DIRECT DIGITAL SYNTHESIS FOR APPLICATIONS IN ATOM INTERFEROMETRY

A Thesis

submitted to

Indian Institute of Science Education and Research Pune

in partial fulfilment of the requirements for the BS-MS Dual Degree Programme

by

Akshay Shanbhag

Registration ID- 20181057

Indian Institute of Science Education and Research Pune

Dr. Homi Bhabha Road,

Pashan, Pune 411008, INDIA.

October, 2023

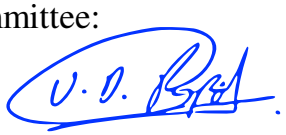
Supervisor: Umakant Rapol

Certificate

This is to certify that this dissertation entitled “Direct Digital Synthesis for Applications in Atom Interferometry” towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/ work carried out by Akshay Shanbhag at Indian Institute of Science Education and Research under the supervision of Prof. Umakant D. Rapol, Department of Physics, during the academic year 2022-2023.

Prof. Umakant D. Rapol

Committee:



Prof. Umakant Rapol

Prof. Shivprasad Patil

To my family, friends and alma maters

Declaration

I hereby declare that the matter embodied in the report entitled Direct Digital Synthesis for Applications in Atom Interferometry are the results of the work carried out by me at the Department of Physics, Indian Institute of Science Education and Research, Pune, under the supervision of Prof. Umakant Rapol and the same has not been submitted elsewhere for any other degree

Akshay Shanbhag

Date: 05/12/2023

Acknowledgements

I have always been keen with experimental Physics and wanted to pursue an Integrated Masters degree in Physics. October 2022 was confusing when it came to choosing a supervisor and expert for my MS project (related to electronics) but it eased out slowly and steadily, and I decided to work with Prof. Umakant Rapol (supervisor) and Prof. Shivprasad Patil (expert), hence joining the Atomic Physics and Quantum Optics lab for the first time. It's been an exciting time in this lab so far.

Building an experimental setup for the DDS and atom interferometer was not a cakewalk. I was lucky to have the people who have always been there to help at every crucial point of the project. First of all, it gives me immense pleasure to thank my supervisor, Prof. Umakant Rapol for assigning a useful project to me. He has always been ready to consider some good ideas for the project and has never been hesitant whenever it comes to doing any experiment. He has been a valuable professor, a leader and an inspiration for all. I would also like to thank my expert Prof. Shivprasad Patil for providing me valuable feedback and some useful suggestions for the project.

My labmates further eased the understanding process in the project. I was always guided by Pranab Dutta and Korak Biswas. They were like a teaching assistant when it came to experiments. I always had the opportunity to ask them even with some silly questions. I would also like to thank my other labmates Vishal Lal, Rayees A S and Shiv Sagar Maurya for all the discussions related to this project.

Building the setup for DDS would have been difficult if not for the support from technical officer Mr. Nilesh Dumbre. I would also like to thank Mr. Prabhakar Angare for handling all the official works related to this project.

I acknowledge IISER Pune for providing a pleasant area for working and funding of my scholarship during BS – MS. I would also like to thank the Kishore Vaigyanik Protsahan Yojana and Department of Science and Technology (Government of India) for providing me the scholarship throughout the undergraduate programme at IISER Pune.

Last but not the least, a big thanks to my family for providing me their constant love and support and making everything of this project possible. I would not have been here without their support.

Date: 05/12/2023 (Akshay Shanbhag)

Table of Contents

List of Tables

List of Figures

Abstract

In this project, we give a brief idea about atom interferometry, the setup of the device and the usage of direct digital synthesis (DDS) in atom interferometry, what exactly a DDS is, the working of a DDS in terms of it's parts, along with the DDS used (AD9850 and AD9910), their working and operation with an emphasis on the two modes of operation used in AD9910- single tone mode and digital ramp mode. The sweeping of frequency for digital ramp mode is done from 80 MHz to 82.5 MHz in steps of 0.84 MHz followed by phase locking for both modes of operation of the DDS. Atom interferometry is crucial in various aspects of science such as measuring gravity and estimating physical constants such as gravitational constant and fine structure constant. Doing the former is crucial for geophysical processes, geology, mineral exploration, and volcanology as monitoring local variations in gravity provides insights into tectonic deformations, tides, ocean and glacier dynamics, and Earth's structure.

Chapter 1 - INTRODUCTION

This thesis presents the use of Direct Digital Synthesis (DDS) in atom interferometry which finds immense applications in the field of science such as measuring gravity and various other fundamental constants, while in this chapter, the principle of atom interferometry and the role DDS plays in its implementation are described. Atom interferometry mainly focusses on the interaction of atoms with light, and works on the principle of diffraction. Atom interferometry is similar to optical interferometry except for the fact that the roles of light and matter are interchanged among each other – matter waves are used instead of electromagnetic waves.

1.1 WHAT IS ATOM INTERFEROMETRY?

Atom interferometry (AI) is one of the most precise techniques to carry out precision measurements in Physics. AI uses interference of matter waves – The wave nature of atoms. Unlike light, atoms possess mass and magnetic moments that get affected by gravitational fields and other physical parameters like magnetic fields, electric fields (depending upon the choice and internal states of the atoms) and thus these parameters can be measured using AI very precisely. In atom interferometry, laser beams are used to generate various analogous optical elements (for e.g.- mirrors, beamsplitters) for atomic state/ trajectory manipulation. The common elements are the 'mirrors', 'beamsplitters' and 'gratings'. Two of these elements the 'mirrors' and 'Beamsplitters' are used in atom interferometry. Analogous to a Mach Zehnder interferometer (see Figure 1.1), initially a beam splitter is used to split the atomic wavepacket in two different directions. The wavepackets are reflected with two mirrors and are finally merged using the second and the final beam splitter. The atomic populations in the two ports of the beam splitter depends on the path difference that the wavepackets gather during their flight in the two paths. Hence, any phase changes can be detected by monitoring the populations at the output ports of the beam splitter. [1]

1.2 WORKING OF AN ATOM INTERFEROMETER

The working of atom interferometer begins with a source that produces a cloud of cold atoms. The source of cold atoms is a magneto-optic trap that uses a combination of lasers and magnetic fields to cool atoms down to micro kelvin temperatures. The sample of cloud of cold atoms is then velocity selected to narrow down the velocity distribution before the implementation of the interferometer which then divides the atomic cloud into two spatially distinct halves, one made up of atoms in the original momentum state or ground state and the other one made up of atoms in a different momentum state or the excited state. A detection stage quantifies the results, and

measures the signal of interest by quantifying the relative numbers of atoms in the two states exiting the interferometer. The relative number of atoms is decided by transition probability which is a measurement of the required signal. [2]

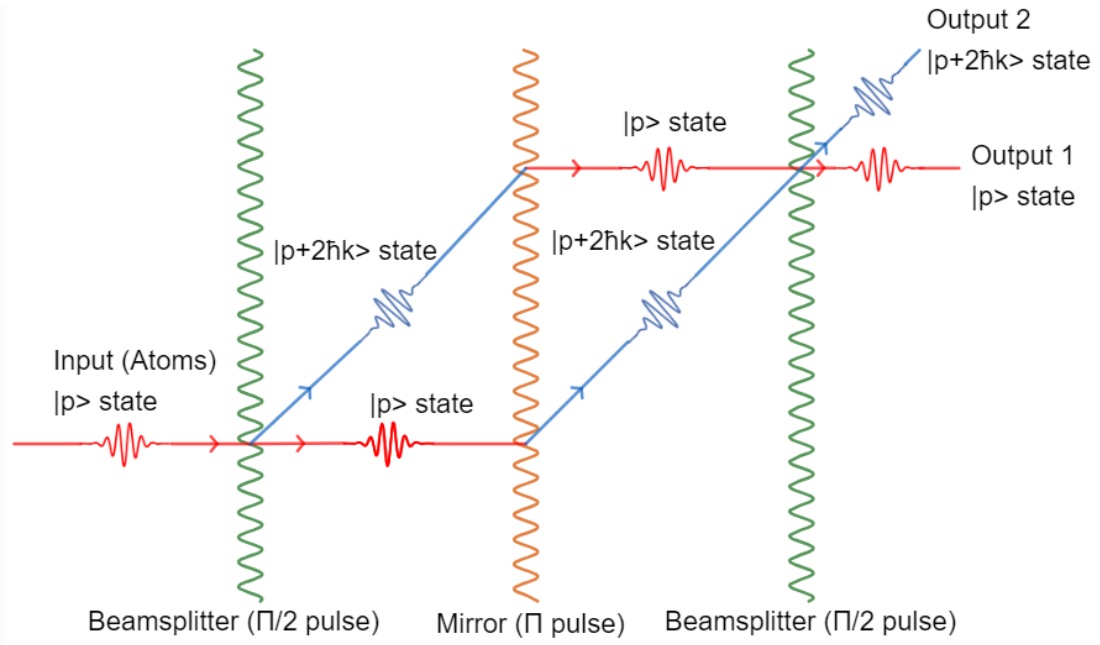


Figure 1.1- Concept and schematic of a cold atom interferometer, red solid lines represent atoms in ground state and blue solid lines represent atoms in excited state

1.3 BASIC USES OF ATOM INTERFEROMETRY

1] Atom interferometry can be used for the determination and measurements of rotations, fine structure constant, atomic polarizability, local gravitational fields and gravity gradients. For gravimeters that detect the local gravitational acceleration, the measurement of g in atom interferometric gravimeters is achieved by using frequency sweep of the moving optical lattice that diffracts atomic wavepackets in different momentum states. The frequency sweep of the optical lattice is needed to compensate for the changing Doppler shift of the accelerating atoms that are falling under gravity. The frequency sweep has to be precise and controllable down to a fraction of a Hz (in ~15 KHz, the frequency difference caused by Doppler shift of the photons due to gravitational acceleration). In order to achieve this frequency sweep, a Direct Digital Synthesis (DDS) based waveform generator is of extreme utility. The laser frequency difference (depending on the atoms used in the atom interferometer) can be modified at each pulse by utilizing a precise and stable DDS/ function generator to switch between three particular frequencies, one per pulse. [3]

2] Due to the high sensitivity of the devices used for atom interferometry, they can also be used in gravity mapping, rotation sensing and magnetic field gradient which cannot be accessed by other interferometers.

1.4 SETUP OF ATOM INTERFEROMETRY

The optical system in case of Rb atoms is comprised of a pair of extended cavity laser diodes (ECDL), of wavelength 780.2 nm. As a frequency selective element, every cavity incorporates a laser diode chip, a collimating lens, and grating. The diodes emit a linearly polarised laser field. By responding on the cylindrical piezoelectric (PZT) actuators that hold the reflecting mirrors, a slow but extensive adjustment of laser frequencies is produced. [3]

For this experiment, a Bose Einstein condensate (BEC) consisting of ^{87}Rb atoms (around 50000 every 15 seconds) is created in an optical dipole trap, by evaporative cooling of the atoms. This evaporative cooling of atoms is done by exponentially decreasing the power of lasers used for cooling. The phase transition to a BEC happens at a temperature below ~100 nK. After this, the atoms are confined and the dipole trap is switched off. Then, a 2 ms time of flight is given to the cooled atoms to reduce the effect of mean field energy between the atoms in the condensate. The laser used in this experiment is then frequency offset locked and the two laser beams that used to create the moving optical lattice are setting at two different frequencies with a frequency offset of (15.09 kHz) and are also phase locked to each other. The phase locking is achieved by phase locking the two DDS that generates the two lattice lasers. The two laser beams moving in opposite directions and obtained from diffraction of first order via a pair of acousto-optic modulators, driven by two DDS or arbitrary function generators both of which have a frequency difference of 15.09 kHz and are phase locked. [4]

To date, most precision atomic inertial sensors present in the atom interferometers use Raman transitions (which are coherent two-photon transitions between various hyperfine ground states of alkali atoms in atom interferometers) to create the optical mirrors and beamsplitters needed to build the device. This enables typical optical or microwave pumping methods to be used to measure the number of atoms at the interferometer output. A Bragg transition is same as a Raman transition by the fact that the two states of the interferometer are now distinct momentum classes of the similar atomic state. The mechanism is analogous to atomic state diffraction from the lattice potential created by an optical standing wave. [5]

The light beam is split and sent via two acousto-optic modulators (AOMs) operated using DDS for providing a pair of phase locked optical frequencies needed for making transitions between different states for momentum possible. Each AOM's first order diffraction is merged on a polarising beamsplitter and sent to the experiment along perpendicular axes of an optical fibre that maintains single mode polarization. The light is collimated with an output coupler of an optical fiber to achieve a waist size enough

to interrogate all the atoms in the cold cloud of falling atoms. There is a quarter-wave plate that rotates the polarisation of the light that returns, by 90^0, which allows for proper interference of each Bragg frequency at the position of the atoms with the other and not with itself, resulting in a pair of opposing optical lattices. From an electronics point of view, a DDS drives the two AOMs and controls frequency, amplitude, and phase of the two Bragg beams. The DDS also provides for updation in amplitude, frequency, and phase. [5]

The Bragg diffraction for this experiment is performed in a BEC. The Bragg diffraction beams have to fulfil a certain condition for frequency difference. The condition for the frequency difference between the laser beams for an n^{th} order Bragg diffraction by an optical wave is equivalent to a 2n photon driven Raman process where absorption of photons takes place in one beam and stimulated emission of photons take place in another. As per energy and momentum conservation, [6]

$$\frac{(nP_{recoil})^2}{2M} = 2\pi n\hbar\Delta f \qquad (1.1)$$

Where $P_{recoil} = \hbar k \sin(\theta/2)$ (the recoil momentum of the atom as a result of the Raman process), θ= Angle of incidence, k= 2π/ λ, λ= Wavelength of ^{87}Rb laser = 780.2 nm, M= Atomic mass of ^{87}Rb, Δf= Frequency difference of lasers [6]. On further calculation, we get the required frequency difference for ^{87}Rb as 15.09 kHz.

