DESIGN, SIMULATION AND CONSTRUCTION

OF A

DIGITAL FREQUENCY COUNTER

BY

ABDULLATEEF OMOLOYE OYELADE,

SPS/12/MPY/00014

B.SC. (HONS.) PHYSICS (ABU) , PGDIP(BUK)

BEING A THESIS SUBMITTED TO THE DEPARTMENT OF PHYSICS, FACULTY OF SCIENCE, BAYERO UNIVERSITY, KANO IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTERS (M.SC) IN ELECTRONICS.

APRIL,2016

DECLARATION

I hereby declare that work is the product of my research efforts undertaken under the supervision of Prof. G.S.M Galadanchi and has not been presented anywhere for the award of a degree or certificate. All sources have been duly acknowledged.

---------- ----------

AbdullateefOmoloyeOyelade (SPS/12/MPY/00014) Date

CERTIFICATION

This is to certify that the research work for this thesis and the subsequent preparation of this thesis by AbdullateefOmoloyeOyelade (SPS/12/MPY/00014) were carried out under my supervision.

---------- ----------

Prof. G S M Galadanchi
(Supervisor) Date

---------- ----------

Prof. G S M Galadanchi
(Head of Department) Date

APPROVAL

This work has been examined and approved for the award of Masters in Electronics.

---------- ----------
(External Examiner) Date

---------- ----------
Dr. Muktar H. Ali Date
(Internal Examiner)

---------- ----------
Prof. G S M Galadanchi Date
(Supervisor)

---------- ----------
Prof. G S M Galadanchi Date
(Head of Department)

---------- ----------
(SPS Representative) Date

ACKNOWLEDGEMENT

Firstly, all praise be to the Almighty Allah for sparing me with life, health and countless blessings to be able to participate in this M.SC course.

Great appreciation goes to my supervisor, Prof.Garba S.M Galadanchi for his guidance and assistance over this research work.

Next, I am highly grateful to Dr. Muktari H. Ali, the Post-Graduate Programs Coordinator of Physics Department, B.U.K (2013), who processed my admission into this course, thereby affording me this huge opportunity. May the Almighty Allah continue to increase his blessings for him. Amin.

Special regards and best wishes also go to all the other Lectures of Physics Department, B.U.K, who put us through during this M.SC course. May Allah bless you all too sir.
Mallam Mustapha Sadiq of Micro Scale Ltd is also appreciated fir assisting me with the source code for my Microcontroller unit.

Great thanks to all the members of the 2013/2014 M.SC Electronics class (SPS/12/MPY) for their group cooperation, hard work and understanding during our class tutorials. And thanks to the class for giving me the opportunity to have been their class captain throughout.

On my home front, special appreciation goes to my family and friends, notably my wife Mrs. RashidatAbdullateef, FatiahAbdullateef, Alhaji M. Sadiq, Mr. A.T Momolosho, Mr. S. Soladoye, and Iya Halima for their support and goodwill.

Numerous thanks also to all others who have in one way or the other been of crucial assistance to me towards the success of this course. Wishing God’s blessings for you all.

DEDICATION

This work is dedicated to the Glory of Almighty God who provided me with the Life, Resources and Journey Protection for me to be able to participate in this Programme. ALHAMDULILLAHI.

TABLE OF CONTENTS

Cover page ----------i

Declaration ----------ii

Certification ----------iii

Approval ----------iv

Acknowledgement ----------v

Dedication ----------vi

Table of contents----------vii

Abstract ---------- ix

CHAPTER ONE: INTROCDUCTION

1.1 Background Study ----------1

1.2 Problem Statement ----------1

1.3 Aim ----------2

1.4 Objectives ----------2

1.5 Scope ----------2

CHAPTER TWO: LITERATURE REVIEW

2.1 Frequency Counters ----------3

2.1.2 Basic Frequency Counter Block Diagram---------- 4

2.1.3 Frequency Counters/ Timers ---------- 6

2.1.4 Counter Timer Basic Block Diagram ----------20

2.2.17 Microcontroller Applications----------21

2.2.18 Diferences between Microcontrollers and Microprocessors----------.-22

CHAPTER THREE: METHODOLOGY, CONSTRUCYION AND TESTING

3.0 Introduction ---------- 24

3.1 The Circuit Design ----------24

3.2 The Design Process ----------25

3.3 The Simulation ----------27

3.4 Operating Principles of the Circuit ----------29

3.5 The Construction ----------31

3.6 Testing the Constructed DFC ----------32

CHAPTER FOUR: RESULTS AND DISCUSSION

4.1 Results ----------34

4.2 Discussion----------36

4.3Contribution to Knowledge ----------36

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary ----------37

5.2 Conclusion----------37

5.3 Recommendation ----------37

References ----------39

Appendix: Source code for the AT89C51 ----------41

ABSTRACT

Digital systems work with digital signals whose frequencies must be determined precisely, as with radio communication signals. But portable, handheld digital frequency counters are rare. In this work a microcontroller-based digital frequency counter capable of counting a range of 0-500 kHz was aimed, designed, and simulated.. The device design was based on an 8051 microcontroller, namely AT89C51, programmed using Keil µVision3, a C-program editor. The frequency counter was simulated using Proteus Circuit Simulator software. Tests during simulation with sample digital signals from the simulator showed that the counter could count up to 500 kHz while after construction the hardwaredevice counted up to a maximum of 400 kHz with live signals.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND STUDY

The history of frequency counters is strongly associated with Radio technology, which started measuring frequencies with *the Grid-Dip Meter*which was only accurate at relatively low frequencies (below 30MHz) and with wide modulation formats (like AM voice).(Margolin and Rider, 2005). Digitalfrequency counters were first built with*Vacuum Tubes***,**such as the Philips E1T Decade Counter Tube. Unfortunately all the vacuum-tube counters were not fast enough to allow direct counting at practical frequencies used in radio communications. (Turner, Capron, Laurence, & Conner, 2001). A major breakthrough was represented by the *TTL Chips*.TTL(transistor-transistor logic) chips which became widely available around 1970 allowed counting frequencies up to about*40MHz.* Accurate digital frequency counters were finally within reach of all radio users including amateurs (Fu, B. H., Liu, K. X., & Ma, J. C., 2014). *Decade Counters***,** such as the95H90, allowed reaching*300MHz*while the 11C90 counted up to*600MHz.* Around 1980*Prescalers*became available for frequency synthesis of TV sets-one of the first such Prescalers, the Siemens S0436, reliably counting upto*1.4GHz***.**(Langlois, J. M. P., & Al-Khalili, D., 2002).

1.2 THE PROBLEM STATEMENT

A frequency counter is an instrument of great importance in electronics since a signal must run in every electronic device or circuit in order to enable it function as intended. Frequency and period are among the characteristics of a signal. Whereas a multimeter can measure such properties as the voltage, current etc, only a frequency counter can measure the

frequency. This instrument is especially useful where the frequency of a signal is to be determined with high precision, such as in a broadcast station where the broadcast frequency must be exactly as allocated by Government. Most frequency counters appear in relatively large sizes, and hand-held versions are scarce to find. Hence the need for the construction of a portable version of frequency counters in this research.

1.3 AIM

To design, simulate and construct a portable, hand-held digital frequency counter using a microcontroller.

1.4 OBJECTIVES

- Design and simulation of a digital frequency counter.
- Construction and testing of the constructed DFC.
- Develop a wireless transmitter- receiver system to add to the system.
- To use the frequency counter produced to measure audio and communication signals.

1.5 SCOPE

The Bandwidth of the intended counter is the range of 0-500 kHz.

