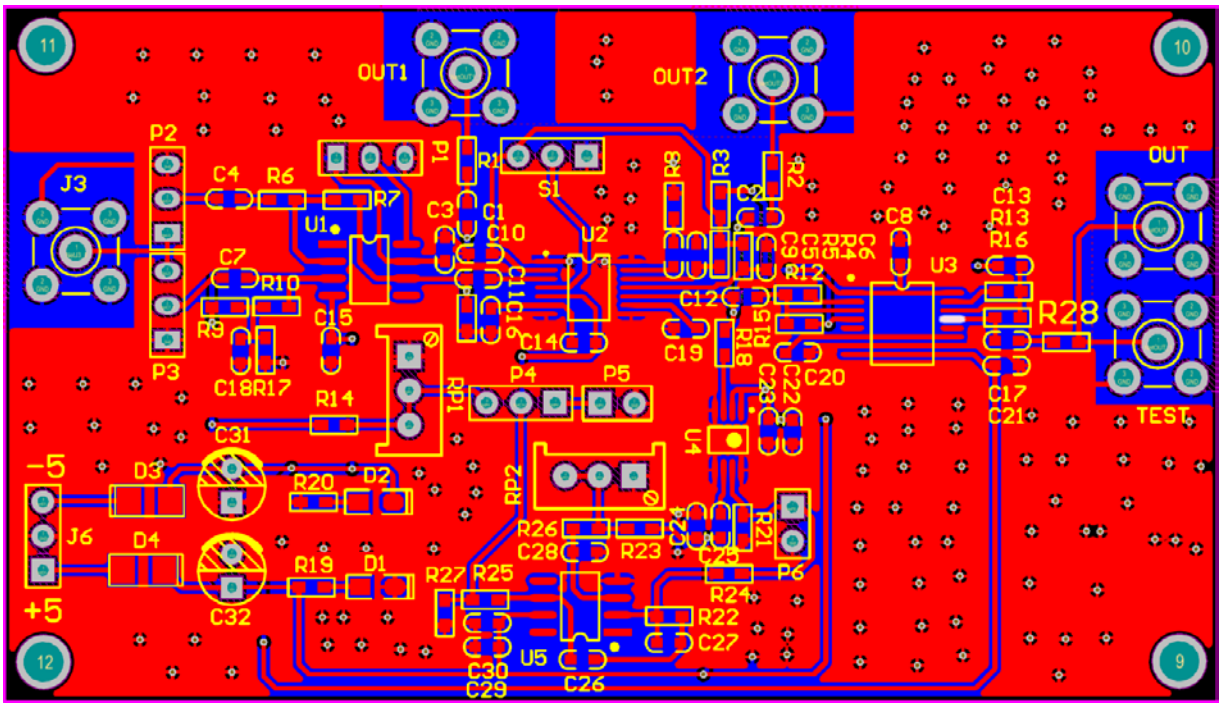
(a)


OUT1
OUT2
OUT3
OUT4
IN1
IN2
IN3
IN4
J2
+5
-5

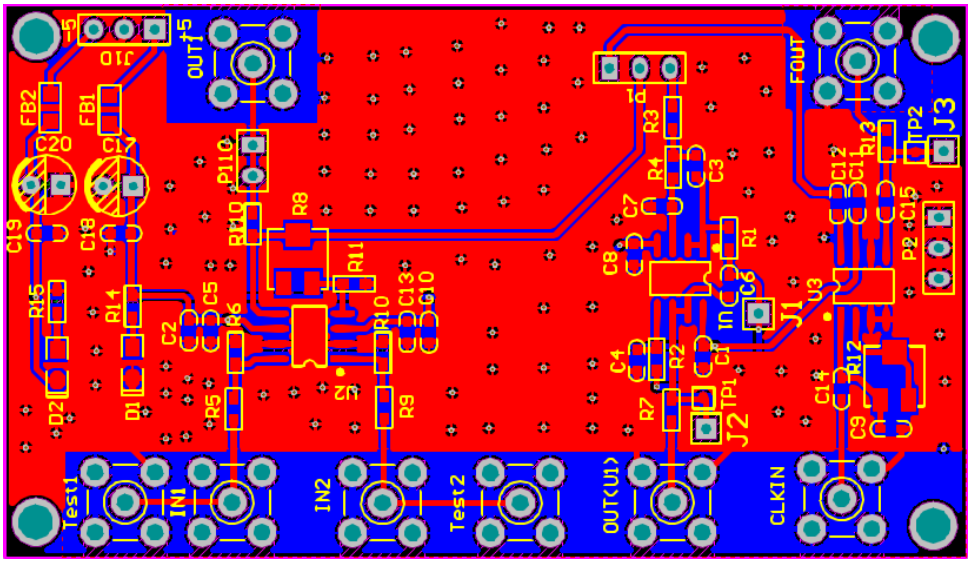
(b)


De Lin
v. 1.0 13-04-17
J12
2.5
-2.5
P5
2
GND
C9 C14
R11
C10
C15
U2
C11
Rf
J13
3
J14
TU-Delft
Capacitive Sensor Interface

(c)



(d)



(e)

Figure I-1: (a). MCU layout; (b). Pre-amplifier layout; (c). Charge amplifier layout; (d). Low-noise amplifier layout; (e). PSD and LPF layout