1.5 USAGE OF DIRECT DIGITAL SYNTHESIS IN ATOM INTERFEROMETER

For the interferometer, a direct digital synthesizer (DDS) is programmed with an input signal and a reference signal that is phase locked to the former signal. It can be controlled by a field programmable gate array (FPGA), that provides adjustment of laser frequency during the set of experiments [3]. Firstly, the order of successive frequencies/ frequency sweeps delivered by the DDS in an experimental cycle is recorded in a computer file that handles the experiment. This file is then transferred from the computer to the soft core processor included on the microcontroller via a USB-UART connector. The data is written to the inbuilt memory by this CPU. After that, a microcontroller, takes data from the inbuilt memory and programmes the DDS using SPI protocol. The DDS is programmed by an external pulse given by the computer at the start of the cycle to ensure that the DDS functions are synchronised with the experimental order. In reality, the DDS switches between a fixed frequency and linear sweeping (for producing linear frequency ramps between two subsequent constant frequencies for short periods of time). [3] It is to be noted that whatever steps are mentioned above, are indirectly evident for this experiment in the thesis.

The measurements for gravity on free falling atoms are performed using Raman interferometry in the most advanced atom interferometer that is realised via an order of three two photon driven Raman transitions (set apart by time T), which split, reroute,

and then merge the atomic wave packets that are then handled by the laser's pulses, resulting in an fringe pattern that has a common minimum which doesn't depend on time assuming systematic bias shifts are negligible. An optical PLL phase locks both lasers so as to get low phase noise. The lasers' frequency difference is swept linearly to account for the Doppler shift resulting from free fall when properly tuned ($2\pi\alpha = 2\pi\alpha_0$ = kg), hence any uncertainty in sweep rate will result in an uncertainty in g. Thus we need a stable sweep rate to measure gravity precisely. For the gravimeter we are sweeping from 80 to 82.5 MHz [7]

Extracting Phase Shift Due to Gravity

The atomic interferometer for the gravimeter can be realised with ultra cold atoms using Bragg diffraction, resulting in a Bose-Einstein condensate. Here, a free falling atom with acceleration g relies on three light pulses of moving optical lattice of far off resonant light in a $\pi/2 - \pi - \pi/2$ order, which is mentioned as follows

1] $\pi/2$ pulse= Splits the condensate into states

2] π pulse= Reflects the two states formed in the $\pi/2$ pulse.

3] $\pi/2$ pulse= Combines the first two states above

This effectively results in phase in the below equation

$$\Phi = n(2k \cdot g - 2\pi\alpha + \varphi_L)T^2 \qquad (1.2)$$

Here T is the interval between interferometer pulses and $k = 2\pi/\lambda$ (λ= Wavelength), is the wave vector, α is the sweep rate. The interferometer phase shows a linear relationship with the Bragg order n. To compensate for Doppler shift, the laser pulses' frequency is swept at a given rate (which is the frequency sweep rate of the DDS) for a particular element, and $\varphi_L = \varphi_1 - 2\varphi_2 + \varphi_3$ indicates phase of the three pulses with respect to a given reference point. By altering sweep rate / phase of the beamsplitter's pulses / DDS, the phase can be chirped to create fringes. Under gravity, the Bragg transition gets modified due to time dependence in the Doppler shift and hence the phase becomes [5]

$$\Phi = n(2k \cdot g - 2\pi\alpha)T^2 \quad (1.3)$$

This results in the relationship between local gravity and sweep rate of the DDS α_0, which precisely compensates the atoms' acceleration for all T, which is given below. [5]

$$g = \frac{2\pi\alpha_0}{2k} = \frac{\alpha_0\lambda}{2} \quad (1.4)$$

In the next chapter, a brief introduction to direct digital synthesis will be given.

Chapter 2 - INTRODUCTION TO DDS

In the previous chapter, the usage of direct digital synthesis in atom interferometry was explained. In this chapter a brief introduction to direct digital synthesis will be given along with it's working in terms of it's individual components. It describes, in general how DDS works

Direct digital synthesis (DDS) is a technique to produce sine (commonly but not limited by sine) waves with much control and precision. In this technique, we generate a time dependent digital wave form and convert it to analogue waveform using digital-to-analogue converter (DAC) [8] and low pass filters. DDS devices provide us with large bandwidth and are easy to synchronise with any reference clock for further phase-noise reduction and keeping other frequency generators and timing sequences in the experiment in sync with DDS clock. The functions for a DDS are primarily digital, allowing it to provide rapid switching between output frequencies, precise frequency resolution, and functioning across a wide frequency spectrum [8]. DDS device is a single-chip IC device that is easy to build and program and is extremely good in phase-continuous and phase-coherent frequency switching modes.

This device also provides almost all the types of modulation like Gaussian or chirped pulses. DDS also provides the infinitely long output with different control (like phase and modulation) at different times so it's very helpful in continuous running experiments.

DDS device has huge applications in current quantum technology, e.g. in quantum control measurement and quantum sensor, quantum computing with neutral atoms and ions and quantum simulator.

2.1 SOME TERMS RELATED TO DDS

Some useful terms related to DDS can help explain the working of the DDS. They are explained below.

2.1.1 Parts of a DDS

Some parts of the DDS are explained below, which include microcontroller, sine lookup table, address counter, digital multiplier, registers, phase accumulator, DAC and low pass filter

1] **Microcontroller**- A microcontroller is a device in an integrated circuit that performs a particular task and runs an application. It often includes memory, programmable

input/output peripherals, and a processor on one chip. Figure 2.1 shows a microcontroller- Arduino Mega 2560

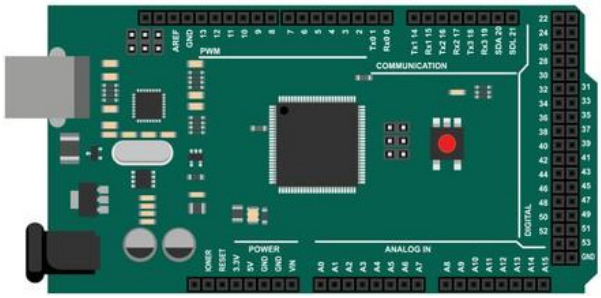


Figure 2.1- A Microcontroller [9]

2] **Sine Lookup Table**- A sine look up table searches for the values of the natural sines of angles in steps of radians. For n bits total number of steps = 2^n. The PROM (programmable read only memory- memory where data can be amended once following production) of a DDS acts as a sine look up table, which stores an integral number of sine wave cycles. [10]

3] **Address Counter**- An address counter counts the number of phase values as input in the sine lookup table, while the sine values are being output. In general, the address counter assesses each and every memory location present in the sine look up table.

4] **Registers**- For a DDS, registers for frequency or phase can be added which provide for pre-programming and content execution of frequency and phase words along a control pin, that also allows for frequency shift keying (FSK) modulation using the programmed one-pin input for suitable “mark” and “space” frequencies. The data in these registers is executed via a dedicated pin on the package, allowing the user to modify an operating parameter without having to go through the control interface instruction cycle. [10]

5] **Digital to Analog Converter**- A **Digital to Analog Converter (DAC)** transforms a digital input signal that is represented with a binary code, into an analog signal as output. It consists of several binary inputs (usually a power of two) and a single output. For a DDS, the contents of the sine lookup table are sent to a high-speed DAC which gives as output an analog sine wave using the digital input words from the sine lookup table. [11] Figure 2.2 shows a schematic of a DAC.

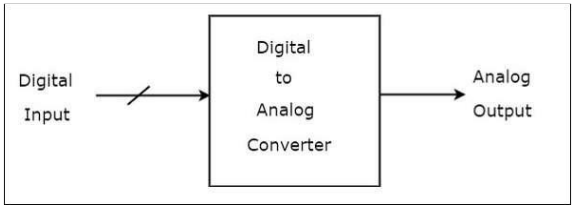


Figure 2.2- A DAC [11]

6] **Digital multiplier**- Digital multiplier is a device placed between the sine lookup table and the DAC, so that it becomes possible for amplitude modulation of the analog output. Length of word provided to this device gives us the output amplitude step size resolution. [10]

7] **Phase accumulator**- Phase accumulator is a device that acts as a modulo N counter with 2^N digital states that are increased every time a clock pulse is input. The increment size changes with the tuning word value given to the accumulator adder stage which then fixes the step size; hence this will determine output waveform frequency. Typically, the phase accumulator's allowed bit range is from 24 to 48 bits. [12]

One way of understanding the phase accumulator's working is to compare it with that of a phase wheel, such that the phase states of the phase accumulator are periodic and may be represented as a set of points on a circle, that can be considered as corresponding to an analogous point on a sine wave revolution; so the projection to the RHS of the phase wheel is the corresponding analog output for every state (as shown in Figure 2.3). The fact that this is a discrete device, the analog output will be stalled in its present state till the clock moves the phase wheel to its next state. The output waveform is made up of one complete revolution of the sine wave that has been quantized [12]. This can be assumed as a vector rotating round a wheel, assuming that the corresponding analog output is generated. A revolution of the vector round the phase wheel is equal to a full revolution of the analog output. A phase accumulator is used to supply linear rotation of the vector round the phase wheel. The phase accumulator's contents correspond to the projected points on the period of the output sinusoid. The resolution, N, of the phase accumulator determines the number of these projected phase points that must be unique and are contained in the "wheel". By using phase accumulator function for digital signal chain, the DDS structure is like an oscillator operated numerically which is a highly versatile DDS core. [10]

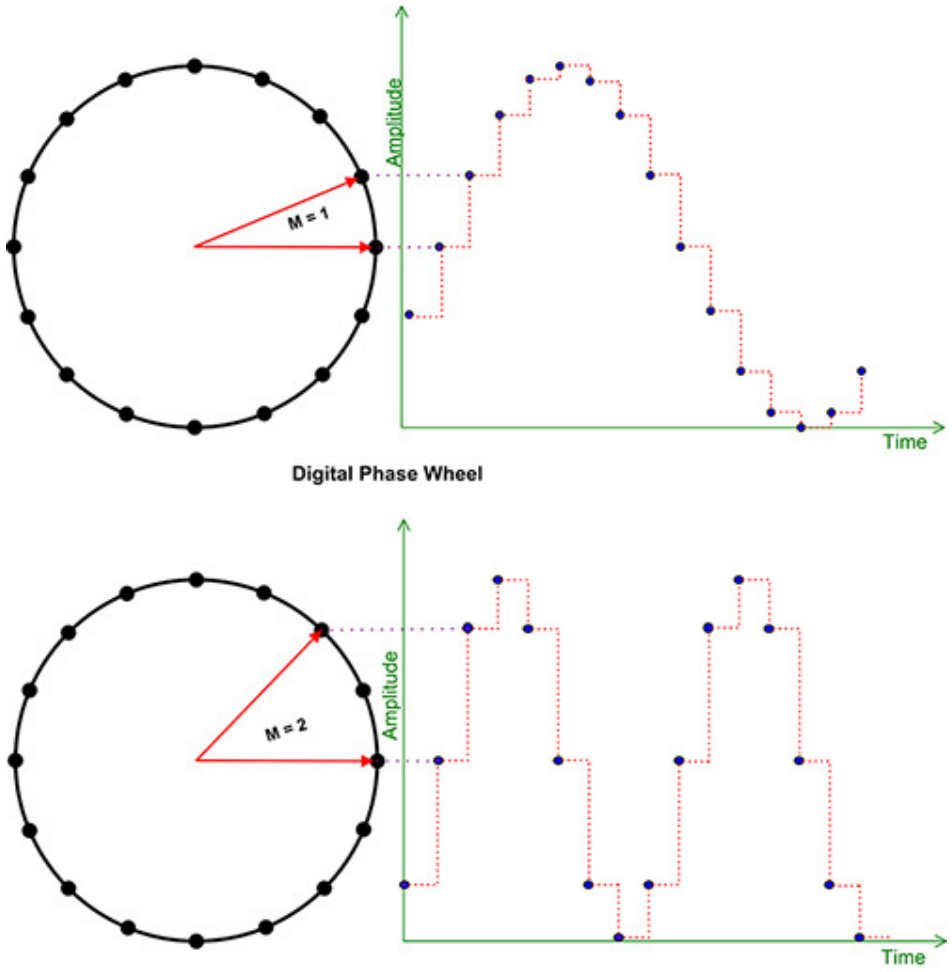


Figure 2.3- The Phase Wheel [12]