CHAPTER TWO

LITERATURE REVIEW

2.1 FREQUENCY COUNTERS

A frequency counter is an electronic instrument used for measuring frequency - measuring the number of pulses or oscillations in a repetitive electronic signal.(Choudhary, 2014).

If the event to be counted is already in electronic form, a simple interfacing is all that is needed while events that are not inherently electronic in nature need to be converted using some form of transducer. A mechanical event could be arranged to interrupt a light beam, and the counter made to count the resulting pulses. Complex signals may need some conditioning to make them suitable for counting.(Shen et al., 2009).

General purpose frequency counters include some form of amplifier, filtering and shaping circuitry at the input. Frequency counter-timers are able to measure time intervals as well as frequency. However, where higher frequencies are to be measured, the timer element is not included and the test equipment is just a frequency counter. Digital frequency counters are used for radio frequency (RF) measurements where it is important to test or measure the precise frequency of a particular signal.(Shen et al., 2009).A frequency counter works by using a counter which accumulates the number of events occurring within a preset period of time, known as the gate time (say, 1 second), transfers the value in the counter to a display and resets the counter to zero.

2.1.2 Basic Frequency Counter Block Diagram

Fig 2.1. is the basic block diagram for a frequency counter, consisting of several main blocks. Any digital frequency counter will generally assume this basic format.(Choudhary, 2014).

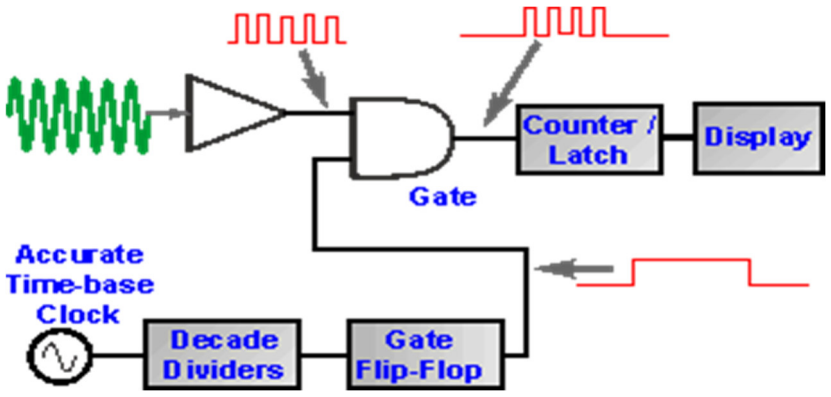

Fig. 2.1: Basic Frequency Counter Block Diagram. (Choudhary, 2014).

Input: If the input signal is in analogue (sinusoidal) form it has first to pass through an analogue to digital converter (ADC) where the signal is converted into a logic rectangular wave. Then the signal enters an input amplifier where it is amplified for processing within the digital circuitry` in the rest of the counter. Normally this stage contains a Schmidt trigger circuit so that noise does not cause spurious edges that would give rise to additional pulses that would be counted.

Accurate Clock Signal: The clock oscillator is used to provide an accurately timed gate signal that will allow through pulses from the incoming signal. It also gives the timing signals within the frequency counter. The time-base/clock is typically a quartz crystal oscillator and in high quality test equipment it would be an oven-controlled crystal

oscillator. In many instruments there will the capability to use a better quality external oscillator, or to use the frequency counter oscillator for other instruments.

Decade Dividers and Flip-Flop:Where thefrequency of the clock signal generated from the oscillator is too high, it is divided by decade dividers and fed into a flip flop and then to the main gate to give the enabling pulse for input signals to pass through the gate.

Gate: The gate has two inputs - one input receives a train of pulses from the incoming signal while the precisely timed gate enabling signal from the clock is applied to other input. The resultant output from the gate is a series of pulses for a precise amount of time. For example, if the incoming signal was at 1 MHz and the gate was opened for 1 second, then 1 million pulses would be allowed through.

Counter/Timer/Latch: The counter takes the incoming pulses from the gate. It has a set of divide-by-10 stages. Each stage divides by ten and therefore as they are chained the first stage is the input divided by ten, the next is the input divided by 10 x 10, and so forth. These counter outputs are then used to drive the display. The timer marks the time gate is open to inward signals.

In order to hold the output in place while the figures are being displayed, the output is latched. Typically the latch will hold the last result while the counter is counting a new reading. In this way, the display will remain static untilthe latch is updated with a new result from the counter and the new reading presented to the display. (Floyd, 2010).

Display: The display takes the output from the latch and displays it in a normal readable format. LCD, or LED displays are the most common. There is a digit for each decade that

the counter can display. Obviously other relevant information may be displayed on the display as well.(Floyd, 2010).

2.1.3 Frequency Counter-Timers

Where it is necessary to measure the time interval (i.e., period) of electronic signals, a counter-timer can be used. Using similar technology to a frequency counter, timer counters are often incorporated within the same test equipment, referred to as Frequency Counter-Timers, or as Interval Timers.(Choudhary, 2014).

Whether they are called frequency counter-timers, timer-counters or interval timers, their function is the same - they utilize similar basic technology and their measurement capabilities include:

Measurement of Time Interval of Waveform:

This is simply the reciprocal of the frequency. It typically measures the time from one positive-going edge of the waveform to another positive-going edge as shown in Fig. 2.2.

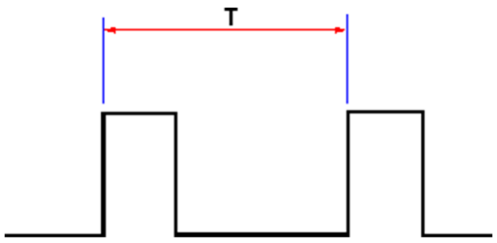

Fig.2.2: Measurement of Waveform Time Interval or Period: +ve-going to +ve –going edges.(Choudhary, 2014).

Measurement of Time Interval of Pulse:

Rather than measuring from one positive-going edge to the next, it is also possible to measure from a positive-going edge to the next negative-going one or alternatively, it could measure a negative-going edge to a positive one. In this way it is possible to use the frequency counter-timer to measure the time interval of a pulse (Fig. 2.3).

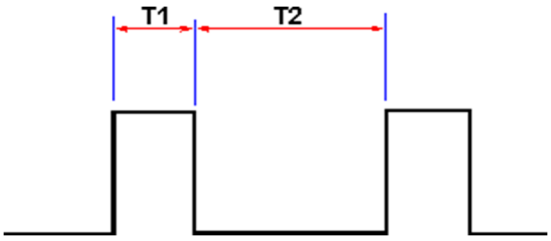

Fig. 2.3:Measurement of Waveform Time Interval or Period: +ve-going to –ve going edge (Choudhary, 2014).

It can be seen that the two options are represented by the two times T_1 and T_2 : T_1 being for the time interval + to – and T_2 for – to + going edges of the waveform.

In this way the functionality of the instrument has been greatly enhanced by the addition of the selection of the waveform edge transition.

Time Interval A to B:

Some counter-timers have dual inputs which can be used with two waveforms A and B and it is normal for there to be positive (rising) or negative (falling) edge selections on both channels. To set this up, it is a matter of connecting the required waveforms to the inputs and then selecting the edges,as in Fig. 2.4.