Appendix II C code

```
/*============================================================
//    function:      AD9959 Demo
//    date:          2017/3/6
//    note:
//    write:         Lin
============================================================*/

/* Includes ------------------------------------------------------------------*/


//#include "hw_config.h"
#include "AD9959_V1.h"
#include "TCL5615.h"
#include "ads1256.h"
#include "fliter.h"
#include "datadis.h"
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
unsigned char System_Flag=1;            // External signal control
unsigned char UART1_RX_Flag=0;
unsigned int Time1_Counter, Time2_Counter, Time3_Counter, Time4_Counter, System_Count, RXfreCount;
unsigned char UART1_Rx_Datas[64],Uart1_Rx_Counter=0;
//unsigned char UART1_Rx_Datas[64] =
{'&','C','H','0','\0','1','2','3','4','5','0','0','0','0','.','2','2','\0','1','2','3','.','4','4','4','\0','1','2','3','3','\0','#'},Uart1_Rx_Counter=0;

unsigned long System_Frequency = 500000;// Working frequency of AD9959
unsigned char Manual_Flag;

unsigned char Haed_String[18]=" AD9958/59   CH0 ";
unsigned char Fre_String[18]="F:100000000.12Hz";
unsigned char Phase_String[18]="Phase:    180.123";
unsigned char Amplitude_String[18]="Amplitude:   1023";
unsigned char Ch_Index=0;
unsigned char Type_Index=2;
unsigned char Position_Index=0;
```

```
unsigned long Set_Ref=1;
unsigned long Ch_Data[4][3];
unsigned char Updata_Flag;
double Ch_Data_double[4][3] = {0.0};
volatile unsigned long results = 0;
unsigned char buff[12] = {0};
int fputc(int ch, FILE *f)
{
      USART_SendData(USART1, (unsigned char) ch);
      while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);

      return (ch);
}
/************************ Supplementary code*****************************
***********/
void Set_System_double(void)
{
      GPIO_Configuration();
      USART_Configuration();
      TIM_Configuration();
      NVIC_Configuration();
      AD9959_Init();
      TCL5615_init();
      SPI_ADS1256_Init();
      ADS1256_GPIO_init();                                  // Initialization
      ADS1256_Init();
      UART1_RX_Flag = 0;
   Ch_Data_double[0][0]=1000000.12;
      Ch_Data_double[0][1]=0.111;
      Ch_Data_double[0][2]=1023;
      Ch_Data_double[1][0]=1000000.12;
      Ch_Data_double[1][1]=120;
      Ch_Data_double[1][2]=1023;
      Ch_Data_double[2][0]=1000000.12;
      Ch_Data_double[2][1]=240.123;
      Ch_Data_double[2][2]=1023;
      Ch_Data_double[3][0]=1000000.12;
      Ch_Data_double[3][1]=90.332;
      Ch_Data_double[3][2]=1023;
      dis_menu_double(0);

      ch_sw(0);//channel one
```

```
  AD99959_phase_double(Ch_Data_double[0][1]);//0
    AD9959_Amp(Ch_Data_double[0][2]);
    AD9959_frequency_double(Ch_Data_double[0][0]);
    ch_sw(1);// channel two
  AD99959_phase_double(Ch_Data_double[1][1]);//0
    AD9959_Amp(Ch_Data_double[1][2]);
    AD9959_frequency_double(Ch_Data_double[1][0]);
    ch_sw(2);// channel three
  AD99959_phase_double(Ch_Data_double[2][1]);//0
    AD9959_Amp(Ch_Data[2][2]);
    AD9959_frequency_double(Ch_Data_double[2][0]);
    ch_sw(3);// channel four
  AD99959_phase_double(Ch_Data_double[3][1]);//0
    AD9959_Amp(Ch_Data_double[3][2]);
    AD9959_frequency_double(Ch_Data_double[3][0]);
    IO_Update();

    UART1_RX_Flag=0;
  Uart1_Rx_Counter=0;
}
/*******************************************************************************
* Function Name   : main.
* Description     : Main routine.
* Input           : None.
* Output          : None.
* Return          : None.
*******************************************************************************/
/***************main function*********************************************
**********/
int main(void)
{
//    unsigned char loop_flag;

    Set_System_double();
    while (1)
    {
        if(UART1_RX_Flag) // Serial port instruction
        {
                if(UART1_Rx_Datas[0]=='&'   && UART1_Rx_Datas[31]=='#')//
                {
                        //uart_datas_dispose();
                    uart_datas_dispose_double();
```

```
                }
/*******************Supplementary code*********************************
**************/
                //9 Character data totally
                else if(UART1_Rx_Datas[32]=='@' && UART1_Rx_Datas[41]=='#')
                {
                        uart_datas_dispose_add();
                }
                else if(UART1_Rx_Datas[42]=='*' && UART1_Rx_Datas[44]=='#')
                {
                        uart_datas_tcl5615_add();
                }
/*******************Supplementary code*********************************
***************/

                Uart1_Rx_Counter = 0;
                UART1_RX_Flag=0;
        }
        if(RXfreCount > 1)
        {
                RXfreCount = 0;
                //results = (Filter(ADS1256_MUXP_AIN1 | ADS1256_MUXN_AINCOM));
              results++;
            if(results == 4096)
                results = 0;
            //sprintf(buff, "%ld", results);
                printf("%04ld",results);
        }
    }
}
#ifdef USE_FULL_ASSERT
/*******************************************************************************
* Function Name   : assert_failed
* Description     : Reports the name of the source file and the source line number
*                   where the assert_param error has occurred.
* Input           : - file: pointer to the source file name
*                   - line: assert_param error line source number
* Output          : None
* Return          : None
*******************************************************************************/
void assert_failed(uint8_t* file, uint32_t line)
{
```

```
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

  /* Infinite loop */
  while (1)
  {}
}
#endif

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Appendix III MATLAB code