8] **Low Pass filter**- A **low pass filter** (LPF) is a device that passes the frequency component of a signal below a certain value and suppresses the frequencies above that certain value to almost zero. It is defined using a specific cut-off frequency. The simple circuit of an LPF consists of a resistor and a capacitor (see Figure 2.4). Its working is dependent on the impedance of the capacitor that changes with frequency of the signal (say f). The formula for impedance Z is $(\omega C)^{-1}$, with $\omega=2\pi f$ and C the capacitance of the capacitor so lower frequency implies more impedance. The cut-off frequency f_0 for the LPF is given by this formula, with R being the resistance of the resistor.

$$f_0 = \frac{1}{2\pi RC} \quad (2.1)$$

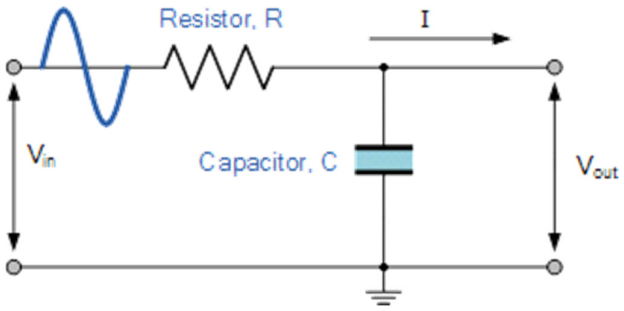


Figure 2.4- A low pass filter [13]

For a DDS, the clock signal that is provided is what's driving the change in output, hence a low pass filter will be designed for this frequency of the DDS (see Figure 2.5). For usual DDS applications, one uses LPF to remove from the output spectrum any image response effects present. To keep low-pass filter threshold requirements reasonable and filter design simple, an accepted guideline is to restrict the output frequency bandwidth to about 40% of reference clock frequency using a cheap low pass filter.

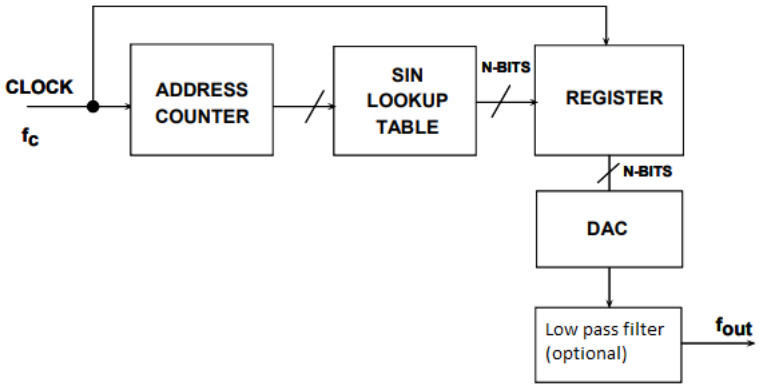


Figure 2.5- A Fundamental DDS System [14]

2.1.2 Most Significant Bit and Least Significant Bit

For any number in binary form, the most significant bit (MSB) has the biggest place value while the least significant bit (LSB) has the smallest place value. Ex- For 1000, the MSB is 1 and the LSB is 0. The fact that these values are mainly used in computation, electronics and similar related areas, these concepts hold importance, especially when it comes to transmitting binary values. [15]

2.1.3 Frequency Tuning Word

The DDS output frequency can be modified by changing the count of phase/ frequency bits that are scrutinized, through a parameter which is called a frequency tuning word (FTW). Each output frequency value for the DDS has a corresponding value of the frequency tuning word which is stored in the DDS registers. The frequency tuning word can be in binary or hexadecimal format. For any DDS the frequency tuning word is given by this formula

$$FTW = \frac{Output\ Frequency}{Input\ Frequency} \times 2^n \quad (2.2)$$

Where n is the number of bits for the DDS.

2.1.4 SPI

Amongst the most demanding interfaces involving microcontrollers and various ICs (sensors, ADCs, DACs, registers, SRAM etc.), is serial peripheral interface (SPI). Here, synchronous data from master/ slave is synced using the rising/ falling edge of the clock pulses. This data can be sent together by the master & slave and in both directions, so it's a full duplex interface as well. One can use either 3-wire/ 4-wire SPI interface, however for this project the latter is used, which uses 4 signals (see Figure 2.6): the clock pulses (SCLK), Chip select (CS), master output slave input (MOSI) & master input slave output (MISO). [16]

Here, master generates clock pulses. Data exchanged between master and slave is synced with the master's clock. SPI has just one master and multiple slaves. From master, the CS signal (an active low pulse and set to high to disable the slave from SPI bus) selects the slave. As several slaves are used, corresponding individual CS signals per slave are needed from the master. For sending data, MOSI & MISO are used with MOSI sending data from master to slave and MISO sending data from slave to master. [16] For this project the DDS is the slave while the microcontroller is the master.

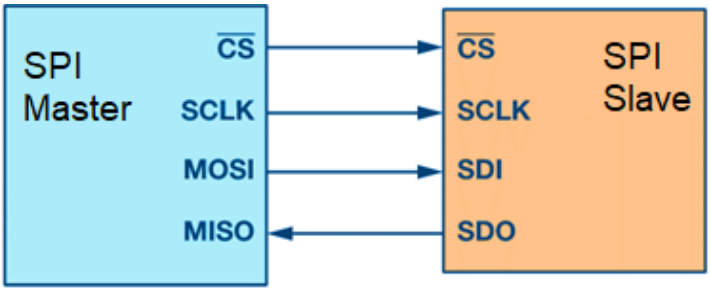


Figure 2.6- 4 wire SPI

2.1.4.1 Working of SPI

For SPI communication to commence, master has to transmit the clock pulses and decide the slave as it activates the CS pulse. The CS is an active low pulse; so master has to send a logic 0 on this signal to decide the slave. Both master and slave transmit data together using the MOSI and MISO lines (since SPI is a full duplex interface). The serial clock edge synchronises shifting and reading/ writing of data depending on how the SPI provides for data reading/ writing and/or shifting using either the clock's rising or falling edge. [16]

2.1.4.2 Clock Polarity and Clock Phase for SPI

In SPI, clock phase (CPHA) and polarity (CPOL) can be selected by master. In the idle state (i.e. the time period between the following- from when CS is high and falling to low, to when CS is low and rising to high), CPOL bit decides polarity of the clock pulses while clock phase is decided by the CPHA bit. Data is sampled and/or shifted using the rising/ falling edge of clock pulses, based on CPHA bit. The master has to choose clock polarity and phase in accordance with the slave's needs. There are 4 SPI modes based on the CPOL and CPHA bit setting which are as follows [16]

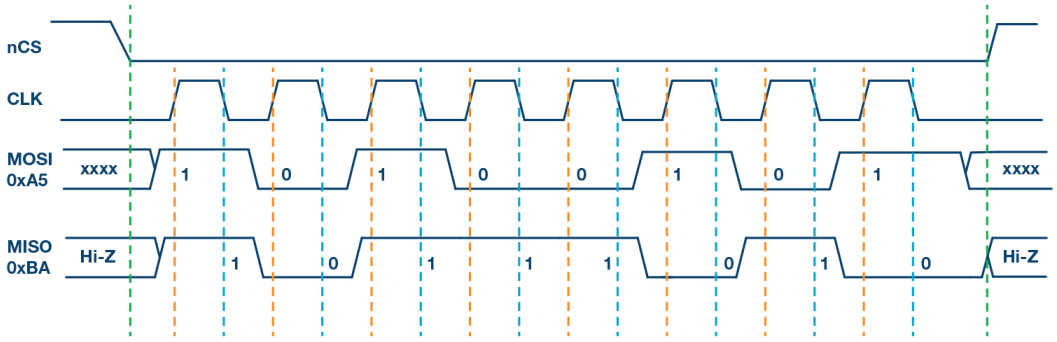


Figure 2.7- SPI Mode 0 [16]

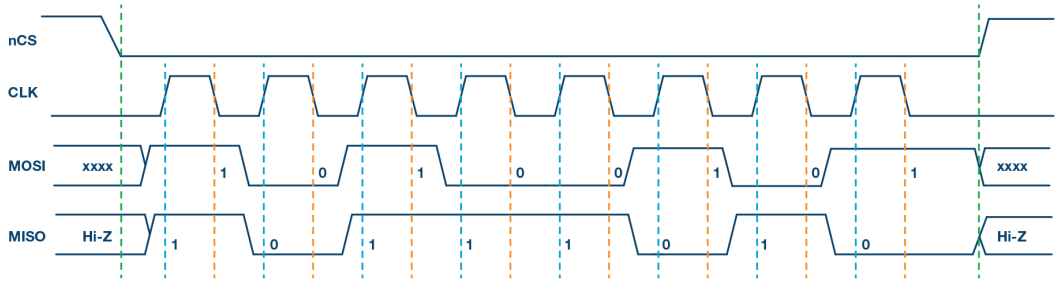


Figure 2.8- SPI Mode 1 [16]

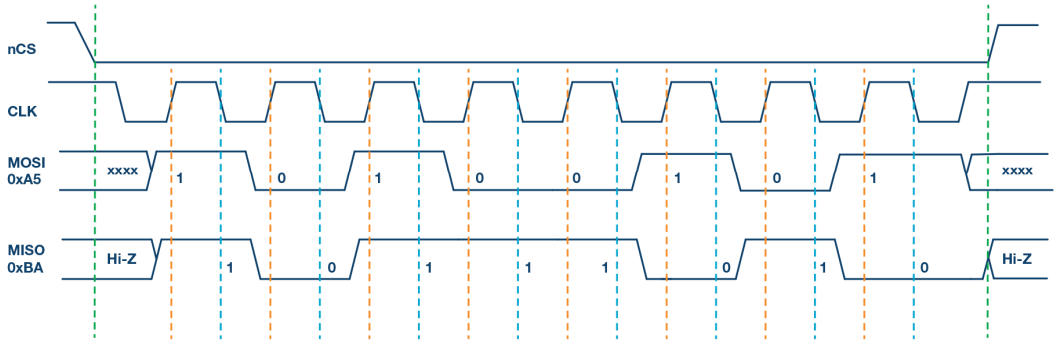


Figure 2.9- SPI Mode 2 [16]

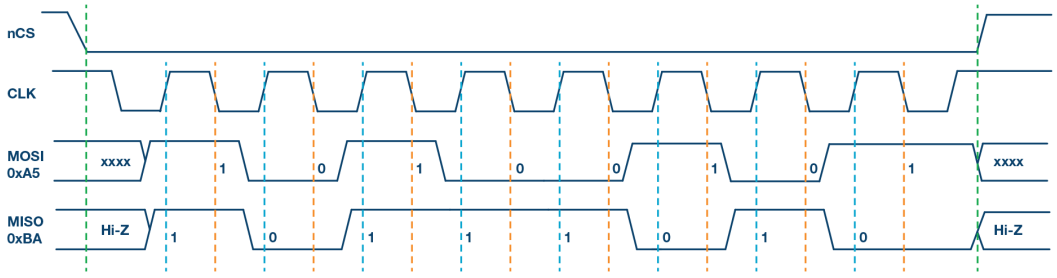


Figure 2.10- SPI Mode 3 [16]

1] **SPI Mode 0**- When both the CPOL and CPHA are low, the clock polarity in the idle state is a logic low and this results in reading/ writing of data at the rising edge and shifting out of data at the falling edge of CLK as shown in Figure 2.7 [16]

2] **SPI Mode 1**- When the CPOL is low and CPHA is high, the clock polarity in the idle state is a logic low and this results in reading/ writing of data at the falling edge and shifting out of data at the rising edge of CLK as shown in Figure 2.8. [16]