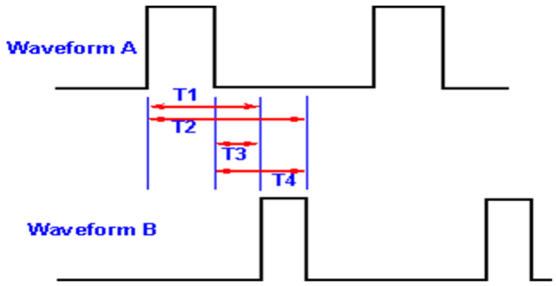

Fig. 2.4: Time Interval Measurements for Two Waveforms-Time Interval A-B. .(Choudhary, 2014)

2.1.4 Counter-Timer Basic Block Diagram

Like the frequency counter, the timer-counter or interval timer has a number of blocks that make up the test instrument. They are very similar to those used in the frequency counter, and just require reconfiguration to give the interval timing function. Fig. 2.5 shows the basic block diagram for a counter-timer operation.(Choudhary, 2014).

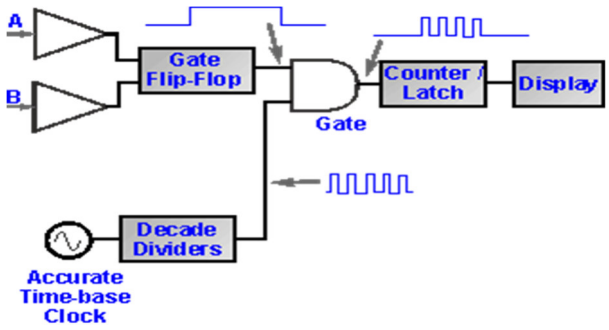

Fig. 2.5: Time Interval Counter Block Diagram. (Choudhary, 2014).

Being a reconfiguration of the circuitry for a traditional frequency counter, the functionalities of the input, gate flip-flop, clock oscillator, gate, counter/latch and display in this timer-counter are the same as explained earlier for the frequency counter block diagram.

2.1.5 I/O Interfaces

Input/output interfaces allow the user to send information to the frequency counter and receive information from it. Commonly used interfaces include USB, GP1B, RS232, Probe and Ethernet. Besides sending measurement results, a counter can notify the user when user-defined limits are exceeded..[Poole, 2015].

2.1.6 Applications of a Digital Frequency Counter

1. Frequency counters are widely used for the accurate measurement of frequency of radio frequency (RF) signals, or for that matter the frequency of any repetitive electronic signal.[Poole, 2015].
2. In Research and Development as well as in Manufacturing Centers, it is used to measure frequency accurately.
3. For many Crystal Oscillators, after being set to a certain frequency by means of a circuit and a capacitor, a digital frequency counter is used to confirm the actual frequency.
4. In broadcasting where it is important to transmit at a specific frequency, a digital frequency counter is needed to ensure that transmission is done at the specified frequency.
5. It can be used in an aircraft to know or receive a signal that is known to be transmitted at a specific frequency. Synthesizers are often used and when such a unit is manufactured the frequency has to be set correctly. Many other applications abound for the frequency counter.

2.2 THE MICROCONTROLLER

The principal element used in the building of the frequency counter in this work is a microcontroller. Also known as embedded controller, it is a functional computer system on a single chip containing a processor core (central processing unit (CPU)) and additional elements such as read-write temporary memory for data storage, memory for permanent data storage, peripherals, and input/output interfaces. A microcontroller usually has no key board, screen, printers or other recognizable I/0 devices as with a personal computer and may lack human interaction devices of any kind while typical input and output devices

include switches, relays, solenoids, LEDs, small or custom LCD displays and sensors for data such as temperature, humidity, light level etc. It emphasizes high integration and this drastically reduces the number of chips and the amount of wiring and circuit board space that would be needed to produce equivalent systems using separate chips.(Seward, 2008).The following are some of the features of a microcontroller.

2.2.1 Microcontroller Programs

Unlike other integrated circuits which only need to be connected to other components and turn the power supply on, the microcontrollers need to be programmed first.

Program writing is a special field of work with microcontrollers and consists of simple instructions written in the order in which they should be executed**.** In order to write a program, several "low-level" programming languages can be used such as Assembly, C and Basic (and their versions as well). There are also many applications running in the Windows environment, such as Genie Studio Wizard, KeilμVision 3, used to facilitate the programming and providing additional visual tools.

When the writing of a program for a microcontroller is completed, say in C- language, the program editor (the C software)provides a Compiler which compiles the program and, if successful, a hex file (a compact machine code in terms of 0's and 1's)is automatically generated for onward transfer to the microcontroller's memory.

Microcontroller programs must fit into the available on-chip program memory, since it would be costly to provide a system with external, expandable memory. (Jing and Xin-Guang, 2010).

2.2.2 Central Processing Unit (CPU)

This is the unit which controls and monitors all processes within the microcontroller and the user cannot affect its work. Its memory locations are called registers (Seward, 2008).

The role of registers is to help with performing various operations or any operations with data wherever data can be found. Memory and CPU have to be interconnected to allow for the exchange of data and functionality. For example, to add the contents of two memory locations and return the result again to memory, a connection is needed between memory and CPU. Simply put, there must be a way through which data goes from one block to another. That “way” is called a “bus”. Physically, it represents a group of 8, 16 or more bits.

There are two types of buses- address bus and data bus. The first consists of as many lines as the amount of memory to be addressed while the other one is as wide as data, in this case 8 bitsor the connection line. The first one serves to transmit address from CPU memory, and the second to connect to all blocks inside the microcontroller.

The CPU consists of several smaller subunits, of which the most important are:

- ***Instruction decoder*** - a part of the electronics which recognizes program instructions and runs other circuits on the basis of that. The abilities of this circuit are expressed in the "instruction set" which is different for each microcontroller family.
- ***Arithmetical Logical Unit (ALU)*** performs all mathematical and logical operations upon data.

Accumulator - an SFR closely related to the operation of ALU. It is a kind of working desk used for storing all data upon which some operations should be executed (addition,

shift etc.). It also stores the results ready for use in further processing. One of the SFRs, called the Status Register, is closely related to the accumulator, showing at any given time the "status" of a number stored in the accumulator (the number is greater or less than zero etc.). Fig. 2.6 shows the inter-relation between registers, memory and CPU.

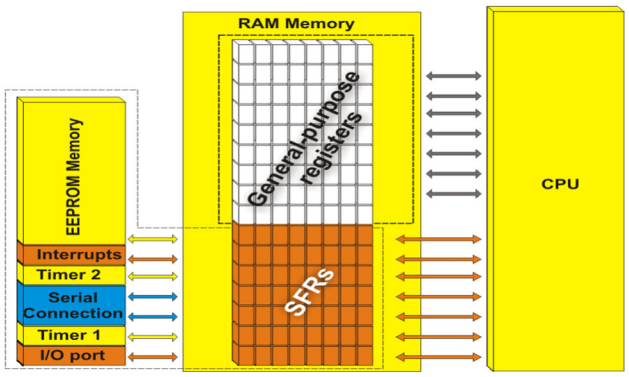

Fig. 2.6: Memory, Registers and CPU.(Seward, 2008).

2.2.3 ADC/DAC

Since microcontrollers are usually used to control devices, sometimes they need to accept input from the device they are controlling. And as they are built to interpret and process digital data (1's and 0's), they won't be able to do anything with the analog signals that may be sent to them by a device. So the analog to digital converter (ADC) is used to convert the incoming data into a form that the processor can recognize. There is also a digital to analog converter (DAC) that allows the processor send data back to the device it is controlling.