```
function varargout = matlabserialstm32(varargin)
% MATLABSERIALSTM32 MATLAB code for matlabserialstm32.fig
%      MATLABSERIALSTM32, by itself, creates a new MATLABSERIALSTM32 or raises the existing
%      singleton*.
%
%      H = MATLABSERIALSTM32 returns the handle to a new MATLABSERIALSTM32 or the handle to
%      the existing singleton*.
%
%      MATLABSERIALSTM32('CALLBACK',hObject,eventData,handles,...) calls the local
%        function named CALLBACK in MATLABSERIALSTM32.M with the given input arguments.
%
%      MATLABSERIALSTM32('Property','Value',...) creates a new MATLABSERIALSTM32 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before matlabserialstm32_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to matlabserialstm32_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 matlabserialstm32

% Last Modified by GUIDE v2.5 10-May-2018 20:39:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @matlabserialstm32_OpeningFcn, ...
                   'gui_OutputFcn',  @matlabserialstm32_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 matlabserialstm32 is made visible.
function matlabserialstm32_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 matlabserialstm32 (see VARARGIN)
global All_Com
    All_Com = get(handles.Com,'string');
global All_baud
    All_baud=get(handles.Baud,'string');
global Dev_Serial
    Dev_Serial = instrhwinfo('serial');
global Usable_Port
    Usable_Port = Dev_Serial.SerialPorts;
% Choose default command line output for matlabserialstm32
set(handles.Multiplier_Factor,'value',20);
handles.output = hObject;
global count k
count = 0;
k=1;

data2_save=[0,0];
data_Cx_save=[0,0];
save data_save data2_save data_Cx_save
% save('data_save','data2_save','-append');
axes(handles.OUTPUT);
    hold on

% 2018.5.2 lin
% % % Create a capture card channel
```

```
global DaqCh ch2 ch1 t

% DaqCh = daq.getDevices;
DaqCh = daq.createSession('ni');
DaqCh.Rate = 1818181.8182;
DaqCh.DurationInSeconds = 0.01;
% DaqCh.IsContinuous = 1;

ch2 = DaqCh.addAnalogInputChannel('Dev1','ai1','Voltage');% Channel 2 is channel 2 of the stm32 device generator, and ai0 is the channel 0 of the capture card.

% ch1 = DaqCh.addAnalogInputChannel('Dev1','ai1','Voltage');
t = timer('period',0.4,'TimerFcn',{@dealDACdata, handles},'BusyMode','queue','ExecutionMode','fixedRate');

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes matlabserialstm32 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function dealDACdata(hObject, eventdata, handles)
global DaqCh
global Dev_Serial
global count
global    k data h x
format long
data = startForeground(DaqCh);
data1 = data(1000:17000);
datadeal = smooth(data1,16000);
data2 = mean(datadeal(100:15900));
% plot(data2);
%fprintf('ch2 volatge：%f\t',data2);
      data2 = data2-0.02332;
%         data2=str2num(char(vpa(data2,9)));
      b=char(vpa(data2*1000,6));%resolution of data
      data_now=data2;
%         set(handles.outputVol,'string',num2str(data2));
      set(handles.outputVol,'string',b);
%         if fix(count) == 1
%               count=0
% %                 clearpoints(h);
```

```
%                                                                                                   %
@(hObject,eventdata)matlabserialstm32('OUTPUT_CreateFcn',hObject,eventdata,guidata(hObject))
%               axes(handles.OUTPUT);
%               cla reset
%         else
%         end
      figure('visible','off');
      axes(handles.OUTPUT);
      plot(handles.OUTPUT,count,data_now,'ro');
      hold (handles.OUTPUT,'on')
%           drawnow
      count=count+1;
      if count==1
            axis auto
      else
      end

if    data2 > 0.001%V
      serialinfo = get(Dev_Serial);
      if strcmp(serialinfo.Status,'open')
            ch2fre = str2double(get(handles.CH2Fre,'string'));
            set(handles.CH2Fre,'string',char(sprintf('%012.2f',ch2fre)));
            ch2fre = get(handles.CH2Fre,'string');
            ch2fre = sprintf('%012s',ch2fre);

            ch2pha = str2double(get(handles.CH2Ph,'string'));
            set(handles.CH2Ph,'string',char(sprintf('%07.3f',ch2pha)));
            ch2pha = get(handles.CH2Ph,'string');
            ch2pha = sprintf('%07s',ch2pha);

            ch2amp = get(handles.CH2Amp,'string');
            ch2amp = int32(str2double(ch2amp))-1;
            set(handles.CH2Amp,'string',char(sprintf('%07.3f',ch2amp)));
            ch2amp = num2str(ch2amp);
            ch2amp = sprintf('%04s',ch2amp);

            fprintf(Dev_Serial,'&CH2');
            fprintf(Dev_Serial,ch2fre);
            fprintf(Dev_Serial,ch2pha);
            fprintf(Dev_Serial,ch2amp);
            fprintf(Dev_Serial,'%c','#');
            %%%%2018.5.3
```