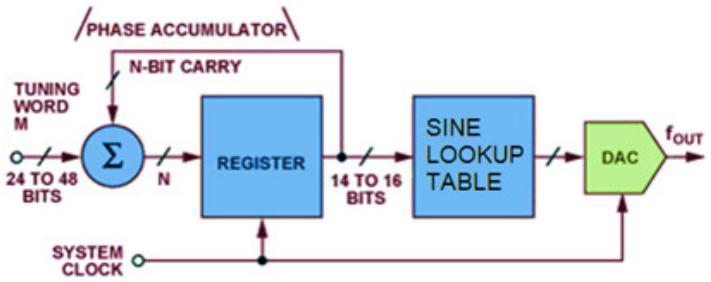


Figure 2.11- Another interpretation of DDS [17]

3] **SPI Mode 2**- When the CPOL is high and CPHA is low, the clock polarity in the idle state is a logic high and this results in reading/ writing of data at the rising edge and shifting out of data at the falling edge of CLK as shown in Figure 2.9. [16]

4] **SPI Mode 3**- When both the CPOL and CPHA is high, the clock polarity in the idle state is a logic high and this results in reading/ writing of data being at the falling edge and shifting out of data at the rising edge of CLK as shown in Figure 2.10. [16]

2.1.5 Phase Locked Loop (PLL)

For a DDS, a phase locked loop (PLL) is a circuit which uses a voltage controlled oscillator which always alters the frequency to make it equal to the input frequency of the DDS. In general, for the DDS, a PLL is a closed feedback loop, that is sensitive to changes in frequency as well as phase. PLL consists of both analog and digital circuit components connected in a negative feedback configuration in order to reduce noise or errors in phase between input/ output frequencies. When the phase difference between both input and output frequencies is zero, the system is said to be locked, hence the name phase locked loop. Apart from setting up input/ output frequencies, this device can also use the phase relationship for these frequencies to generate a suitable voltage for control. Here are the components for a PLL (see Figure 2.12). [18]

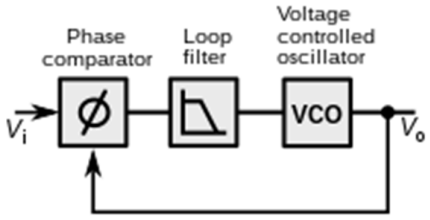


Figure 2.12- A Phase Locked Loop [18]

1] **Phase Detector/ phase comparator/ mixer**- This compares the phases of the input/ output frequencies and induces a potential difference using their phase difference and multiplies the reference input with the voltage controlled oscillator's output. [18]

2] **Low Pass Filter (LPF)-** For a PLL, this is a loop filter that rejects high frequency components to smoothen the signal and make the signal as close as possible to a DC signal. [18]

3] **Voltage controlled oscillator (VCO)-** This generates a sine wave whose frequency matches the central frequency as given by the LPF. [18]

Chapter 3 - COMMONLY USED DDS

In this chapter, the commonly used DDS of this project will be discussed and explained about in detail along with their working. This chapter has a detailed description of many DDS chips

3.1 AD 9850

This module of DDS generates sine and square waves and is provided with a 125 MHz oscillator (usually made of quartz). The innovative high speed core of this particular DDS gives a frequency tuning word of 32 bits, resulting in a frequency resolution output of 29.1 mHz for a 125 MHz clock oscillator. [19]

For AD9850, frequencies up to 62.5 MHz can be generated, although I was able to generate frequencies of up to 10 – 30 MHz). The square waves were not perfect square waves primarily due to the fact that the square wave is expressible a Fourier series, with the DDS unable to process an infinite summation. Assuming f(x) is the requisite function for the square wave.

$$f(t) = \sum_{n=1}^{\infty} \frac{1}{2n-1} sin\,(2n-1)\omega t \quad (3.1)$$

Proof for the Fourier Series

The original function for the square wave is sgn(sin ωt), where sgn(x) is called signum function, the ratio of the absolute value of x to the real value of x, it's output is 1 for positive numbers, 0 for zero, and -1 for negative numbers. Then Fourier series is given by

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n\, cos \frac{2n\pi t}{T} + \sum_{n=1}^{\infty} b_n\, sin \frac{2n\pi t}{T} \quad (3.2)$$

Since cos(x) is an even function and sgn(sin(x)) is an odd function, $a_n = 0$, also $a_0 = 0$ as the value sgn(sin(x)) is 1 for (0, π) and -1 for (π, 2π). Now coming to b_n

$$b_n = \frac{2}{T}\int_0^T sgn(sin\,\omega t)\, sin \frac{2n\pi t}{T} = \frac{2}{T}\left[\int_0^{\frac{T}{2}} sin \frac{2n\pi t}{T}\, dt - \int_{\frac{T}{2}}^{T} sin \frac{2n\pi t}{T}\, dt\right] = \frac{4}{n\pi} \quad (3.3)$$

But f(t)= -f(t+ T/2), since signum function has opposite signs halfway in a full cycle.

$$sin\frac{2n\pi t}{T} = -sin\left(\frac{2n\pi t}{T} + n\pi\right) \quad (3.4)$$

If n is even, then

$$sin\frac{2n\pi t}{T} = -sin\frac{2n\pi t}{T} = 0 \quad (3.5)$$

If n is odd then this becomes an equality, hence the odd terms are considered

Another possible reason why the square waves are not perfect is an effect which is called the ringing effect. **Ringing effects** are faults in the signal that cause it to appear as spurious signals around sharp transitions in the signal. The major cause of ringing is a signal that is band limited (particularly, does not have high frequencies) or has been processed through a low pass filter, the latter being predominantly employed in DDS. The ripples in the analogue function, which is the impulse response, are the cause of this form of ringing in the time domain.

3.1.1 Parts of AD9850

These are the various parts of AD9850 in the form of pins, as shown in Figure 3.1.

1] **VCC**- This connects the DDS to power source, hence it's a voltage supply pin. [20]

2] **W_CLK**- This generates the reference clock pulses. It is also used to supply the frequency / phase/ control words either serially or parallelly. [20]

3] **DATA**- This generates or serially loads the data [20]

4] **FQ_UD**- This functions as a chip selector, when it is on (or chip selector is set to 1) the clock pulses and data are not generated. From this pin, the DDS updates frequency (or phase) stored in the input data register using the rising edge of the clock pulses; then resets the DDS. [20]

5] **RESET**- This pin is for resetting the DDS by cleaning all the registers. Also called master reset, when this is set to high, it clears all registers (other than the one for input) and the DAC's output following an extra number of clock cycles. [20]

6] **GND**- This connects the DDS to the ground

7] **SQ Wave out**- Generates output (for 1st pin)/ complementary output (for 2nd pin) of the comparator. [20]

8] **Sine Wave out**- Generates analog output (for 1st pin) and complementary analog output (for 2nd pin). [20]

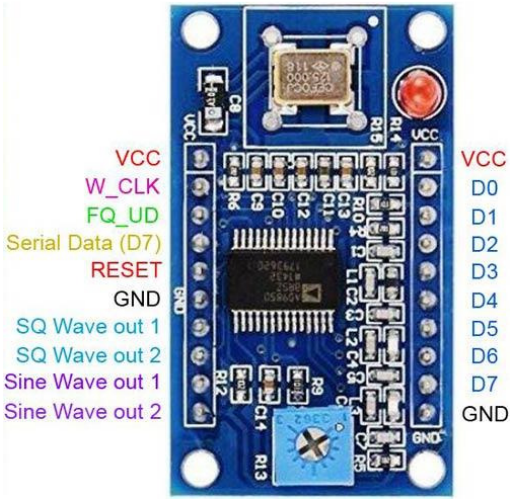


Figure 3.1- AD9850 DDS [20]

3.1.2 Working

The AD9850 works on DDS technology like an oscillator operated numerically, such that when the AD9850 is referenced to a reliably accurate reference clock source, it results in spectrally pure, analog output as sine waves (that are programmable) which can be of use as a frequency source, or later transformed to a square wave for use as a precise clock generator. An on board 10-bit high speed D/A converter converts the digital sine wave to analogue form, and an on-board high speed comparator converts the analog output to a low jitter output square wave such that TTL/ CMOS is compatible. To generate square waves with low noise, a high speed comparator is present in the DDS that is potentially setup to allow the externally filtered output of the DAC, such that the device's usage in precise clock generator applications is boosted. The tuning/ modulation words for phase / frequency/ control are loaded into this DDS using parallel/ serial loading format. [19]

1] **Parallel loading**- The parallel load format uses 5 iterations of an 8 bit word (byte) for control. First 8 bits control phase modulation, triggering of off and formatting of load; while the next 32 bits include the frequency tuning word. The registers are addressed and set by W_CLK & FQ_UD signals. Setting up the data word for control of 40 bits to the DDS and resetting the address pointer is done by rising edge of FQ_UD while loading the data of 8 bits and shifting the pointer to the following register is done by the next W_CLK rising edges. After 5 loads, the W_CLK edges are not considered till reset or when the rising edge of FQ_UD resets the address pointer. Parallel loading transfers 8 data bits per I/O clock cycle, but at the cost of the fact that several pins are required on the devices. [19]

2] **Serial loading**- This is achieved using only a single pin via a 40 bit serial data stream. Shifting of data of a bit via the 40 bits of programmed information is done by the upcoming rising edges of W_CLK, but after this is done, the output frequency (or

phase) is updated for which FQ_UD pulse is necessary. Serial loading offers the advantage of simplicity but at the disadvantage of low speed. [19]

The AD9850's output waveform when changed is continuous with regards to phase. The circuitry is essentially a digital frequency divider function with increasing divisions set by the W_CLK frequency divided by the 2^N, N being the count of bits for the tuning word. The phase accumulator is an N-modulus counter that increases the value stored in it every time a clock pulse is received and when the counter is filled with data, it wraps around. Hence the resulting output is contiguous. [19]

The tuning word value gives the frequency output of the DDS (Figure 3.3) and hence this output frequency increases with the increase in the tuning word's value. The sampling rate and the duration between samples at the output are constant. The output frequency changes with the gain of the tuning word, so as the value of the tuning word grows, lesser are the steps in each output cycle, hence raising the frequency. One can increase the tuning word value as long as there are at least 2 samples per cycle, bringing the DDS output frequency to Nyquist frequency, or half the system clock rate. The DDS is designed to have an output frequency strictly below the Nyquist limit. The tuning word is calculated by this formula given n, the number of bits, which is also the length of the phase accumulator. [19]

$$Frequency\ Tuning\ Word\ (FTW) = Frequency\ of\ Wave\ \times \frac{2^n}{Frequency\ of\ oscillator}$$

$$= Frequency\ of\ Wave\ (Hz) \times \frac{2^{32}}{125000000} \quad (3.6)$$

The analog output of this DDS is a sampled signal, so the output spectrum comprises of fundamental and aliased signals (pictures) that are present at multiples of reference clock frequency ± selected output frequency [19].

3.1.3 Uses of AD9850

AD9850 functions as a clock generator. In this case the designated output frequency is restricted to within 33% of the reference clock frequency, so as to refrain from giving aliased signals that fall in proximity with, the output band of interest; thus, the sophistication and price of the external filter necessity for this application are reduced. This DDS is also a great technique of generating the read/ write clock to the ADC, particularly when the ADC read/ write frequency must be set by software and fixed to the system clock. [19]

3.2 AD9910

The AD9910 is a DDS with a built in 14 bit DAC. It can handle sample rates up to 10^9 samples per second. This employs innovative, patented DDS technology that helps to significantly minimize power usage without affecting performance. The DDS and DAC together can create controllable, high frequency sinusoidal output that can be changed, and it can generate frequencies of up to 400 MHz. The DDS parameters (i.e. frequency, phase, and amplitude) are available to the user while the 32 bit accumulator allows for rapid frequency hopping and frequency tuning resolution (that is of 230 mHz) with a sampling rate of 1 GSPS. The DDS also allows for rapid phase & amplitude switching. [21]

To control its internal control registers, the AD9910 is programmed via a serial I/O interface. This DDS comprises of embedded RAM that uses flip flop like circuitry and provides a variety of modulations for frequency, phase, and/ or amplitude. This DDS may operate in a digitally controlled, user defined ramp mode wherein the frequency, phase or amplitude can be modified with time linearly. [21]

A reference signal is generated by the direct digital synthesiser (DDS) block whose attributes (frequency, phase, amplitude etc.) are used by the DDS parameter's (frequency, phase offset, amplitude etc.) control inputs. The frequency tuning word (FTW) determines it's output frequency (f_{OUT}). The relationship among f_{OUT}, FTW (M), and f_{SCLK} is [21]

$$f_{OUT} = M \times \frac{2^n}{f_{SCLK}} \quad (3.7)$$

where M, the frequency tuning word is a numerical value from 0 to $2^{31} - 1$. [21]

3.2.1 Main Parts of AD9910

The main parts of AD9910 are shown in Figure 3.2 but will only mention the parts used in the project

1] **SCLK (Serial Clock)**- This transfers data back and forth to the AD9910 at once and powers the internal machines for state. [21]

2] **CS**- CS is an active low input allowing for many devices share a serial line for communications. When this input is high, both SDO (serial data output) and SDIO (serial data input/ output) pins will have high impedance. If during a communications cycle, CS is changed to high, the cycle is halted until CS is reenabled to low. [21]

3] **SDIO**- Through this pin, data is given to the AD9910, although it can be utilised as a two directional line for data. The default setting is cleared, which makes this pin two directional. [21] This pin acts as a MOSI for SPI when a microcontroller is used.