2.2.4 Memory Unit

Memory is part of the microcontroller whose function is to store data. The easiest way to explain it is to describe it as one big closet with lots of drawers. If the drawers are marked in a way that they cannot be confused, any of their contents will then be easily accessible. It is

enough to know the designation of the drawer and so its contents will be known for sure. Memory components are exactly like that. For a certain input, the contents of a certain addressed memory location are obtained and that's all. This brings to surface two new concepts-addressing and memory location. Memory consists of all memory locations and addressing is selecting one of them. This means that the desired memory location has to be selected on one hand, and on the other hand wait for the contents of that location. Besides reading from a memory location, memory must also provide for writing onto it. This is done by supplying an additional line called control line, designated as R/W (read/write). The control line is used in the following way: if r/w = 1, reading is done and if opposite is true then writing is done on the memory location. There are several types of memory, namely: ROM, RAM, EEPROMand FLASH.

2.2.5 Read Only Memory (ROM)

ROM is a type of memory used to permanently save the program being executed. The size of the program that can be written depends on the size of this memory. ROM can be built in the microcontroller or added as an external chip, which depends on the type of the microcontroller. Both options have some disadvantages. If ROM is added as an external chip, the microcontroller is cheaper and the program can be considerably longer. At the same time, a number of available pins is reduced as the microcontroller uses its own input/output ports for connection to the chip. The internal ROM is usually smaller and more expensive, but leaves more pins available for connecting to peripheral environment. The size of ROM ranges from 512B to 64kB. However, this type of memory is fast becoming outdated with the development of re-programmable ROMs such as EEPROM and FLASH memory.

2.2.6 Random Access Memory (RAM)

RAM is a type of memory used for temporarily storing data with intermediate results created and used during the operation of the microcontroller. The content of this memory is cleared once the power supply is off. For example, if the program performs an addition, it is necessary to have a register standing for what in everyday life is called the “sum”. For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition. The size of RAM goes up to a few kilobytes.

2.2.7 Electrically Erasable Programmable ROM (EEPROM)

The EEPROM is a special type of memory not contained in all microcontrollers. Its contents may be changed during program execution (similar to RAM), but remains permanently saved even after the loss of power (similar to ROM). It is often used to store values, created and used during operation (such as calibration values, codes, values to count up to etc.), which must be saved after turning the power supply off. A disadvantage of this memory is that the process of programming is relatively slow - it is measured in milliseconds.

2.2.8 Special Function Registers (SFRs)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore. Since their bits are physically connected to particular circuits within the microcontroller, such as A/D converter, serial communication module etc., any change of their state directly affects the operation of the microcontroller or some of the circuits. For example, writing zero or one to the SFR controlling an input/output port causes the appropriate port pin to be configured as input or output. In other words, each bit of this register controls the function of one single pin.

2.2.9 Program Counter

A program Counter is an engine running the program and points to the memory address containing the next instruction to be executed. After each instruction execution, the value of the counter is incremented by 1. For this reason, the program executes only one instruction at a time just as it is written. However the value of the program counter can be changed at any moment, which causes a “jump” to a new memory location. This is how subroutines and branch instructions are executed. After jumping, the counter resumes even and monotonous automatic count +1, +1, +1.

2.2.10 Interrupts

It is mandatory that microcontrollers provide real time response to events in the embedded system they are controlling. When certain events occur an interrupt system can suspend the processor to suspend processing the current instruction sequence and to begin an interrupt service routine (ISR). The ISR will perform any processing required based on the source of the interrupt before returning to the original instruction sequence. Possible interrupts are source dependent, and often include events such as internal timer overflow, an analogue to digital conversion completing, a logic level change on an input such as from a button being pressed and data received from a communication link.

Where power consumption is important as in battery-operated devices, interrupts may also wake a microcontroller from a low power sleep state where the processor is halted until required to do something by a peripheral event.

2.2.11 Timers/Counters

In order to function, the microcontroller needs a timer which provides information about time, duration, protocol etc. The basic unit of the counter is a free-run counter which is in fact a register whose numeric value increments by one in even intervals. If these registers use an internal quartz crystal oscillator as a clock source, then it is possible to measure the time between two events - if the register value is T1 at the moment measurement started, and T2 at the moment it finished, then the elapsed time is equal to the result of subtraction T2-T1. If the registers use pulses coming from external source, then such a timer is turned into a counter. / Most programs use these miniature electronic "stopwatches" in their operation. These are commonly 8- or 16-bit SFRs whose contents are automatically incremented by each coming pulse. Once the register is completely loaded, an interrupt is generated. A variety of timers exist with microcontrollers, one of the most common types being the programmable interval timers (PIT). A PIT just counts down from some value to zero. Once it reaches zero, it sends an interrupt to the processor indicating that it has finished counting - useful for devices such as thermostats which periodically tests the temperature around them to see if they need to turn the air conditioner on, the heater on, etc.Another feature is the time processing unit (TPU) which is essentially just another timer, but more sophisticated. In addition to counting down, the TPU can detect input events, generate output events and other useful operations.

2.2.12 Watchdog Timer

This timer is to ensure a flawless functioning of the microcontroller. Suppose that as a result of some interference (which often does occur in industry), the microcontroller stops executing a program, or worse, it starts working incorrectly. When this happens with a

computer it is simply reset and it will keep working. However there is no reset button to push on a microcontroller to solve this problem. To overcome this obstacle, a block is introduced called watchdog timer. This is in fact another free-run counter where the program needs to write a zero every time it executes correctly. In case that program gets "stuck", zero will not be written in and the counter alone will reset the microcontroller upon achieving its maximum value. This will result in executing the program again, and correctly this time around. And that is an important element for every program to be reliable without man's supervision. The Watchdog Timer is connected to a completely separate RC oscillator within the microcontroller.

2.2.13 I/O (Input-Output) Ports

In order to make the microcontroller useful it is necessary to connect it to peripheral devices and this is done through ports and pins. There are several types of ports – input, output and bidirectional ports. Ports act like a memory location-something is simply being written to it or read from it, and it could be noticed on the pins of the microcontroller. When working with ports it is necessary to choose which one to work with, and then send data to, or receive from.

2.2.14 Oscillator

The function of the oscillator to the microcontroller is akin to the heart in a human body – continuously pumping blood round the body. In like manner, the signal from the oscillator stimulates the microcontroller to execute the program instructions in its memory.

The oscillator generates high frequency square waves of great stability called clock signal for the harmonic and synchronous control of all circuits within the microcontroller. The oscillator is usually a quartz-crystal or ceramics resonator (like RC oscillator). It is important to say that program instructions are not executed at the rate imposed by the

oscillator itself, but several times slower. It happens because each instruction is executed in several steps. For some microcontrollers, the same number of cycles is needed to execute any instruction, while it is different for other microcontrollers. Accordingly, if the system uses a quartz crystal with a frequency of 20MHz, the execution time of an instruction is not expected 50nS, but 200, 400 or even 80nS, depending on the type of the microcontroller.(Seward, 2008):

2.2.15 Serial Communication

As there are separate lines for receiving and sending, it is possible to send and receive data (info.) at the same time. The so-called full duplex block which enables this way of communication is called a serial communication block (Fig. 2.7). Unlike the parallel transmission, data moves here bit by bit, or in a series of bits which defines where the term serial communication comes from. After the reception of data it has to be read from receiving location and stored in memory as opposed to sending where the process is reversed. Data goes from memory through the bus to the sending location, and then to the receiving unit according to protocol. Protocol refers to the set of rules which must be applied in order that devices can correctly interpret data that they mutually exchange. Fortunately, the microcontrollers automatically takes care of this, so the work of the programmer/user is reduced to a simple write (data to be sent) and read (received data).

Parallel connections between the microcontroller and peripherals established over I/O ports are the ideal solution for short distances up to several meters. However, when it is necessary to establish communication between two devices on longer distances it is obviously not possible to use parallel connections. Then, serial communication is the best solution.