```
            data_save=load('data_save');
            data2_save=data_save.data2_save;
            data_Cx_save=data_save.data_Cx_save;
            data2_save(k,:)=[data2,str2double(ch2amp)]
            save('data_save','data2_save','-append')

            data_ampCH2=str2num(get(handles.CH2Amp,'string'));
            data_ampCH0=str2num(get(handles.CH0Amp,'string'));
            phaseCH2=(str2num(get(handles.CH0Ph,'string'))/180)*pi;
            phaseCH0=(str2num(get(handles.CH2Ph,'string'))/180)*pi;
            data_Cx=(-1)*(data_ampCH0/data_ampCH2)*32.25574*cos(phaseCH2-phaseCH0);
            data_Cx=char(vpa(data_Cx,8));%save 8 wei data
            set(handles.edit_Cvalue,'string',data_Cx);

            data_Cx_save(k,:)=[str2num(data_Cx),data_ampCH2];
            save('data_save','data_Cx_save','-append')
            k=k+1;
%               set(handles.edit_Cvalue,'string',num2str(data_Cx));
            FreCH0=str2num(get(handles.CH0Fre,'string'));
            if sin(phaseCH2-phaseCH0)==0
                  data_Rx=inf;
            else

data_Rx=(data_ampCH2/data_ampCH0)/(2*pi*FreCH0*32.25574/(10^12)*sin(phaseCH2-
phaseCH0));
            end
            data_Rx=char(vpa(data_Rx,8));
            set(handles.edit_Resistor,'string',data_Rx);
      else
            errordlg('Don’t worry, just open the serial port');
      end
elseif data2 < -0.0007%v
      serialinfo = get(Dev_Serial);
      if strcmp(serialinfo.Status,'open')
            ch2fre = str2double(get(handles.CH2Fre,'string'));
            set(handles.CH2Fre,'string',char(sprintf('%012.2f',ch2fre)));
            ch2fre = get(handles.CH2Fre,'string');
            ch2fre = sprintf('%012s',ch2fre);

            ch2pha = str2double(get(handles.CH2Ph,'string'));
            set(handles.CH2Ph,'string',char(sprintf('%07.3f',ch2pha)));
            ch2pha = get(handles.CH2Ph,'string');
```

```
            ch2pha = sprintf('%07s',ch2pha);

            ch2amp = get(handles.CH2Amp,'string');
            ch2amp = int32(str2double(ch2amp))+1;
            set(handles.CH2Amp,'string',char(sprintf('%07.3f',ch2amp)));
            ch2amp = num2str(ch2amp);
            ch2amp = sprintf('%04s',ch2amp);

            fprintf(Dev_Serial,'&CH2');
            fprintf(Dev_Serial,ch2fre);
            fprintf(Dev_Serial,ch2pha);
            fprintf(Dev_Serial,ch2amp);
            fprintf(Dev_Serial,'%c','#');
            %2018.5.3
            data_save=load('data_save');
            data2_save=data_save.data2_save;
            data_Cx_save=data_save.data_Cx_save;
            data2_save(k,:)=[data2,str2double(ch2amp)]
            save('data_save','data2_save','-append')
            data_ampCH2=str2num(get(handles.CH2Amp,'string'));
            data_ampCH0=str2num(get(handles.CH0Amp,'string'));
            phaseCH2=(str2num(get(handles.CH0Ph,'string'))/180)*pi;
            phaseCH0=(str2num(get(handles.CH2Ph,'string'))/180)*pi;
            data_Cx=(-1)*(data_ampCH0/data_ampCH2)*32.25574*cos(phaseCH2-phaseCH0);
            data_Cx=char(vpa(data_Cx,8));%save 8 wei data
            set(handles.edit_Cvalue,'string',data_Cx);% Here data_Cx is a string form, if you want to
save, the next step is to convert it to num, then save
            data_Cx_save(k,:)=[str2num(data_Cx),data_ampCH2];
            save('data_save','data_Cx_save','-append');
            k=k+1;
%               set(handles.edit_Cvalue,'string',num2str(data_Cx));
            FreCH0=str2num(get(handles.CH0Fre,'string'));
            if sin(phaseCH2-phaseCH0)==0
                  data_Rx=inf;
            else

data_Rx=(data_ampCH2/data_ampCH0)/(2*pi*FreCH0*32.25574/(10^12)*sin(phaseCH2-
phaseCH0));
            end
            data_Rx=char(vpa(data_Rx,8));
            set(handles.edit_Resistor,'string',data_Rx);
      else
```

```
        errordlg(' Don’t worry, just open the serial port ');
    end
else
        data_ampCH2=str2num(get(handles.CH2Amp,'string'));
        data_ampCH0=str2num(get(handles.CH0Amp,'string'));
        phaseCH2=(str2num(get(handles.CH0Ph,'string'))/180)*pi;
        phaseCH0=(str2num(get(handles.CH2Ph,'string'))/180)*pi;
        data_save=load('data_save');
        data2_save=data_save.data2_save;
        data_Cx_save=data_save.data_Cx_save;
        data_A=data2_save(:,2);
        num_data_A=size(data_A,1);
%
%                           if    (num_data_A>5)&&((data_A(num_data_A,1)-data_A(num_data_A-
4,1))>0)%&&((data_A(num_data_A-1)-data_A(num_data_A-5))/4>0)
%                   lib1=find(data_A==(data_ampCH2-4));
%                   lib2=find(data_A==(data_ampCH2-1));
%               else

        if                          (num_data_A>4)&&(data_A(num_data_A)==data_A(num_data_A-
1))&&(data_A(num_data_A-2)==data_A(num_data_A-1))&&(data_A(num_data_A-
3)==data_A(num_data_A-2))&&(data_A(num_data_A-3)==data_A(num_data_A-4))
            data2_zhuangtai=true;
        else
            data2_zhuangtai=false;
        end

%           if data2_zhuangtai
            lib1=find(data_A==(data_ampCH2+1));
            lib2=find(data_A==(data_ampCH2+4));
%           end
        if size(lib1,1)==1
            lib1=lib1;
        else
            lib1=max(lib1);
        end
        if size(lib2,1)==1
            lib2=lib2;
        else
            lib2=max(lib2);
        end
        if isempty(lib1)||isempty(lib2)
```