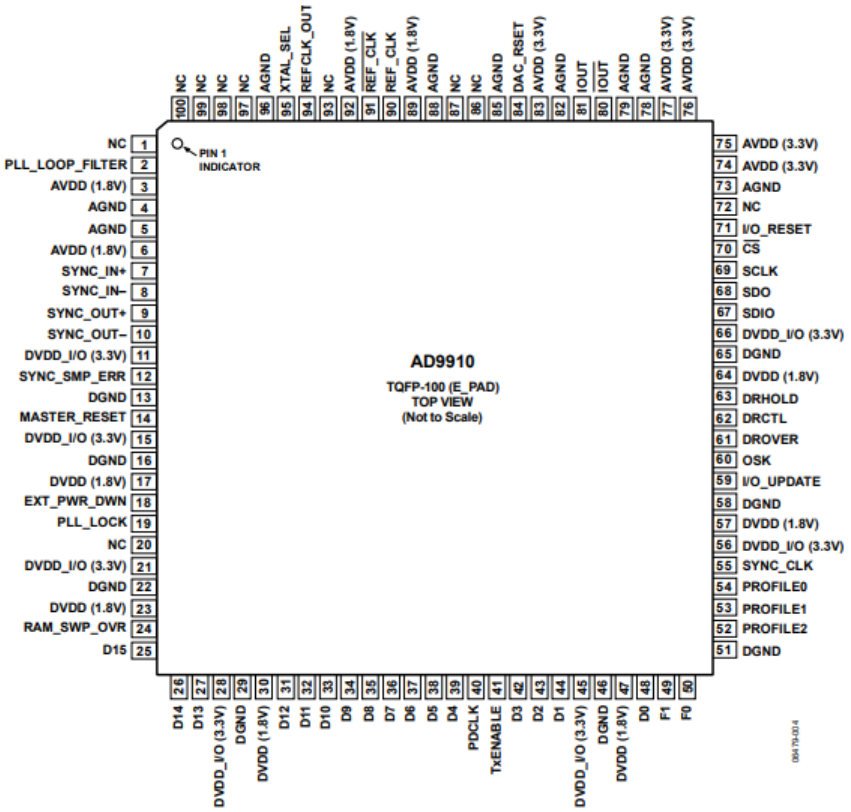


Figure 3.2- AD9910. Note- The pad exposed to be connected to ground & NC= Do not connect [21]

4] **SDO**- This pin is for reading data for protocols which employ different lines for transmission and receival of data. Whenever the DDS functions in single two directional I/O mode, this pin has high impedance, hence it doesn't output data. [21] This pin acts as a MISO for SPI when a microcontroller is used

5] **I/O RESET**- The current communication cycle is terminated by an active high on this pin. Once this pin returns Logic 0 (or low), another communication cycle can commence, and the writing of the instruction byte to the DDS takes place. [21]

6] **I/O UPDATE**- This pin starts the transmission of information given from I/O port buffer to active registers and works using the rising clock edge. This pin is for updating DDS parameters. It's an input/ output pin, but that depends on how the active internal bit of this pin is programmed. [21]

7] **DRCTL**- This pin decides the digital ramp generator's (DRG) slope. A high on this pin indicates positive slope and a low on this pin indicates negative slope. [21]

8] **DRHOLD**- This pin stops the operations of the digital ramp generator (DRG) in the current state. [21]

9] **MASTER RESET**- This resets the DDS by clearing elements that store data and setting registers to default values. [21]

10] **PLL LOCK**- This is for phase locking the DDS. A high on this pin enables phase locking of the DDS (and the frequency multiplier); a low on this pin disables both of them.

11] **EXT_PWR_DOWN**- A high on this pin externally powers down the DDS depending on how it is programmed

12] **P0, P1, P2**- They are the frequency profile pins for single tone mode (see section 2.2.2.1). Any changes in this pin causes contents (related to I/O) to be sent to the corresponding registers.

13] **SYNC_CLK-** This runs at one fourth of the system clock frequency, it's rising edge is used for I/O_UPDATE & profile changes.

14] **Tx_ENABLE**- A high on this pin enables the continuous mode (or burst mode) which is used in the single tone mode.

3.2.2 Working

AD9910 DDS has four operational modes- single tone mode, digital ramp modulation mode, RAM modulation mode and parallel data modulation mode (only first two will be explained). There is also a output shift keying (OSK) functionality present separately. The DDS's amplitude parameter is the only one that is modified in the digital ramp mode when used by this functionality. OSK has more priority to drive the DDS amplitude parameter, As a result, when OSK is enabled, only OSK can drive the amplitude. [21]

3.2.2.1 Single Tone Mode

The DDS parameters are given straight from the registers that are programmed for this mode. This involves usage of profiles which are stand-alone registers that store control settings for the signal. There are 8 profile registries accessible each of which can be accessed independently. To pick the desired profile, one can use one or more of these 3 pins. At the following rising edge of SYNC_CLK, any alteration in the state of the profile pins causes an update in the DDS with parameters defined by the profile decided. [21]

The output of single tone mode is a sine wave whose frequency is given by the formula in terms of the frequency tuning word. On a Fourier representation, this is represented by a peak of a Dirac delta function at that particular frequency (usually on oscilloscope

or spectrum analyser). Figure 3.5 gives the resulting output of single tone mode for a 300 MHz signal, however for phase locking the output with a sinusoidal signal whose frequency found out to be 40 MHz, an 80 MHz signal in single tone mode is fired through the code (see section 3.2.5). A function generator was used to generate the sine wave of the required frequency to be phase locked with the 80 MHz signal.

3.2.2.2 Digital Ramp Modulation Mode

The DDS parameters are obtained straight from the digital ramp generator (DRG) for this mode (see Figure 3.3). The serial I/O port is utilised to control the ramp generation parameters, that allow for control of the ascending and descending ramp slopes. The higher & lower ramp boundaries, as well as the step size/ rate of the ascending/ descending ramp slopes, are all controllable. This ramp is generated digitally with a 32-bit output resolution that can be configured to denote the DDS parameters (frequency/ phase/ amplitude). All 32 bits are used when representing frequency, but when representing phase or amplitude, only 16 or 14 MSBs are used. The DRCTL pin controls the ramp direction. (ascending or descending). Using an extra pin, one can temporarily halt the ramp generator in its current condition (DRHOLD). The DRG involves 9 control register bits, 3 external pins, two registers of 64 bits, and a register of 32 bits. The main control for the DRG is a bit to enable the device. [21]

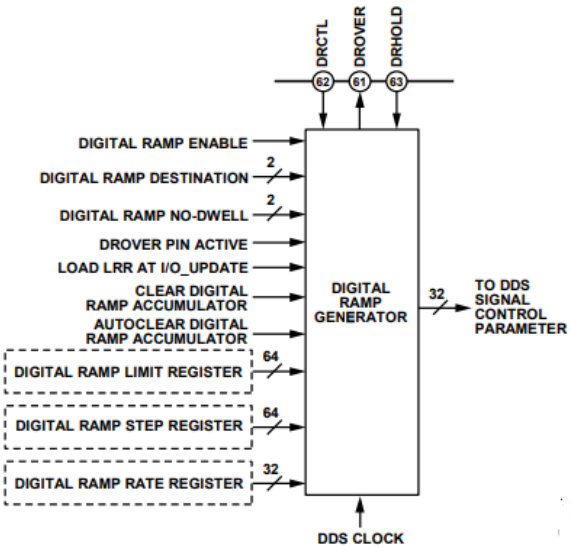


Figure 3.3- A schematic of the digital ramp generator. Note that the boxed areas of this device are primarily used in this project [21]

The output of the digital ramp mode is a sine wave whose frequency is increasing/ decreasing linearly with time (usually we do the former) at a constant rate called the

chirp rate. The function for the digital ramp mode is given as follows assuming it is in the range of frequencies that one is sweeping across.

$$V(t) = V_0 \sin(\omega(t)\, t + \varphi) \quad (3.8) \quad where\ \omega(t) = \omega_0 + \alpha t \ \ and\ \omega_0 \leq \omega(t) \leq \omega_1 \quad (3.9)$$

$$V(t) = V_0 \sin(\alpha t^2 + \omega_0 t + \varphi) \quad (3.10)$$

In equations (3.8), (3.9) and (3.10) above α= Chirp Rate of DDS, ω_0= Initial Frequency, ω_1= Final Frequency, t= Time, φ= Phase Difference (if any)

On a Fourier representation this is represented by the shift in peak of a Dirac Delta Function towards the right (usually on an oscilloscope or a spectrum analyser), since the fast Fourier transform of a pure sine wave is a Dirac Delta function. In this project, the sweeping was carried out from 80 MHz to 82.5 MHz at a chirp rate of 25.1 MHz/s, this was phase locked in the same manner as single tone mode, with the same frequency used on the arbitrary function generator (i.e. 40 MHz).

Experimental Data for Digital Ramp Mode

Following were the ten observations noted for the time taken to sweep from 79.99999998 to 82.50000015 MHz (accuracy taken to 8 decimal places). Each time interval noted has an uncertainty of 0.2 ms.

Table 3.1- Experimental Data for Digital Ramp Mode

Observation Number	Time (ms)
1	100
2	100
3	99.6
4	99.6
5	99.4
6	99.8
7	99.8
8	99.8
9	100
10	99.8

3.2.3 Serial Programming of AD9910

The serial port for this DDS is versatile and synchronous for serial communications, such that it can interact with a wide range of industry-standard microcontrollers and microprocessors. Most synchronous transmission formats are supported by serial I/O. This interface allows us to read/ write to all of the registers. Furthermore, the port for

serial interface can be set as a one pin input/ output (SDIO) or as two one way pins for input/ output (SDIO/ SDO). With the AD9910, two alternative pins (I/O_RESET, CS) provide an ability to design the system freely. [21]

A serial communications cycle consists of two stages, starting with the instruction phase, for which the writing of the instruction byte to the DDS takes place (see section 3.2.3.1 for more). [21]

In the next (i.e. 2nd) phase of a write cycle, data flow from serial port controller to serial port buffer takes place. Every register being read determines the number of bytes transmitted. Each bit of data is stored in the registers using rising edge of SCLK. The serial port controller should access every content in the registers; else, it will be not placed in sequence for the upcoming communication cycle. To stop and reset the communication cycle, one uses I/O_RESET. The instruction byte then comes after an I/O_RESET. Finally the serial port controller for this DDS expects the following eight rising SCLK edges to end a communication cycle and represent the instruction byte for the next cycle. [21]

The inactive programmed data is saved in the serial port buffer after a write cycle. The I/O UPDATE can be issued at any moment during the communication cycle or after all serial operations have completed. Furthermore, any alteration in profile pins can result in an I/O upgrade. Phase 2 of a read cycle is identical to that of a write cycle, except for the following [21]

1. Instead of the serial port buffer, data is read from the active registers.

2. Data is sent out on the SCLK's falling edge.

It should be remembered that any profile register must be read back using the three external profile pins. [21]

3.2.3.1 Instruction Byte

As given by Table 3.2, the instruction byte comprises of the following info. [21]

1] Read / Write— 7th bit (i.e. D7) of this byte decides if a read/ write transfer of data takes place following writing of the byte. Logic 1 represents a read operation and cleared represents that a writing operation has taken place. [21]

2] The instruction byte's bits X, X— D6 (6th bit), and D5 (5th Bit) are disregarded. [21]

3] A4, A3, A2, A1, A0—D4, D3, D2, D1 & D0 for an instruction byte determine the register to be used when data is sent as a part of the communication cycle. [21]

Table 3.2- Instruction Byte [21]

D7	D6	D5	D4	D3	D2	D1	D0
Read / Write	X	X	A4	A3	A2	A1	A0

3.2.4 Register Descriptions for AD9910

Following tables give the register descriptions of AD9910 for all registers used in the different modes of operation for this project (along with that of the control function registers used).