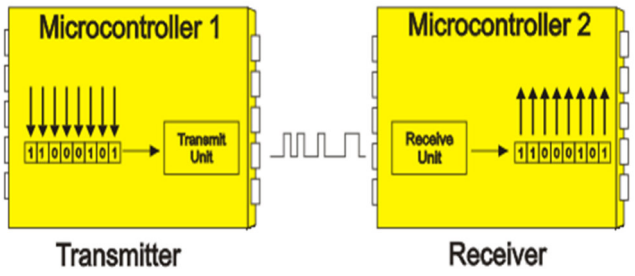

Fig. 2.7: Serial Communication.(Seward, 2008).

Today, most microcontrollers have several different systems for serial communication built in as standard equipment. Which of them will be used depends on many factors of which the most important are:How many devices the microcontroller has to exchange data with?How fast the data exchange has to be?What is the distance between devices?Is it necessary to send and receive data simultaneously?

In a nutshell, a microcontroller is a single integrated circuit commonly found with the following features:

- Central processing unit-ranging from small and simple 4-bit processors to complex 32 or 64 bit processors.
- Discrete input and outputs bits, allowing control or detection of the logic state of an individual package pin.
- Serial input/output such as serial ports.
- Other serial communication interfaces like serial peripheral interface and controller Area Network for system interconnect.
- Peripherals such as events counters, PWM generators and watchdog timers.

- ❖ Some designs include general purpose microprocessor cores, with one or more ROM, RAM or I/O functions integrated onto the same package. Other designs are purposely built for certain control applications.(Jing and Xin-Guang, 2010).

They consume relatively little power (milliwatts) and generally have the ability to remain functional while waiting for events such as a button press or interrupt. Power consumption while sleeping (CPU and peripherals off) may be just nanowatts, making them ideal for low power and long lasting battery.

Microcontrollers have proved to be highly popular in embedded systems since their introduction in the 1970s, the first microcontroller being Intel 8084, released in 1976.

The popularity increased when EEPROM memory was incorporated to replace the one-time programmable ROM memory. With EEPROM, the development cycle of programming, testing and erasing a part could be repeated many times.

Small and efficient, microcontrollers are dedicated to one task and run one specific program which is stored in the ROM and generally does not change. The instruction set usually has many instructions intended for bit-wise operations to make control programs more compact.

2.2.16 Classes of Microcontrollers

There are several kinds of programmable microcontrollers, common types being classified by several parameters such as Bits, Flash size, RAM size, number of input/output lines, packaging type, supply voltage and speed.According to bit size, microcontrollers may be classified into:

- 8 bit Microcontrollers
- 16 bit Digital Signal Controllers (DSC)
- 16 bit General Purpose Microcontrollers
- 32 bit Microcontrollers

Once the microcontroller bit size has been chosen, the search can be narrowed down by its technical attributes, namely RAM, Flash, number of input lines, speed and supply voltage to name a few. Also using these filters, the right LCD, low power, USB, Wireless or PIC microcontroller can be found. Microcontrollers may also be searched for using the manufacturers' names, viz:Atmel Corporation,Cypress, Innovasic,Microchip, NXP, Renesas Electronics,STMicroelectronics, ZILOG.(Jing and Xin-Guang, 2010). Two samples of microcontrollers are shown in Fig. 2.8.

.

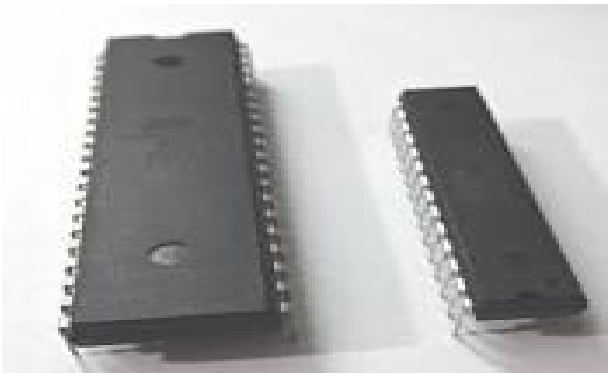

Fig. 2.8: Some Microcontrollers. (Jing and Xin-Guang, 2010).

2.17Microcontroller Applications

Microcontrollers are hidden inside a vast number of products these days. If a microwave ovenhas an LED or LCD screen and a keypad, it contains a microcontroller. All modern automobiles contain at least one microcontroller, and can have as many as six or seven. The engine is controlled by a microcontroller, as are the anti-lock brakes, the cruise control and

so on. Any device that has remote control almost certainly contains a microcontroller – TV sets, satellite decoders and high-end stereo systems all fall into this category. Also included are digital cameras, GSM phones, camcorders, answering machines, laser printers, feature-laden refrigerators, dishwashers, washers and dryers (the ones with display and keypads), as well traffic light controls. Basically, any product or device that interacts with its users has a microcontroller buried inside.

2.2.18 Differences Between Microprocessors and Microcontrollers

The terms microprocessor and microcontroller are often used interchangeably. Both of them of them are IC's and have been designed for real time applications. They share many common features while at the same time have significant differences. They cannot be distinguished by looking at them and are available in different versions starting from 6 pins to as high as 80 to 100 pins or even higher depending on the features.

A microprocessor has only the CPU inside it i.e. only the processing powers such as Intel's Pentium 1,2,3,4, core 2 duo, i3, i5 etc. It does not have RAM, ROM, and other peripherals on the chip. A system designer has to add them externally to make them functional. But this is not the case with microcontrollers. A microcontroller has a CPU, in addition to a fixed amount of RAM, ROM and other peripherals all embedded on a single chip. At times it is termed as a mini computer or a computer on a single chip.

Microcontrollers are designed to perform specific tasks - applications where the relationship of input and output is defined – i.e. depending on the input, some processing needs to be done and output is delivered. Examples of such tasks include keyboards, mouse, washing machine, digicam, pendrive, microwave, cars, bikes, telephone, mobiles, watches, etc. Since

the applications are very specific, they need small resources like RAM, ROM, I/O ports etc and hence can be embedded on a single chip. This in turn reduces the size and the cost.

On the other hand, Microprocessors find application where tasks are unspecific like developing software, games, websites, photo editing, creating documents, PCs (personal computers) etc. In such cases the relationship between input and output is not pre-defined. They need high amount of resources like RAM, ROM, I/O ports etc.

The clock speed of the Microprocessor is quite high as compared to that of a microcontroller. Whereas the microcontrollers operate from a few MHz to 30 to 50MHz, today's microprocessors operate above 1GHz as they perform complex tasks - the microprocessor in some PCs have a speed of over 250 GHz.(Shen et al., 2009).

Undoubtedly a microcontroller is far cheaper than a microprocessor. However a microcontroller cannot be used in the place of a microprocessor and using a microprocessor is not advised in the place of a microcontroller as it makes the application quite costly. A microprocessor cannot be used stand alone. It needs other peripherals like RAM, ROM, buffer, I/O ports etc [connected externally] and hence a system designed around a microprocessor is quite costly.

Today different manufacturers produce microcontrollers with a wide range of features available in different versions, examples being Intel 8048, Intel 80386, Intel 8742, 8051 microcontrollers(8051 being the most commonly used microcontroller), PIC microcontroller (which is a family of Harvard architecture microcontrollers made by Microchip Technology and many others). The other manufacturers of microcontrollers include ATMEL, Microchip, TI, Freescale, Philips, Motorola etc. Microprocessors include Intel's Pentium 1,2,3,4, core 2 duo, i3, i5 etc.