```
            lib1=find(data_A==(data_ampCH2-4));
            lib2=find(data_A==(data_ampCH2-1));
            if size(lib1,1)==1
                lib1=lib1;
            else
                lib1=max(lib1);
            end
            if size(lib2,1)==1
                 lib2=lib2;
            else
                 lib2=max(lib2);
            end
            if isempty(lib1)||isempty(lib2)
                data_stepValue=0;
            else
                data_stepValue=(data2_save(lib2,1)-data2_save(lib1,1))/3; %average value
            end
        else
            data_stepValue=(data2_save(lib2,1)-data2_save(lib1,1))/3; % average value
        end
        if data2_zhuangtai
            data_stepValue=data_stepValue;
        else
            data_stepValue=0;
        end
        if data_stepValue==0
            data_Cx=(-1)*(data_ampCH0/data_ampCH2)*32.25574*cos(phaseCH2-phaseCH0);
        else
            data_Cx=(-1)*(data_ampCH0/data_ampCH2)*32.25574*cos(phaseCH2-
phaseCH0)+(data2/data_stepValue)*(1/1024)*32.25574;
        end

        data_stepValue=char(vpa(data_stepValue*1000,2));
        data_Cx=char(vpa(data_Cx,8));
        set(handles.edit_stepValue,'string',data_stepValue);
        set(handles.edit_Cvalue,'string',data_Cx);
        data2_save(k,:)=[data2,data_ampCH2];
        save('data_save','data2_save','-append')
        data_Cx_save(k,:)=[str2num(data_Cx),data_ampCH2];%同上
        save('data_save','data_Cx_save','-append')
        k=k+1;
end
```

```
% --- Outputs from this function are returned to the command line.
function varargout = matlabserialstm32_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;



function CH0Ph_Callback(hObject, eventdata, handles)
% hObject      handle to CH0Ph (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH0Ph as text
%        str2double(get(hObject,'String')) returns contents of CH0Ph as a double


% --- Executes during object creation, after setting all properties.
function CH0Ph_CreateFcn(hObject, eventdata, handles)
% hObject      handle to CH0Ph (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH1Ph_Callback(hObject, eventdata, handles)
% hObject      handle to CH1Ph (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH1Ph as text
```

```
%               str2double(get(hObject,'String')) returns contents of CH1Ph as a double


% --- Executes during object creation, after setting all properties.
function CH1Ph_CreateFcn(hObject, eventdata, handles)
% hObject       handle to CH1Ph (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%               See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH2Ph_Callback(hObject, eventdata, handles)
% hObject       handle to CH2Ph (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH2Ph as text
%               str2double(get(hObject,'String')) returns contents of CH2Ph as a double


% --- Executes during object creation, after setting all properties.
function CH2Ph_CreateFcn(hObject, eventdata, handles)
% hObject       handle to CH2Ph (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%               See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH1Fre_Callback(~, eventdata, handles)
% hObject       handle to CH1Fre (see GCBO)
```

```
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% tempdata = str2double(get(hObject,'string'));
% set(hObject,'string',char(sprintf('%012.2f',tempdata)));
% Hints: get(hObject,'String') returns contents of CH1Fre as text
%              str2double(get(hObject,'String')) returns contents of CH1Fre as a double


% --- Executes during object creation, after setting all properties.
function CH1Fre_CreateFcn(hObject, eventdata, handles)
% hObject      handle to CH1Fre (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH3Ph_Callback(hObject, eventdata, handles)
% hObject      handle to CH3Ph (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH3Ph as text
%              str2double(get(hObject,'String')) returns contents of CH3Ph as a double


% --- Executes during object creation, after setting all properties.
function CH3Ph_CreateFcn(hObject, eventdata, handles)
% hObject      handle to CH3Ph (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function CH0Fre_Callback(hObject, eventdata, handles)
% hObject    handle to CH0Fre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH0Fre as text
%        str2double(get(hObject,'String')) returns contents of CH0Fre as a double


% --- Executes during object creation, after setting all properties.
function CH0Fre_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH0Fre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function CH2Sli_Callback(hObject, eventdata, handles)
% hObject    handle to CH2Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.CH2Sli,'SliderStep',[0.001,0.001])
Va=get(hObject,'Value');
if rem(Va,1)~=0
    set(hObject,'Value',fix(Va));
end
set(handles.CH2Amp,'String',num2str(get(hObject,'Value')));
% set(handles.CH2Amp,'String',get(hObject,'Value'));
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
```

```
function CH2Sli_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH2Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function CH3Sli_Callback(hObject, eventdata, handles)
% hObject    handle to CH3Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.CH3Amp,'String',get(hObject,'Value'));
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function CH3Sli_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH3Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function CH0Sli_Callback(hObject, eventdata, handles)
% hObject    handle to CH0Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
set(handles.CH0Amp,'String',get(hObject,'Value'));
```

```
% --- Executes during object creation, after setting all properties.
function CH0Sli_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH0Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function CH1Sli_Callback(hObject, eventdata, handles)
% hObject    handle to CH1Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.CH1Amp,'String',get(hObject,'Value'));
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function CH1Sli_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH1Sli (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function CH2Fre_Callback(hObject, eventdata, handles)
% hObject    handle to CH2Fre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH2Fre as text
```

```
%               str2double(get(hObject,'String')) returns contents of CH2Fre as a double


% --- Executes during object creation, after setting all properties.
function CH2Fre_CreateFcn(hObject, eventdata, handles)
% hObject       handle to CH2Fre (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%               See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH3Fre_Callback(hObject, eventdata, handles)
% hObject       handle to CH3Fre (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CH3Fre as text
%               str2double(get(hObject,'String')) returns contents of CH3Fre as a double