Table 3.3- Single Tone Mode Register Descriptions (Address- 0x0E to 0x15) [21]

Bits	Parameter	Register Description
63 : 62	Open	
61 : 48	Amplitude Scaling Factor	Controls output amplitude
47 : 32	Phase Offset Word	Controls output phase
31 : 0	Frequency Tuning Word	Controls output frequency

Table 3.4– Digital Ramp Limit Register Descriptions (Address- 0x0B) [21]

Bits	Parameter	Register Description
63 : 32	Digital Ramp Upper Limit	Gives the upper limit of the digital ramp
31 : 0	Digital Ramp Lower Limit	Gives the lower limit of the digital ramp

Table 3.5- Digital Ramp Step Size Register Descriptions (Address- 0x0C) [21]

Bits	Parameter	Register Description
63:32	Digital ramp step size (increasing)	Gives step size of an increasing ramp
31:0	Digital ramp step size (decreasing)	Gives step size of a decreasing ramp

Table 3.6- Digital Ramp Rate Register Description (Address- 0x0D) [21]

Bits	Parameter	Register Description
31:16	Digital Ramp Negative Slope	Gives the negative chirp rate of DDS using the time taken for a decreasing step size
15:0	Digital Ramp Positive Slope	Gives the positive chirp rate of DDS using the time taken for an increasing step size

Table 3.7- Register Descriptions for Control Function Register 1 (Address- 0x00) [21]

Bits	Parameter	Register Description
31	Enable RAM	0= Disables RAM (default), 1= Enables RAM

Bits	Parameter	Register Description
30:29	RAM playback destination	Not applicable as RAM was not enabled
28:24	Open	
23	OSK external control	0= OSK (output shift keying) pin inactive 1= OSK pin active
22	Inverse sinc filter	0= Disabled (default), 1= Enabled
21	Open	
20:17	Profile control	Not applicable since RAM not enabled
16	DDS sine output	0= Cosine output (default), 1= Sine output
15	Ramp timer loading @ I/O UPDATE	0 = default working, 1 = ramp timer loaded whenever DDS updates are given/ profile change occurs
14	Auto clear DRG accumulator	0= default working, 1= DRG accumulator is reset, then auto resumes default working whenever an update to DDS is given/ profile change occurs till this bit is unset.
13	Auto clear phase accumulator	0= default working, 1= Resets phase accumulator in sync with DDS updates/ profile change
12	Clear DRG accumulator	0= default working of DRG accumulator 1= DRG accumulator stays reset until this bit is unset.
11	Clear phase accumulator	0= default working of DDS phase accumulator 1= resets the DDS phase accumulator.
10	ARR (amplitude ramp rate) load @ I/O UPDATE	0= Default working, 1= OSK ARR timer reloaded when updates to DDS are given / profile change occurs.
9	OSK	0= Inactive (default), 1= Active.
8	Auto OSK	0= Manual operation (default), 1= automatic operation
7	Digital power off	0= clock signals for digital core are on (default). 1= clock signals for digital core are inactive.
6	Power off DAC	0 = DAC is active (default). 1 = DAC is inactive.
5	Power off REFCLK input	0= REFCLK input on (default). 1= REFCLK input inactive
4	Power off Auxiliary DAC	0= auxiliary DAC on (default). 1= auxiliary DAC off
3	Control external power down	0 = Affects full power down (default). 1 = Affects quick recovery power down.
2	Open	
1	SDIO input only	0= Two direction operation for SDIO (default). 1= Input only, for SDIO
0	LSB first	0= MSB first (default), 1= LSB first

Table 3.8- Register Descriptions for Control Function Register 2 (Address- 0x01) [21]

Bits	Parameter	Register Description
31:25	Open	
24	Amplitude scaler (for single tone mode profiles)	0 = amplitude scaler is disabled and powered down (default). 1 = amplitude scaler scales amplitude using the scaling factor using the enabled profile.

Bits	Parameter	Register Description
23	Enable Internal I/O UPDATE	0= External trigger of I/O UPDATE (input) (default). 1= Internal trigger of I/O UPDATE pin (output)
22	SYNC_CLK enable	0= disables SYNC_CLK, 1= enables SYNC_CLK pin gives a signal of frequency ¼ f_{SYSCLK} (default).
21:20	Inverse sinc filter	0= Disabled/ bypassed (default), 1= Enabled
19	Digital Ramping	0= Disabled (default), 1= Enabled
18	Digital ramp no dwell high	0= Disabled (default), 1= Enabled
17	Digital ramp no dwell low	0= Disabled (default), 1= Enabled
16	Read FTW	0= SDIO read operation of the FTW register notes its contents (default), 1= SDIO read operation of the FTW register notes the FTW
15:14	I/O UPDATE rate control	Sets prescale ratio of the divider, driving an auto I/O UPDATE timer as shown: 00= divides by 1 (default), 01= divides by 2, 10= divides by 4, 11= divides by 8.
13:12	Open	
11	PDCLK (parallel data clock)	0= PDCLK off; sent to a Logic 0 state 1= PDCLK signal shows at PDCLK (default)
10	Invert PDCLK	Not applicable since parallel data was not used.
9	Invert TxEnable	0= Doesn't take place, 1= Takes place.
8	Open	
7	Enable Matched Latency	0= concurrent use of DDS parameters for output in the sequence given (default). 1= concurrent use of DDS parameter updates reach the output at once.
6	Hold last value of Data assembler	Not applicable since parallel data was not used
5	Disable Sync timing validation	0= SYNC_SMP_ERR pin shows synced pulse sampling errors, 1= SYNC_SMP_ERR pin disabled
4	Parallel data port modulation	0 = disabled (default), 1 = enabled
3:0	FM Gain	FM gain of the signal, not applicable since parallel data was not used

Table 3.9- Register Descriptions for Control Function Register 3 (Address- 0x02) [21]

Bits	Parameter	Register Description
31:30	Open	
29:28	DRV0	Controls REFCLK_OUT pin, default- 01b.
27	Open	
26:24	VCO SEL	Decides range of frequencies of PLL VCO, default- 111b.
23:22	Open	
21:19	Charge pump current (I_{CP})	Sets I_{CP} in PLL, default- 111b.
18:16	Open	
15	REFCLK input divider	0 = enabled (default), 1 = disabled

Bits	Parameter	Register Description
14	Reset REFCLK input divider	0 = input divider reset, 1 = input divider default operation
13:11	Open	
10	Reset PFD (phase frequency detector)	0= default operation, 1= phase detector disabled
9	Open	
8	Enable PLL	0 = PLL disabled (default), 1= PLL used
7:1	N	Stores PLL feedback divider modulus; default-0000000b.
0	Open	

3.2.5 Uses of AD9910

AD9910 in the digital ramp mode is useful for gravimetry as a part of atom interferometry. The chirp rate provides insights for measuring the gravity at a given place. The detailed explanation of the usage of DDS in atom interferometry was given in the previous chapter.

3.3 TRIGGERING THE DDS

Triggering the DDS is to send a reference electronic signal (like a square pulse), which acts as a switch to begin the operations of the DDS. The output retains it's value until there is a sufficient change in the input, hence the name 'trigger'. DDS can be triggered using a hardware trigger or software trigger.

3.3.1 Software Trigger

Here the DDS is triggered directly using programming on it's pins. No external voltage is required and instead trigger takes place via resetting or updating the DDS. For the AD9850, the software trigger was used to begin the writing of the byte and was done so by generating pulse from the FQ_UD pin (see Appendix A1]).

3.3.2 Hardware Trigger

Here the DDS is triggered using a microcontroller or other external hardware sources (such as function generators) that are pre-programmed resulting in an electronic pulse signal being applied to begin the operation of the DDS. This is due to the fact that external voltage from the hardware is being applied to the DDS to commence it's operation or update the DDS. Hardware trigger takes place on MOSFETs present in the DDS, since they act as a switch. For the AD9910, the hardware trigger was used

to begin the writing of the byte, it was applied on the DRCTL pin to begin the required function/ operation of the DDS (in this case frequency sweeping as an upward digital ramp). See the appendix A2], Arduino Codes, code number 4 for the code of hardware trigger on AD9910

Chapter 4 - RESULTS AND DISCUSSION

In the previous chapter, a brief idea about the DDS used and the experiments performed with the DDS in the form of their operational modes were explained along with a mention of the experimental data. In this chapter, a brief mention about the procedure and results for the experiments done will be given along with a description some of the waveforms generated using the DDS chips mentioned in chapter 3.

4.1 PROCEDURE

4.1.1 AD9850

For the AD9850, all the designated pins were connected to the microcontroller using jumper cables with the designated output pins (sine output, square wave output) connected to the oscilloscope using BNC cables and test clips. Then, the function of the individual pins was tested by connecting the microcontroller to the oscilloscope, a logic probe with the help of the code (see appendix A1) in order to get the designated function of chip select, reset of DDS and the frequency tuning word as the output on the oscilloscope. The same function was repeated by replacing the logic probe with the AD9850 DDS, using the same code, to get the required sinusoidal output/ square wave output. All steps are repeated by varying the frequency as defined in the code.

4.1.2 AD9910

For the AD9910, all the designated pins were connected to the microcontroller or enabled using jumper cables or connected directly to the DDS with the designated output pins, connected to the oscilloscope using BNC cables/ test clips. Then, the function of the individual pins was tested by connecting the microcontroller to the oscilloscope, a logic probe with the help of Arduino and Python codes (see Appendix A2) in order to get the designated function of chip select, reset of DDS and the frequency tuning word as the output on the oscilloscope. For single tone mode, the same was repeated by replacing the logic probe with the AD9910 DDS, using the single tone mode code (both Python and Arduino codes, see Appendix A2), ensuring that the mode bit was set to 0x00 to get the required sinusoidal output in single tone mode. For ramping mode, the codes for the ramping mode and it's trigger (see Appendix A2) were executed (ensuring that the mode bit was set to 0x01 and the trigger is given every five seconds with the help of a microcontroller which is indirectly connected to DRCTL pin), and the frequencies (initial and final), ramping step size and the bits for the ramp mode in the code were varied so that the sweep rate of DDS is around 25.1 MHz (with the help of AD9910 Evaluation Software to get the exact bits for the registers). For phase locking, a function generator was connected to the

oscilloscope and to the DDS via the external oscillator pin. The frequency of the DDS/ function generator, the bits for the control function registers in the code were varied accordingly (with the help of AD9910 Evaluation Software to get the exact bits for the registers), until the output on the oscilloscope shows the function generator's output sine wave which is stationary. To analyze the output of the AD9910 in both modes, the BNC cable may be connected to the spectrum analyzer instead of the oscilloscope, to get a stationary peak at that frequency for single tone mode and a moving peak in the range of frequencies in ramping mode. It must be noted that the AD9910 DDS has to be enclosed in a box.

Figure 4.1- The square wave output for AD9850 of frequency 1 MHz

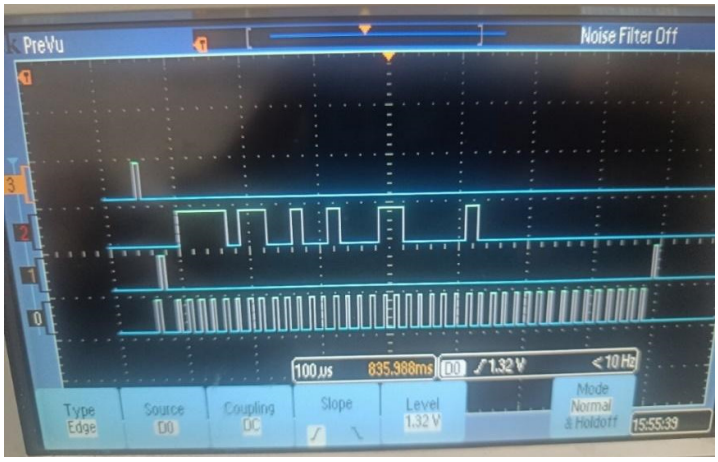


Figure 4.2- The representation of the frequency tuning word for the AD9850 for 1 MHz. Here 0 is for the W_CLK (reference clock), 1 is for CS (chip select) or FQ_UD, 2 is for DATA and 3 is for RESET.