CHAPTER THREE

METHODOLOGY, CONSTRUCTION AND TESTING

3.0 INTRODUCTION

Being a digital device, there is no analogue circuit that may be set up to produce this digital frequency counter except by the use of logic elements such as ICs (Integrated circuits) and microcontrollers or microprocessor. To produce this digital frequency counter by the means of ICs would require an arrangement of a large number of IC's thus making the circuit bulky. This is because ICs are usually made to perform one peculiar function or the other and which cannot be altered by the user.

Now, the functions of a large number of ICs can be performed using software (computer program). Such software is installed inside a microcontroller or microprocessor which performs the programmed tasks. This way, a microcontroller chip can be programmed to carry out the work of numerous IC's put together, thereby producing small, simple circuits that perform complex tasks. This is consonant with the drive in today's world which is towards producing simple, minute circuits that carryout delicate and complex function.

Hence the procedure followed for the design of the circuit, simulation, construction and testing of this digital frequency counter (DFC) was as follows:

3.1 THE CIRCUIT DESIGN

The design of this frequency counter circuitwas based on the fact that a microcontroller is a computer on a single chip which is programmed to perform a specified task, and then

provided with the right enabling environment in which to operate,theenvironment beingthe presence of (Seward, 2008):

i) Quartz Crystal Oscillator,

ii) Power Supply of 5V dc

iii) Reset.

iv) Program code

Using simulator software, namely Proteus Isis Circuit Simulator, a general purpose microcontroller AT89C51 was selected from the list of microcontrollers in-built, whose features are listed as follows (AllDataSheets, 2016):

- an 8-bit microcontroller produced by Atmel Corporation, U.S.A.,
- having 4KB of Flash programmable and erasable read only memory (PEROM) ,
- with 128 bytes of RAM (volatile memory),
- erasable and reprogrammable up to a maximum of 1000 times,
- inbuilt UART for serial communication,
- having 6 interrupts -two timer/counters and four hardware interrupts,
- With 40 pins, with four ports designated as P_1, P_2, P_3 and P_0(all 8-bit bi-directional ports, *i.e.*, usable as both as input and output ports).

The block diagram of the design of the frequency counter constructed is shown in Fig. 3.1.

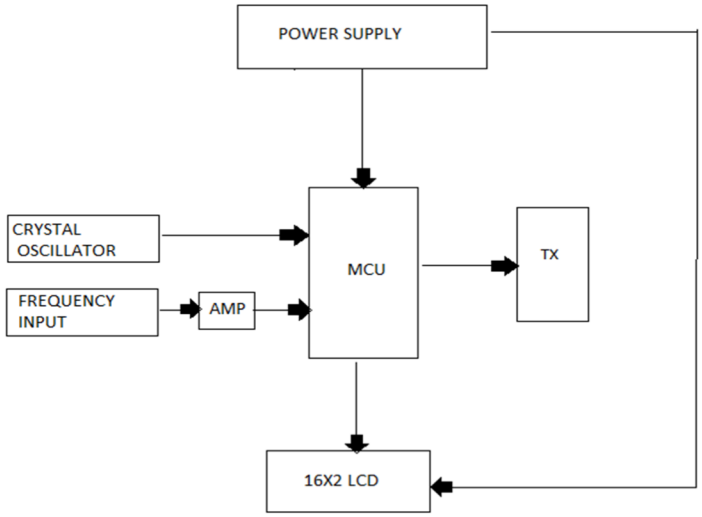

Fig. 3.1: The DFC's Circuit Design Block Diagram

3.2 THE DESIGN PROCESS

1. After selecting the microcontroller on the simu supply unit of 5.0 V dc was set up needed to power circuit, consist down transformer, rectifier, filter and regulator. The transformer steps d s voltage from 220Vac to 12Vac. A full wave bridge rectifier converts t to dc. The electrolytic filter and smoother connected to the rectifier deliver a clean 12V dc which is regulated to a stable 5V dc by an RS206L Regulator.
2. Next, a quartz crystal oscillator rated 11.0952 MHz was called up and connected to the pins designated as XTAL 1 and XTAL 2 on the microcontroller, along with a Reset Block, an Input Amplifier, a Buffer and a 16 x 2 LCD for the output display of the frequency counted.
3. The LCD was connected to the microcontroller via port 2 from pins P2.0 to P2.6.

4. The reset block which was connected to both the power supply's Vcc and the ground, was joined to the microcontroller via the pin designated as RST. (The Reset for this microcontroller sends logic "0" to it to activate it at power ON. If this reset were not present the microcontroller would not be able to function.)(Shen et al., 2009).
5. The input amplifier in the circuit is to amplify the signal to a level readable by the microcontroller.
6. With the circuit set up completed, a program code (displayed in Appendix A1) was written on Keil µVision3, a C program editor, for the microcontroller to act as a digital signal frequency counter.Fig. 3.2 shows the actual circuit diagram of the Frequency Counter as formed on the Proteus Isis Simulator screen with the microcontroller AT89C51 connected to the 16 x 2 LCD, quartz crystal oscillator, reset, input amplifier, wireless transmitter and to the power supply unit (PSU).

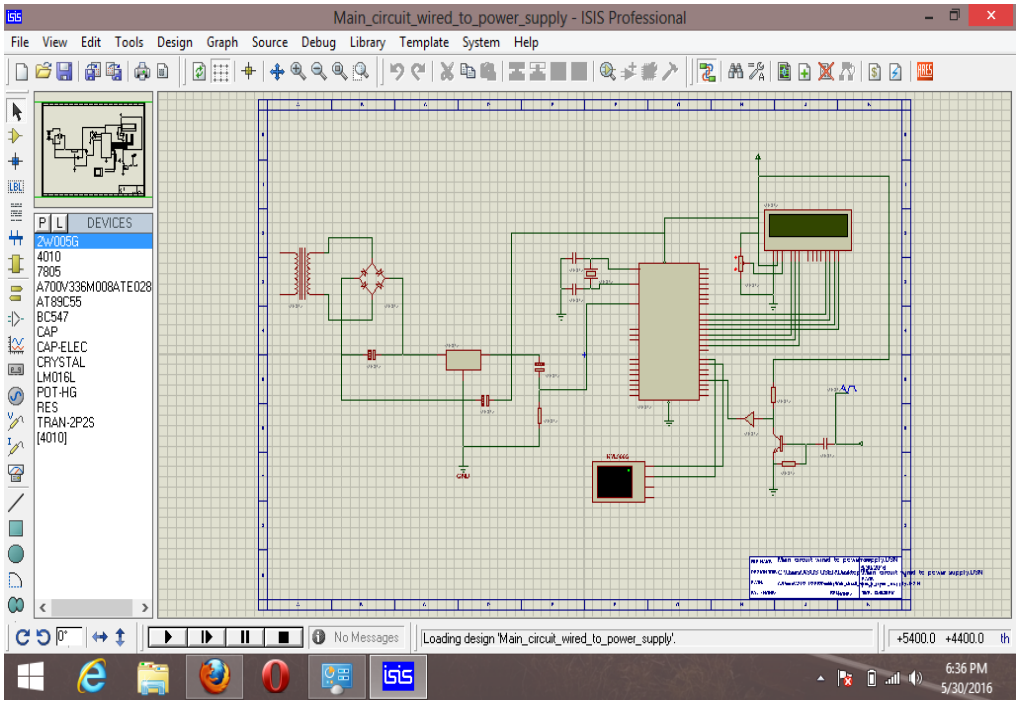

Fig. 3.2: The intended frequency counter's schematic on Proteus Simulator

3.3 LINKING THE PROGRAM CODE TO THE MICROCONTROLLER INSIDE THE SIMULATOR

When the circuit set up was completed inside the Proteus simulator a program code was written in C+ language on another software, KeilμVision3, which is a C-editor(see Appendix) which is the program executed by the microcontroller in this circuit to function as a digital frequency counter. When the program writing was completed, the option "COMPILE" was clicked on the C-Editor. This made the editor to immediately compile and also automatically convert that program into anhexadecimal format and saved it in a file called "HEX. FILE".