% --- Executes during object creation, after setting all properties.
function CH3Fre_CreateFcn(hObject, eventdata, handles)
% hObject       handle to CH3Fre (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%               See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH0Amp_Callback(hObject, eventdata, handles)
% hObject       handle to CH0Amp (see GCBO)
```

```
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.CH0Sli,'value',str2double(get(hObject,'string')));
% Hints: get(hObject,'String') returns contents of CH0Amp as text
%        str2double(get(hObject,'String')) returns contents of CH0Amp as a double


% --- Executes during object creation, after setting all properties.
function CH0Amp_CreateFcn(hObject, eventdata, handles)
% hObject      handle to CH0Amp (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH1Amp_Callback(hObject, eventdata, handles)
% hObject      handle to CH1Amp (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.CH1Sli,'value',str2double(get(hObject,'string')));
% Hints: get(hObject,'String') returns contents of CH1Amp as text
%        str2double(get(hObject,'String')) returns contents of CH1Amp as a double


% --- Executes during object creation, after setting all properties.
function CH1Amp_CreateFcn(hObject, eventdata, handles)
% hObject      handle to CH1Amp (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function CH2Amp_Callback(hObject, eventdata, handles)
% hObject    handle to CH2Amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.CH2Sli,'value',str2double(get(hObject,'string')));
% Hints: get(hObject,'String') returns contents of CH2Amp as text
%        str2double(get(hObject,'String')) returns contents of CH2Amp as a double


% --- Executes during object creation, after setting all properties.
function CH2Amp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH2Amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CH3Amp_Callback(hObject, eventdata, handles)
% hObject    handle to CH3Amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.CH3Sli,'value',str2double(get(hObject,'string')));
% Hints: get(hObject,'String') returns contents of CH3Amp as text
%        str2double(get(hObject,'String')) returns contents of CH3Amp as a
%        double9uc9b2


% --- Executes during object creation, after setting all properties.
function CH3Amp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CH3Amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function RefClk_Callback(hObject, eventdata, handles)
% hObject    handle to RefClk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
crystal_fre = str2double(get(hObject,'string'));
if crystal_fre<20 ||crystal_fre>30
    errordlg(' The crystal reference clock frequency range can only be 20Mhz~30Mhz ');
    set(hObject,'string','25');
    return;
end
% Hints: get(hObject,'String') returns contents of RefClk as text
%           str2double(get(hObject,'String')) returns contents of RefClk as a double


% --- Executes during object creation, after setting all properties.
function RefClk_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RefClk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in Multiplier_Factor.
function Multiplier_Factor_Callback(hObject, eventdata, handles)
% hObject    handle to Multiplier_Factor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Multiplier_Factor contents as cell array
%           contents{get(hObject,'Value')} returns selected item from Multiplier_Factor
```

```
% --- Executes during object creation, after setting all properties.
function Multiplier_Factor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Multiplier_Factor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



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

% Hint: get(hObject,'Value') returns toggle state of Vco_Switch


% --- Executes on button press in Load.

% --- Executes on selection change in Com.
function Com_Callback(hObject, eventdata, handles)
global Usable_Port
global All_Com
global All_baud
global Dev_Serial
if isempty(Usable_Port)
    errordlg('Connect the serial port, insert USB ');
else
    num = numel(Usable_Port);% Check to see how many serial ports are inserted.
    % Get the currently selected port number and compare it with the port number actually inserted
by the computer.
    % If the selected one matches the actual insert, create a serial port object.
    handles.COM_value = 0;
    for i = 1:num
        if strcmpi(All_Com{get(handles.Com,'Value')},Usable_Port{i})
```

```
                handles.COM_value = i;
                guidata(hObject,handles);
                break;
            else
                handles.COM_value = 0;
                guidata(hObject,handles);
            end
        end
        if handles.COM_value == 0
            errordlg('Pick the correct serial port number pls ');
        else
        % Create a serial port object
        Dev_Serial                                                                                    =
serial(Usable_Port{handles.COM_value},'BaudRate',str2double(All_baud{get(handles.Baud,'Value')
}),...
                    'OutputBufferSize',1000,'InputBufferSize',1000);

        set(handles.Open,'Enable','on');% The port number is selected to allow the serial port to be opened,
indicating that the serial port object has been established.
        end



end
% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
            set(hObject,'BackgroundColor','white');
        end
% hObject       handle to Com (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Com contents as cell array
%             contents{get(hObject,'Value')} returns selected item from Com

function    RXcallback(hObject, event, handles)
%UNTITLED
global k data h    flag
flag = 0;
% data(k) = zeros(100,1);
if hObject.BytesAvailable ~= 0
head = fscanf(hObject,'%c',4);
```

```
data(k) = str2double(head);
% set(handles.outputVol,'string',data(k));%2018.5.3
end
% if flag == 0
%    addpoints(h,x(k),data(k));
%    drawnow
% end
% if flag == 1
%    addpoints(h,x(k),data(k));
%    drawnow;
%    flag = 0;
% end
% if    k == 100 && flag~= 1
%          clearpoints(h);
%          k = 1;flag = 1;
% end
% k = k+1;




% --- Executes during object creation, after setting all properties.
function Com_CreateFcn(hObject, eventdata, handles)
% hObject       handle to Com (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called


% --- Executes on selection change in Baud.
function Baud_Callback(hObject, eventdata, handles)
% hObject       handle to Baud (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
global All_baud
global Dev_Serial
stopasync(Dev_Serial);
Dev_Serial.baudrate = str2double(All_baud{get(handles.Baud,'Value')});
% Hints: contents = cellstr(get(hObject,'String')) returns Baud contents as cell array
%              contents{get(hObject,'Value')} returns selected item from Baud
```