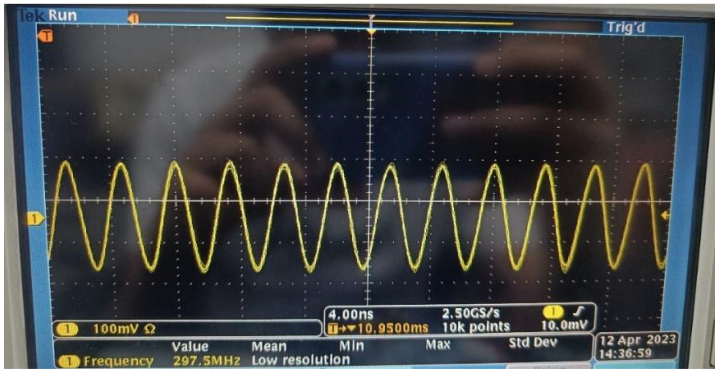


Figure 4.3- The 300 MHz output for single tone mode in AD9910

4.2 RESULTS FOR THE INDIVIDUAL DDS

For the AD9850 DDS, the maximum output frequency which could be generated is of the order of 10 MHz (for both sine and square waves) while for the AD9910, the maximum output frequency which could be generated is around 300 MHz in the form of sine waves, and as a Dirac Delta function on the spectrum analyzer. This is dependent not just on the sampling rate of not just the DDS but also of the oscilloscope on which the waves are being generated on. Both modes of operation of the AD9910 DDS can be successfully carried out, with ramping mode possible, as long as a trigger is given. However, for the AD9850, the square wave output did have some imperfections, considering their representation as a Fourier series and the inability of the DDS to process this infinite summation (especially considering the presence of the low pass filter). Overall, this indicates that the AD9850 has a simple architecture while the AD9910 has a sophisticated architecture. Figures 4.1 to 4.3 show the resulting outputs of all the DDS used.

4.3 EXPERIMENTAL RESULTS

Based on the experimental data as given in Table 3.1, and on the formulas from the previous chapters, the following are the results for the sweeping rate of the DDS and the estimated value of acceleration due to gravity from the DDS sweeping rate, assuming the frequency range for the sweeping is from 80 MHz to 82.5 MHz in steps of 0.84 Hz as mentioned in the last chapter.

1] Average Sweeping time of the DDS= 99.78 ± 0.2 ms

2] Chirp Rate of the DDS= 25.06 ± 0.05 MHz/s

3] Estimated value for the acceleration due to gravity (if gravimetry is performed)= 9.78 ± 0.02 m/s^2

These results are valid within suitable limits of error from the actual values of the chirp rate for ^{87}Rb and the acceleration due to gravity. Apart from the sweeping rate, the required frequency of a sine wave (as given from a function generator) to phase lock a sine wave of frequency 80 MHz generated by the DDS in single tone mode was found out to be 40 MHz, and this indicates that an output frequency of the DDS can be a multiple of frequency of the function generator to be phase locked with the function generator, depending on the code.

Chapter 5 - SUMMARY AND OUTLOOK

5.1 SUMMARY

The DDS that were used could support up to frequencies of 62.5 MHz and 400 MHz for AD9850 and AD9910 respectively, although I was able to generate frequencies up to 10 MHz (for AD9850) and 300 MHz (for AD9910). Apart from the frequency ranges, the circuitry and pins of both DDS used indicate that the AD9850 is probably the simplest of all direct digital synthesizers to generate pulses and waves for simple applications, while AD9910 finds massive applications in the field of Physics considering it's sophisticated architecture.

In AD9910, for single tone mode, the analog output of this DDS was successfully generated and shown as a Dirac Delta Function with peak at the frequency of the output on a spectrum analyzer (which gives the Fourier representation of the output). The ramping of the DDS in steps of 0.84 Hz was successfully carried out from 80 MHz to 82.5 MHz, for an average time period of 99.78 ms, to get the desired chirp rate of 25.06 (± 0.05) MHz/s. This was observed on a spectrum analyser as a Dirac Delta Function with the peak moving in the range of frequencies mentioned. The triggering of the ramp mode was successfully carried out on DRCTL pin (to control ramping), using an extra microcontroller to trigger the ramping every five seconds. For both single tone mode and ramp mode, the output was phase locked with an arbitrary function generator and the required frequency to make it happen was found out to be around 40 MHz.

5.2 FUTURE OUTLOOK

The results for the chirp rate obtained for frequency sweeping are compatible enough for atom interferometry with Rb having a wavelength of 780.2 nm, provided phase locking is present. However, to make the entire system user friendly and to have a good control over frequency, sweep rate in frequency and phase of the lattice beam, one must make the DDS operate like a function generator for driving the acousto-optic modulator. Hence, in doing so, the box to fit the DDS inside is cut in a configuration similar to that of a power supply and then appropriate changes to the code can be made to ensure that the DDS operates in that manner. Another change which can be incorporated is by using an identical pair of DDS, instead of a DDS and a function generator for the phase locking.

5.3 ADVANTAGES OF DDS

1] DDS provides excellent output frequency tuning resolution (in the μHz scale) and is capable of tuning sub-degree phase, all of which are fully controlled digitally. [22]

2] DDS offers incredibly quick sweeping rate in terms of establishing frequency/ phase outputs, phase continuous frequency sweeps free from overshoot or time delays for analog related loop settling and phase continuous frequency sweeps. [22]

3] With analogue synthesiser systems, component aging and temperature drift need manual system tuning and tweaking, which is not the case for DDS digital design [22].

4] The interface for digital control of DDS architecture offers a scenario in which systems may be distantly controlled and painstakingly tweaked under processor control. [22]

5] When utilised as a quadrature synthesiser, DDS gives unmatched similarity and output control for in phase and out of phase components, as quadratures. [22]

5.4 DISADVANTAGES OF DDS

1] Limit on maximum output frequency [23]

2] Can generate spurious frequency content resulting in higher phase noise. Usually such signals are unnecessary ones and outside the given range of frequencies and they are caused by spurious changes in the phase of the synthesizer's output. This results in energies at frequencies other than the desired one. In a sensitive receiver, phase noise will hide a weak signal that would otherwise be detected. [23]

3] Cannot be used to convert signals from DC to AC at very high voltages (>5 V).

References

[1] "*Atom Interferometry Introduction*," Muller Group, [Online]. Available: http://matterwave.physics.berkeley.edu/atom-interferometry.

[2] M. Travagnin, "*Cold atom interferometry sensors: Physics and technologies. A scientific background for EU policymaking*," Publications Office of the European Union, Luxembourg, 2020.

[3] R. Karcher, F. P. Dos Santos and S. Merlet, "*Impact of direct-digital-synthesizer finite resolution on atom gravimeters*," Physical Review, vol. 101, no. 043622, pp. 1-2, 28 April 2020.

[4] Shiv Sagar Maurya, Pranab Dutta, Korak Biswas, Jay Mangaonkar, Kushal Patel and Umakant Rapol, "*Progress of Bose- Einstein Condensate based gravimeter at IISER Pune*," p. 3.

[5] P. A. Altin, M. T. Johnsson, V. Negnevitsky, G. R. Dennis, R. P. Anderson, J. E. Debs, S. S. Szigeti, K. S. Hardman, S. Bennetts, G. D. McDonald, L. D. Turner, J. D. Close and N. P. Robins, "*Precision atomic gravimeter based on Bragg diffraction*," New Journal of Physics, vol. 15, no. 023009, pp. 4-7, 6 February 2013.

[6] M. Kozuma, L. Deng, E. W. Hagley, J. Wen, R. Lutwak, K. Helmerson, S. L. Roston and W. D. Phillips, "*Coherent splitting of Bose-Einstein condensed atoms with optically induced Bragg diffraction,*" Physical Review Letters, p. 3, 1999.

[7] Juan-Juan Tao, Min-Kang Zhou, Qiao-Zhen Zhang, Jia-Feng Cui, Xiao-Chun Duan, Cheng-Gang Shao and Zhong-Kun Hu, "*Directly measuring the direct digital synthesizer frequency chirp rate for an atom interferometer*," Rev Sci Instrum, vol. 86, no. 096108, p. 3, 2015.

[8] Eva Murphy and Colm Slattery, "*All About Direct Digital Synthesis*," August 2004. [Online]. Available: https://www.analog.com/media/en/analog-dialogue/volume-38/number-3/articles/all-about-direct-digital-synthesis.pdf.

[9] Shutterstock, [Online]. Available: https://www.shutterstock.com/image-vector/arduino-mega-top-view-illustration-1604877349.

[10] Analog Devices, "*DDS Tutorial,*" 12 February 1999. [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/450968421DDS_Tutorial_rev12-2-99.pdf.

[11] "*Digital to Analog Converters*," [Online]. Available: https://www.tutorialspoint.com/linear_integrated_circuits_applications/linear_integrated_circuits_applications_digital_to_analog_converters.htm.

[12] Art Pini, "*The Basics of Direct Digital Synthesizers (DDSs) and How to Select and Use Them,*" DigiKey, 20 March 2019. [Online]. Available: https://www.digikey.in/en/articles/the-basics-of-direct-digital-synthesizers-ddss#:~:text=The%20phase%20accumulator%20is%20a,size%20of%20the%20counter%20gain..

[13] [Online]. Available: https://www.electronics-tutorials.ws/filter/filter_2.html.

[14] Analog Devices, "*Fundamentals of Direct Digital Synthesis*," [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf.

[15] Robert Sheldon, "*Most Significant and Least Significant Bit*," Tech Target, July 2022. [Online]. Available: https://www.techtarget.com/whatis/definition/most-significant-bit-or-byte#:~:text=The%20most%20significant%20bit%20(MSB,0111%2C%20the%20MSB%20is%200..

[16] Piyu Dhaker, "Introduction to SPI Interface," Analog Devices, September 2018. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html.

[17] Brendon Cronin, "*DDS Generates High Quality Waveforms*," Analog Devices, January 2012. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/dds-generates-high-quality-waveforms-efficiently.html.

[18] Rahul Awati, "*Phase Locked Loop*," TechTarget, June 2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/phase-locked-loop.

[19] Analog Devices, "*AD9850*," [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9850.pdf.

[20] Components101, "*AD9850 Signal Generator*," 3 July 2020. [Online]. Available: https://components101.com/modules/ad9850-dds-signal-generator-module.

[21] Analog Devices, "*AD9910 Data Sheet*," [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9910.pdf.

[22] "*What are the advantages of using a DDS?*," [Online]. Available: https://www.analog.com/en/education/education-library/faqs/faq_the_advantages_of_using_a_dds.html#:~:text=They%20offer%20extremely%20fast%20%E2%80%9Chopping,related%20loop%20settling%20time%20anomalies..

[23] "*Direct Digital Synthesis*," Engineering Projects, 17 April 2022. [Online]. Available: https://bestengineeringprojects.com/direct-digital-synthesis/.

Appendix

A1] Code to Generate Output for AD9850

“

```
const int W_CLK=13; // pin 2 - Signal, the timer bus (W_CLK)
const int FQ_UD=12; // Pin 3 - change the frequency (FQ)
const int DATA=11; // Pin 4 - data Line of the serial interface (DATE)
const int RESET=10; // Pin 5 - Reset module (RESET)
bool
ch[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0};
#define pulseHigh(pin) {digitalWrite(pin, HIGH); delay(1); digitalWrite(pin,
LOW);} // a function of the heartbeat is set at the time the high status
void setup() {
 // pin configuration Arduino
 pinMode(FQ_UD, OUTPUT);
 pinMode(W_CLK, OUTPUT);
 pinMode(DATA, OUTPUT);
 pinMode(RESET, OUTPUT);
 pulseHigh(RESET);
 pulseHigh(W_CLK);
 pulseHigh(FQ_UD); // toggles serial interface
 Serial.begin(9600);
}
// frequency is calculated according to the formula where frequency entered is
in SI units and then tuning word computed and processed
void sendFrequency(double frequency) {
 int64_t freq = frequency * 4294967296/125000000; // frequency 125 MHz AD9850
 int length=40,i;
  bool op[length];
  i=0;
  while(freq>=0,i<=length)
  {
    op[i]=freq%2;
    i++;
    freq=freq/2;
  }
  for (int j = 0; j< length; j++)
  {
        Serial.print(op[j]);
  }
  digitalWrite(W_CLK,LOW);
  digitalWrite(DATA,LOW);
  digitalWrite(FQ_UD,LOW);
  delay(1);
  for(int k=0;k<length;k++)
  {
    if(ch[k]==0)
    {
      digitalWrite(FQ_UD,LOW);
      if(ch[k-1]==0 || ch[k-1]==1)
      {
```

```
        if(op[k]==0)
        {
          pulseHigh(W_CLK);
          digitalWrite(DATA,LOW);
          delay(1);
        }
        else if(op[k]==1)
        {
          pulseHigh(W_CLK);
          digitalWrite(DATA,HIGH);
          delay(1);
        }
      }
      else
      {
         digitalWrite(W_CLK,LOW);
        digitalWrite(DATA,LOW);
        delay(1);
        digitalWrite(W_CLK,LOW);
        digitalWrite(DATA,LOW);
        delay(1);
      }
    }
    else if(ch[k]==1)
    {
      digitalWrite(FQ_UD,HIGH);
      digitalWrite(W_CLK,LOW);
      digitalWrite(DATA,LOW);
      delay(1);
      digitalWrite(W_CLK,LOW);
      digitalWrite(DATA,LOW);
      delay(1);
    }
  }
  digitalWrite(W_CLK,LOW);
  digitalWrite(DATA,LOW);
  pulseHigh(FQ_UD);
  delay(1);
}
void loop()
{
  sendFrequency(1000000);
  while(1);
}
”
```