To link this Hex.file to the microcontroller AT89C51 inside the simulator the steps were as follows:

1. The microcontroller in used was called out in the simulator and then right-clicked.
2. A dialog box appeared and the option "PROGRAM" was selected by clicking on it.
3. A "PROJECT" directory came up from which the folder containing my hex.file was highlighted.
4. "OK" button on the simulator was clicked.

Followed by clicking on the "RUN" button, thereby completing the transfer process of the hex.file into the microcontroller's memory on the simulator.

Screen shots of sections of the program code and the corresponding hex file are displayed in Figs. 3.3 and 3.4respectively.

Fig. 3.8: The DFC and the 1 MHz generator used to supply digital signals to it for testing.

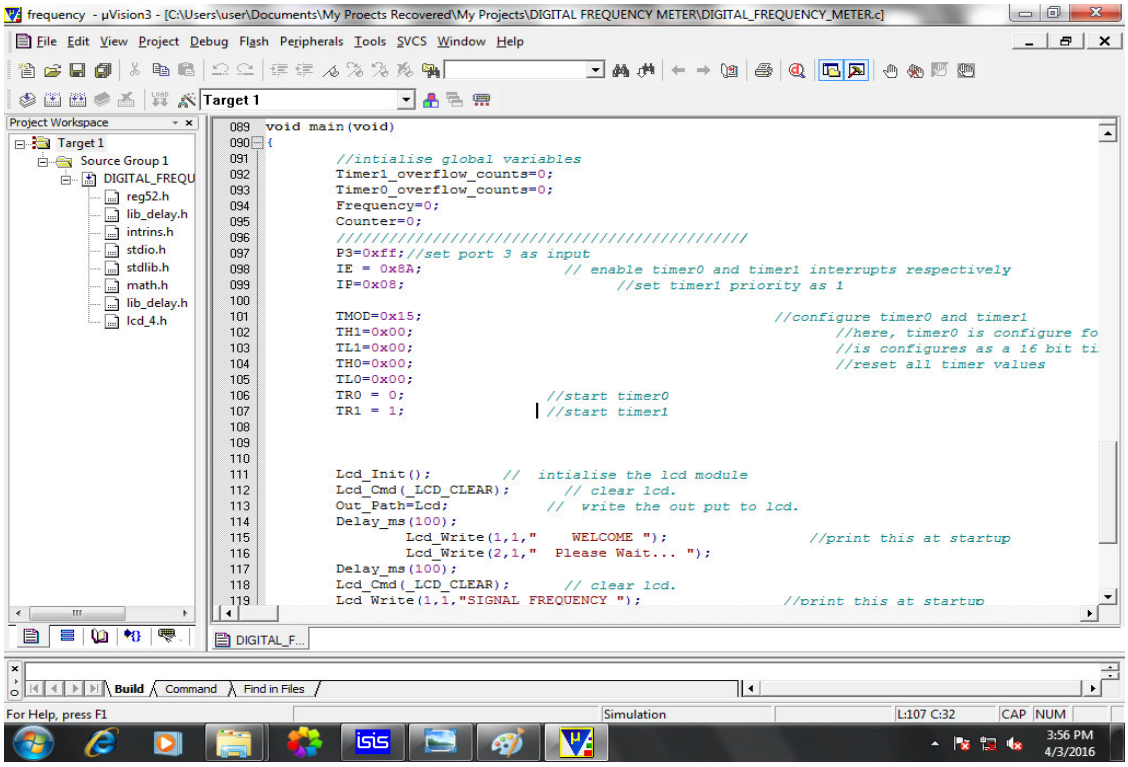

Fig. 3.3: A Screen Shot of a section of the Source Code on Keil.

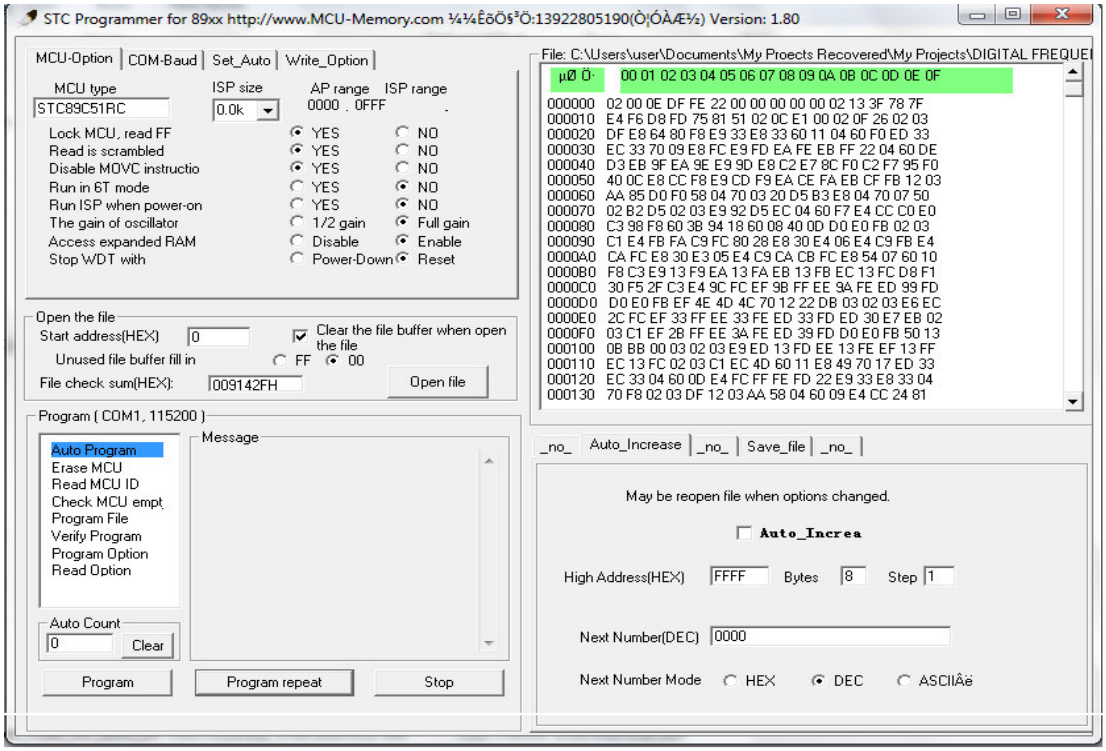

Fig. 3.4: A Screen Shot of a section of the Hex file also on Keil.

3.4 THE SIMULATION

With the hex file now inside the microcontroller's memory in the Proteus Simulator, the circuit was now ready for simulation, thus:

1. A square wave signals generator was called up onto the screen.
2. Placing the computer cursor on the signal input point of the DFC's circuit on the Proteus screen and then double clicking, the signals generator presented a dialog box on which to enter a desired frequency value.
3. With a sample signal frequency value of 10,000 Hz entered into the dialog box and its 'OK' tab clicked, and then the 'RUN' tab of the Simulator clicked, all points in the circuit on the screen became vividly activated (blinking with red and blue lights) and after some seconds the circuit's LCD displayed a frequency count approximately equal in value to the applied value i.e., 9.989 kHz.
4. An inbuilt frequency meter earlier called out onto the Proteus screen (for the purpose of cross checking the proposed DFC's performance with its own count) measured precisely the frequency applied, 10,000 Hz, as shown on its display on the simulator.
5. Fig. 3.5 is a screen shot of simulating the DFC using the 10 kHz test input signal.