```
% --- Executes during object creation, after setting all properties.
function Baud_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Baud (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in Open.
function Open_Callback(hObject, eventdata, handles)
% hObject    handle to Open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Do not modify the serial port number after opening
% Open the program when it is not open. When it is opened, it will prompt: it is already open, which
is not equal to 0.
global Dev_Serial

judge = strcmp(get(hObject,'BackgroundColor'),'r');
if  judge == 0 && handles.COM_value ~= 0
    set(handles.Com,'Enable','off');
    set(handles.Baud,'Enable','off');
    Dev_Serial.BytesAvailableFcn = @(hObject,event) RXcallback(hObject, event,handles);
    Dev_Serial.BytesAvailableFcnMode='byte';            % Set event trigger to accept trigger
    Dev_Serial.timeout = 0.01;
    Dev_Serial.BytesAvailableFcnCount = 4;
    fopen(Dev_Serial);  %Open serial port
    set(hObject,'BackgroundColor','r');
    set(hObject,'Enable','off');
end

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

```
% Find the serial port object
scoms = instrfind;
% Try to stop and close the delete serial port object
if ~isempty(scoms)
        stopasync(scoms);
        fclose(scoms);
%           delete(scoms);
end
% delete(instrfindall);
set(handles.Com,'Enable','on');
set(handles.Open,'Enable','on');
set(handles.Open,'BackgroundColor','w');
set(handles.Baud,'Enable','on');
% --- Executes on button press in LCH1.
function LCH1_Callback(hObject, eventdata, handles)
% hObject        handle to LCH1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUID    ATA)
global Dev_Serial
  serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')

        ch1fre = str2double(get(handles.CH1Fre,'string'));
        set(handles.CH1Fre,'string',char(sprintf('%012.2f',ch1fre)));
        ch1fre = get(handles.CH1Fre,'string');
        ch1fre = sprintf('%012s',ch1fre);
        ch1pha = str2double(get(handles.CH1Ph,'string'));
        set(handles.CH1Ph,'string',char(sprintf('%07.3f',ch1pha)));
        ch1pha = get(handles.CH1Ph,'string');
        ch1pha = sprintf('%07s',ch1pha);
        ch1amp = get(handles.CH1Amp,'string');
        ch1amp = int32(str2double(ch1amp));
        ch1amp = num2str(ch1amp);
        ch1amp = sprintf('%04s',ch1amp);
        fprintf(Dev_Serial,'&CH1');
        fprintf(Dev_Serial,ch1fre);
        fprintf(Dev_Serial,ch1pha);
        fprintf(Dev_Serial,ch1amp);
        fprintf(Dev_Serial,'%c','#');
else
        errordlg(' Don’t worry, just open Serial Port ');
end
```

```
% --- Executes on button press in LCH2.
function LCH2_Callback(hObject, eventdata, handles)
% hObject        handle to LCH2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
global Dev_Serial
  serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')
        ch2fre = str2double(get(handles.CH2Fre,'string'));
        set(handles.CH2Fre,'string',char(sprintf('%012.2f',ch2fre)));
        ch2fre = get(handles.CH2Fre,'string');
        ch2fre = sprintf('%012s',ch2fre);

        ch2pha = str2double(get(handles.CH2Ph,'string'));
        set(handles.CH2Ph,'string',char(sprintf('%07.3f',ch2pha)));
        ch2pha = get(handles.CH2Ph,'string');
        ch2pha = sprintf('%07s',ch2pha);

        ch2amp = get(handles.CH2Amp,'string');
        ch2amp = int32(str2double(ch2amp));
        ch2amp = num2str(ch2amp);
        ch2amp = sprintf('%04s',ch2amp);

        fprintf(Dev_Serial,'&CH2');
        fprintf(Dev_Serial,ch2fre);
        fprintf(Dev_Serial,ch2pha);
        fprintf(Dev_Serial,ch2amp);
        fprintf(Dev_Serial,'%c','#');
else
        errordlg(' Don’t worry, just open Serial Port ');
end

% --- Executes on button press in LCH3.
function LCH3_Callback(hObject, eventdata, handles)
% hObject        handle to LCH3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
global Dev_Serial
  serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')
        ch3fre = str2double(get(handles.CH3Fre,'string'));
```

```
        set(handles.CH3Fre,'string',char(sprintf('%012.2f',ch3fre)));
        ch3fre = get(handles.CH3Fre,'string');
        ch3fre = sprintf('%012s',ch3fre);
        ch3pha = str2double(get(handles.CH3Ph,'string'));
        set(handles.CH3Ph,'string',char(sprintf('%07.3f',ch3pha)));
        ch3pha = get(handles.CH3Ph,'string');
        ch3pha = sprintf('%07s',ch3pha);
        ch3amp = get(handles.CH3Amp,'string');
        ch3amp = int32(str2double(ch3amp));
        ch3amp = num2str(ch3amp);
        ch3amp = sprintf('%04s',ch3amp);
        fprintf(Dev_Serial,'&CH3');
        fprintf(Dev_Serial,ch3fre);
        fprintf(Dev_Serial,ch3pha);
        fprintf(Dev_Serial,ch3amp);
        fprintf(Dev_Serial,'%c','#');
else
        errordlg(' Don’t worry, just open Serial Port ');
end

% --- Executes on button press in LCH0.
function LCH0_Callback(hObject, eventdata, handles)
% hObject        handle to LCH0 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
global Dev_Serial
  serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')
        ch0fre = str2double(get(handles.CH0Fre,'string'));
        set(handles.CH0Fre,'string',char(sprintf('%012.2f',ch0fre)));
        ch0fre = get(handles.CH0Fre,'string');
        ch0fre = sprintf('%012s',ch0fre);
        ch0pha = str2double(get(handles.CH0Ph,'string'));
        set(handles.CH0Ph,'string',char(sprintf('%07.3f',ch0pha)));
        ch0pha = get(handles.CH0Ph,'string');
        ch0pha = sprintf('%07s',ch0pha);
        ch0amp = get(handles.CH0Amp,'string');