A2] Codes to Generate Output for AD9910

Arduino Codes

1] To read incoming bytes and define the pins used for the DDS pins

“

```
#include <SPI.h>
```

```
#define chipSelect  10
#define P0  4
#define P1  3
#define P2  2
#define MR  A3
#define EPD  5
#define IOU  6
#define IOR  9
#define DRCTL 7
#define DRHOLD 8

const int ByteCount = 21;
int Bytes [ByteCount];
int WritePermit = 0;

void setup() {
   pinMode(MR, OUTPUT);
   pinMode(EPD, OUTPUT);
   pinMode(IOR, OUTPUT);
   pinMode(P0, OUTPUT);
   pinMode(P1, OUTPUT);
   pinMode(P2, OUTPUT);
   pinMode(chipSelect, OUTPUT);
   pinMode(IOU, OUTPUT);
   pinMode(DRCTL,OUTPUT);
   pinMode(DRHOLD,OUTPUT);
   digitalWrite(DRCTL, LOW);
   SPI.begin();
   Serial.begin(115200);
   SPI.setDataMode( SPI_MODE0);
   SPI.setBitOrder(MSBFIRST);
}

//To read any incoming bytes
void  loop(){
 ReadIncomingBytes(ByteCount, Bytes);
 if (Bytes[0] == 0x00)
 {
   if (WritePermit == 1)
   {
     ExecuteSingleToneMode();
     WritePermit = 0;
   }
 }
 if (Bytes[0] == 0x01)
 {
   if (WritePermit == 1)
   {
     ExecuteDigitalRampMode();
     WritePermit = 0;
   }
 }
 while(1);
}
”
```

2] To demonstrate the functions and modes of operation of DDS by reading and writing the registers, with phase locking present (Note- for SPI, the registers address are written first followed by the four hexadecimal characters as shown in the register description tables above).

“

```
//To give a pulse to pin
void GivePulseToPin(int PinNum, int OffTime, int OnTime)
{
  delay(OffTime);
  digitalWrite(PinNum, HIGH);
  delay(OnTime);
  digitalWrite(PinNum, LOW);
}

//To reset DDS
void ResetDDS()
{
   digitalWrite(MR, HIGH);
   delay(10);
   digitalWrite(MR, LOW);
   digitalWrite(EPD, LOW);
   digitalWrite(IOR, LOW);
   digitalWrite(P0, LOW);
   digitalWrite(P1, LOW);
   digitalWrite(P2, LOW);
   digitalWrite(chipSelect, HIGH);
   digitalWrite(IOU, LOW);
   digitalWrite(DRCTL, LOW); //
   digitalWrite(DRHOLD, LOW);
}

//To Update IO on IOUPDATE pin
void UpdateIO()
{
  GivePulseToPin(IOU, 100, 100);
}

//To Update DRCTL pin
void UpdateDRCTL()
{
  GivePulseToPin(DRCTL, 5, 8);
}

//To Read and Write the Registers
void    ReadWriteRegister(int    RegAddress,    int    ByteAddressStart,    int
ByteAddressStop)
{
    digitalWrite(chipSelect, LOW);
    SPI.transfer(RegAddress);
    for(int i=ByteAddressStart; i<=ByteAddressStop; i++)
    {
      SPI.transfer(Bytes[i]);
    }
    digitalWrite(chipSelect, HIGH);
}
```

```
// To execute single tone mode
void ExecuteSingleToneMode()
{
    ResetDDS();
    digitalWrite(chipSelect, LOW);

    // Writing CFR2
    SPI.transfer(0x01);
    SPI.transfer(0x01);
    SPI.transfer(0x40);
    SPI.transfer(0x08);
    SPI.transfer(0x20);
    digitalWrite(chipSelect, HIGH);
    UpdateIO();
    digitalWrite(chipSelect, LOW);

    // Writing CFR3
    SPI.transfer(0x02);
    SPI.transfer(0x1D);
    SPI.transfer(0x3F);
    SPI.transfer(0x41);
    SPI.transfer(0x3c);

    digitalWrite(chipSelect, HIGH);
    UpdateIO();

    // Writing the profile registers
    Bytes[1]=0x08;
    Bytes[2]=0xb5;
    Bytes[3]=0x00;
    Bytes[4]=0x00;
    Bytes[5]=0x11;
    Bytes[6]=0x11;
    Bytes[7]=0x11;
    Bytes[8]=0x11;
    ReadWriteRegister(0x0e, 1, 8)
    UpdateIO();
}

// To execute digital ramp mode
void ExecuteDigitalRampMode()
{
    ResetDDS();
    digitalWrite(chipSelect, LOW);
    // Writing CFR1
    SPI.transfer(0x00);
    SPI.transfer(0x00);
    SPI.transfer(0x00);
    SPI.transfer(0xE0);
    SPI.transfer(0x00);

    digitalWrite(chipSelect, HIGH);
    digitalWrite(chipSelect, LOW);
    // Writing CFR2
    SPI.transfer(0x01);
```

```
    SPI.transfer(0x00);
    SPI.transfer(0x48);
    SPI.transfer(0x08);
    SPI.transfer(0x20);
    digitalWrite(chipSelect, HIGH);
    digitalWrite(chipSelect, LOW);

    // Writing CFR3
    SPI.transfer(0x02);
    SPI.transfer(0x0d);
    SPI.transfer(0x3F);
    SPI.transfer(0x41);
    SPI.transfer(0x3c);

    digitalWrite(chipSelect, HIGH);
    digitalWrite(chipSelect, LOW);

    // Writing Auxillary DAC
    SPI.transfer(0x03);
    SPI.transfer(0x00);
    SPI.transfer(0x00);
    SPI.transfer(0x7f);
    SPI.transfer(0x7f);

    // Writing the registers
    digitalWrite(chipSelect, HIGH);
    Bytes[1]=0x11;
    Bytes[2]=0x99;
    Bytes[3]=0x99;
    Bytes[4]=0x9A;
    Bytes[5]=0x11;
    Bytes[6]=0x11;
    Bytes[7]=0x11;
    Bytes[8]=0x11;
    Bytes[9]=0x00;
    Bytes[10]=0x00;
    Bytes[11]=0x00;
    Bytes[12]=0x03;
    Bytes[13]=0x00;
    Bytes[14]=0x00;
    Bytes[15]=0x00;
    Bytes[16]=0x03;
    Bytes[17]=0x00;
    Bytes[18]=0x0A;
    Bytes[19]=0x00;
    Bytes[20]=0x0A;
    ReadWriteRegister(0x0b, 1, 8);
    ReadWriteRegister(0x0c, 9, 16);
    ReadWriteRegister(0x0d, 17, 20);
    UpdateIO();
    //UpdateDRCTL();
 }
"
```

3] To read any incoming bytes for communication

“

```
void ReadIncomingBytes(int TotBytes, int* List)
{
 for (int i = 0; i < TotBytes; i++)
 {
    while (!Serial.available());
        List[i] = Serial.read();
 }
 WritePermit = 1;
}
```

”

4] To perform hardware trigger on the AD9910

“

```
int CLK= 7;
void setup() {
  pinMode(CLK,OUTPUT);
  Serial.begin(9600);
}

void loop() {
  //code to trigger ramping every five seconds, when ramping must begin
  digitalWrite(CLK,HIGH);
  delay(100);
  digitalWrite(CLK,LOW);
  delay(4900);
}
```

”

Python Code

Aim- To demonstrate both modes of operation of AD9910

“

```
import serial
import time
import struct
import numpy as np
import timeit
# Frequency mapping

F_OUT    = 80e6  # input frequency
F_SYSCLK = 1200e6

FTW = round((4294967295.0 *((F_OUT / F_SYSCLK))));

FTWHex = hex(FTW)[2:]
for elems in range(8-len(FTWHex)): FTWHex = '0'+FTWHex

FTWByteStr = []
FTWByte    = []
for elems in range(0,len(FTWHex),2):
    FTWByteStr.append('0x'+FTWHex[elems]+FTWHex[elems+1])
for elems in FTWByteStr  : FTWByte.append(int(elems,16))
```

```
# Amplitude mapping
AmpOUT    = 1    # input amplitude (Range 0-1)
if AmpOUT <= 1: ScaledAmp = int(AmpOUT * 16383)
else: ScaledAmp = int(16383)
AmpHex = hex(ScaledAmp)[2:]
for elems in range(4-len(AmpHex)): AmpHex = '0'+AmpHex

AmpByteStr = []
AmpByte    = []

for elems in range(0,len(AmpHex),2):
    AmpByteStr.append('0x'+AmpHex[elems]+AmpHex[elems+1])
for elems in AmpByteStr  : AmpByte.append(int(elems,16))

# Digital ramp mode (Frequency)
# Starting and Ending Frequency
InitFreq  = 80e6
FinalFreq = 82.5e6

# Frequency increment and decrement
FreqIncr  = 0.84
FreqDecr  = 0.84

InitFreq  = round((4294967295.0 *((InitFreq / F_SYSCLK))));
FinalFreq = round((4294967295.0 *((FinalFreq / F_SYSCLK))));
FreqIncr  = round((4294967295.0 *((FreqIncr / F_SYSCLK))));
FreqDecr  = round((4294967295.0 *((FreqDecr / F_SYSCLK))));

InitFreqHex = hex(InitFreq)[2:]
for elems in range(8-len(InitFreqHex)): InitFreqHex = '0'+ InitFreqHex

InitFreqByteStr = []
InitFreqByte    = []

for elems in range(0,len(InitFreqHex),2):
    InitFreqByteStr.append('0x'+InitFreqHex[elems]+InitFreqHex[elems+1])
for elems in InitFreqByteStr  : InitFreqByte.append(int(elems,16))
FinalFreqHex = hex(FinalFreq)[2:]
for elems in range(8-len(FinalFreqHex)): FinalFreqHex = '0'+ FinalFreqHex

FinalFreqByteStr = []
FinalFreqByte    = []

for elems in range(0,len(FinalFreqHex),2):
    FinalFreqByteStr.append('0x'+FinalFreqHex[elems]+FinalFreqHex[elems+1])
for elems in FinalFreqByteStr  : FinalFreqByte.append(int(elems,16))

FreqIncrHex = hex(FreqIncr)[2:]
for elems in range(8-len(FreqIncrHex)): FreqIncrHex = '0'+ FreqIncrHex
FreqIncrByteStr = []
FreqIncrByte    = []

for elems in range(0,len(FreqIncrHex),2):
    FreqIncrByteStr.append('0x'+FreqIncrHex[elems]+FreqIncrHex[elems+1])
for elems in FreqIncrByteStr  : FreqIncrByte.append(int(elems,16))
FreqDecrHex = hex(FreqDecr)[2:]
for elems in range(8-len(FreqDecrHex)): FreqDecrHex = '0'+ FreqDecrHex
FreqDecrByteStr = []
FreqDecrByte    = []
for elems in range(0,len(FreqDecrHex),2):
```

```
    FreqDecrByteStr.append('0x'+FreqDecrHex[elems]+FreqDecrHex[elems+1])
for elems in FreqDecrByteStr  : FreqDecrByte.append(int(elems,16))

# Writing Data to Arduino
Arduino = serial.Serial('COM5', 115200)
time.sleep(1)
Arduino.write(struct.pack("!B",Mode))

# Condition for Single Tone Mode
if Mode == 0x00 :
     for elems in AmpByte: Arduino.write(struct.pack("!B",elems))
     Arduino.write(struct.pack("!B",0x00))
     Arduino.write(struct.pack("!B",0x00))
     for elems in FTWByte: Arduino.write(struct.pack("!B",elems))
     for elems in range(12): Arduino.write(struct.pack("!B",0x00))

# Condition for Digital Ramp Mode
if Mode == 0x01 :
     for elems in FinalFreqByte: Arduino.write(struct.pack("!B",elems))
     for elems in InitFreqByte: Arduino.write(struct.pack("!B",elems))
     for elems in FreqIncrByte: Arduino.write(struct.pack("!B",elems))
     for elems in FreqDecrByte: Arduino.write(struct.pack("!B",elems))
     Arduino.write(struct.pack("!B",0xff))
     Arduino.write(struct.pack("!B",0xff))
     Arduino.write(struct.pack("!B",0xff))
     Arduino.write(struct.pack("!B",0xff))
”
```