        ch0amp = int32(str2double(ch0amp));
        ch0amp = num2str(ch0amp);
        ch0amp = sprintf('%04s',ch0amp);
        fprintf(Dev_Serial,'&CH0');
        fprintf(Dev_Serial,ch0fre);
```

```
        fprintf(Dev_Serial,ch0pha);
        fprintf(Dev_Serial,ch0amp);
        fprintf(Dev_Serial,'%c','#');
else
        errordlg(' Don’t worry, just open Serial Port ');
end


% --- Executes on button press in Load.
function Load_Callback(hObject, eventdata, handles)
% hObject        handle to Load (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
% Vco_Switch is not enabled by default
global Dev_Serial
vco = get(handles.Vco_Switch,'value');
if vco == 0
        vco = '0';
else
        vco = '1';
end
CurrMulFac = get(handles.Multiplier_Factor,'value'); % obtain the input reference clock
if CurrMulFac<4 ||CurrMulFac>20
        CurrMulFac = 1;
        set(handles.Multiplier_Factor,'value',CurrMulFac);
end
        Sysclk = str2double(get(handles.RefClk,'string'))*CurrMulFac;
if Sysclk<0 ||Sysclk>500
        errordlg('Reset frequency and multiple');
        return;
end
if Sysclk > 255 && vco ~= '1'
        errordlg('Open VCO please');
        return;
end
  serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')% If the serial port is open, then works
        Sysclk = num2str(Sysclk);
        Sysclk = sprintf('%03s',Sysclk);
        CurrMulFac = num2str(CurrMulFac);
        CurrMulFac = sprintf('%02s',CurrMulFac);
        fprintf(Dev_Serial,'%c','@');
```

```
        fprintf(Dev_Serial,Sysclk);
        fprintf(Dev_Serial,CurrMulFac);
        fprintf(Dev_Serial,'%c',vco);
        fprintf(Dev_Serial,'%c','#');
else
        errordlg(' Don’t worry, just open Serial Port ');
end


% --- Executes on selection change in outputVol.
function outputVol_Callback(hObject, eventdata, handles)
% hObject       handle to outputVol (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
% set(handles.outputVol,'color','r');
% Hints: contents = cellstr(get(hObject,'String')) returns outputVol contents as cell array
%              contents{get(hObject,'Value')} returns selected item from outputVol
global Dev_Serial
serialinfo = get(Dev_Serial);
if strcmp(serialinfo.Status,'open')
        strval = get(hObject,'string');
        fprintf(Dev_Serial,'%c','*');
        strval = sprintf('%05s',strval);
        fprintf(Dev_Serial,'%s',strval);
        fprintf(Dev_Serial,'%c','\0');
        fprintf(Dev_Serial,'%c','#');
else
        set(hObject,'BackGroundColor','w');
end

% --- Executes during object creation, after setting all properties.
function outputVol_CreateFcn(hObject, eventdata, handles)
% hObject       handle to outputVol (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
% function OUTPUT_CreateFcn(hObject, eventdata, handles)
% hObject        handle to OUTPUT (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called
% clc;
% global k h x
% k = 1;
% set(hObject,'Xlim',[0,50]);
% set(hObject,'Ylim',[-2048,2048]);%[0,4096]);
% h =    animatedline('color','r');%('parent',h1,'color','r');;
% numpoints = 100;
% x = linspace(0,20,numpoints);

% Hint: place code in OpeningFcn to populate OUTPUT


% --- Executes on button press in begingetcartdata.
function begingetcartdata_Callback(hObject, eventdata, handles)
% hObject        handle to begingetcartdata (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
global t count k
% if ishandle(t)
%         start(t)
% else
%         fprintf('Restart Software')
% end
if get(handles.begingetcartdata,'value')
      start(t)
else
      stop(t)
%         delete(t)
end



% --- Executes during object creation, after setting all properties.
function begingetcartdata_CreateFcn(hObject, eventdata, handles)
% hObject        handle to begingetcartdata (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
```

```
% handles       empty - handles not created until after all CreateFcns called
%timer creat



function edit_Cvalue_Callback(hObject, eventdata, handles)
% hObject       handle to edit_Cvalue (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_Cvalue as text
%              str2double(get(hObject,'String')) returns contents of edit_Cvalue as a double


% --- Executes during object creation, after setting all properties.
function edit_Cvalue_CreateFcn(hObject, eventdata, handles)
% hObject       handle to edit_Cvalue (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end



function edit_stepValue_Callback(hObject, eventdata, handles)
% hObject       handle to edit_stepvalue (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_stepvalue as text
%              str2double(get(hObject,'String')) returns contents of edit_stepvalue as a double


% --- Executes during object creation, after setting all properties.
function edit_stepValue_CreateFcn(hObject, eventdata, handles)
% hObject       handle to edit_stepvalue (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton_clear.
function pushbutton_clear_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_clear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global count
axes(handles.OUTPUT) ;
cla reset
count=0;
hold on



function edit_Resistor_Callback(hObject, eventdata, handles)
% hObject    handle to edit_Resistor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_Resistor as text
%        str2double(get(hObject,'String')) returns contents of edit_Resistor as a double


% --- Executes during object creation, after setting all properties.
function edit_Resistor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_Resistor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```