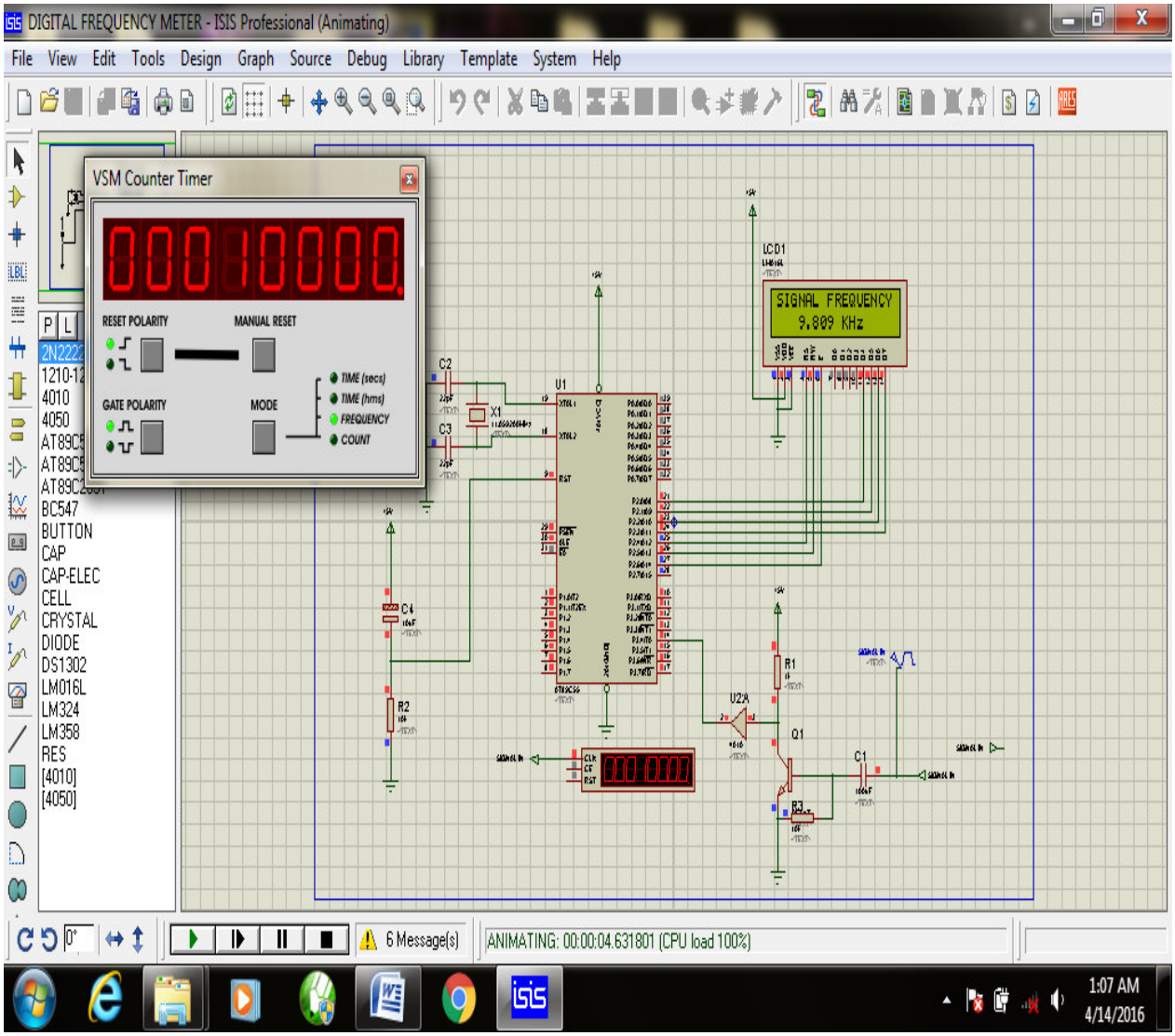

Fig. 3.5: Simulating with a 10 kHz input – the DFC displays a count of 9.889 kHz

3.5 LIST OF COMPONENTS

For this DFC,the following were the components used:

- MICROCONTROLLER AT89C51, a member of the 8051 family of microcontrollers.
- LCD----------LM016L
- R1----------10K
- R2----------10K
- R3----------10K
- R4----------1K
- C1----------22pF

- C2----------22pF
- C3----------180µF
- C4----------100µF
- Q1----------BC547
- C1(PSU)----------100F
- C2(PSU----------100µF
- BRIDGE RECTIFIER(PSU----------2W005G
- VERO BOARD
- CONNECTING WIRES
- CASING FOR THE CIRCUIT
- PROBES FOR INPUT SIGNALS
- VOLTAGE REGULATOR RS206L

3.6 CONSTRUCTION

The components were soldered together as set out on the Schematic in fig. 3.2 using a vero board. The transformer of the Power Supply Unit (PSU) was first soldered followed by four diodes, the resistor, capacitor and regulator RS206L Next, the microcontroller was soldered very carefully on a prominent area of the board. It was linked on the board from the PSU's capacitor to its "Vcc" and then to the "GND". After soldering the reset block's capacitor and resistor accordingly, it was linked to the microcontroller at its RST pin from the capacitor while the resistor was connected to the PSU's "GND". The DFC's input block, consisting of the transistor, capacitor, resistor and buffer was soldered accordingly and then linked to the microcontroller at its TX0 point and also to the PSU's positive and GND points. The LCD was joined by its belt to the microcontroller at pins P2.0 to P2.6 of Port 2 .The quartz crystal oscillator was soldered to the board and connected to the microcontroller at the point designated as XTAL 1 and XTAL2, with it "gnd" joined to that of the PSU. The main's connection to the transformer was intercepted on one leg using an ON/OFF switch. The input signal was provided for by connecting a probe to the input point.

3.7 TRANSFER OF HEX. FILE INTO MICROCONTROLLER BEFORE PLACEMENT INTO CONSTRUCTED CIRCUIT

The hex file produced by the Keil C-Editor was first passed into the real microcontroller AT89C51 purchased in the market before it was soldered into the vero-board to join the rest of the components. To do this transfer, a factory–made circuit called “PROGRAMMER” or “BURNER” which contained a microcontroller (MC) socket was used.

The MC was placed into that socket and the USB cable of the programmer was connected to a Laptop PC. The programmer used here was an STC Programmer (STC being a brand name). So the STC Programmer’s Software was first installed inside the PC which would work with that particular programmer. The STC software was then launched in my laptop and the following steps were used:

1. Clicking on “DEVICE TYPE” the MC in question (i.e. AT89C51) was selected.
2. “OPEN” button was clicked to bring up a dialog box from which my relevant hex file was selected.
3. And then clicking on “PROGRAM” to “Burn” or “Program” the hex file into the MC’s Chip.

On the STC software screen, a progress bar was displayed, showing how far gone in the transfer. At the end of the process, a status message of “SUCCESS” was displayed at the bottom of this screen confirming the transfer to the MC.

Therefore the STC Programmer was unplugged from the PC, the MC was removed from its socket on the programmer and then taken to the DFC circuit where it was soldered finally,

as explained in section 3.6 above. Fig. 3.6 below shows a screen shot of the STC Programmer software during the hex file transfer.

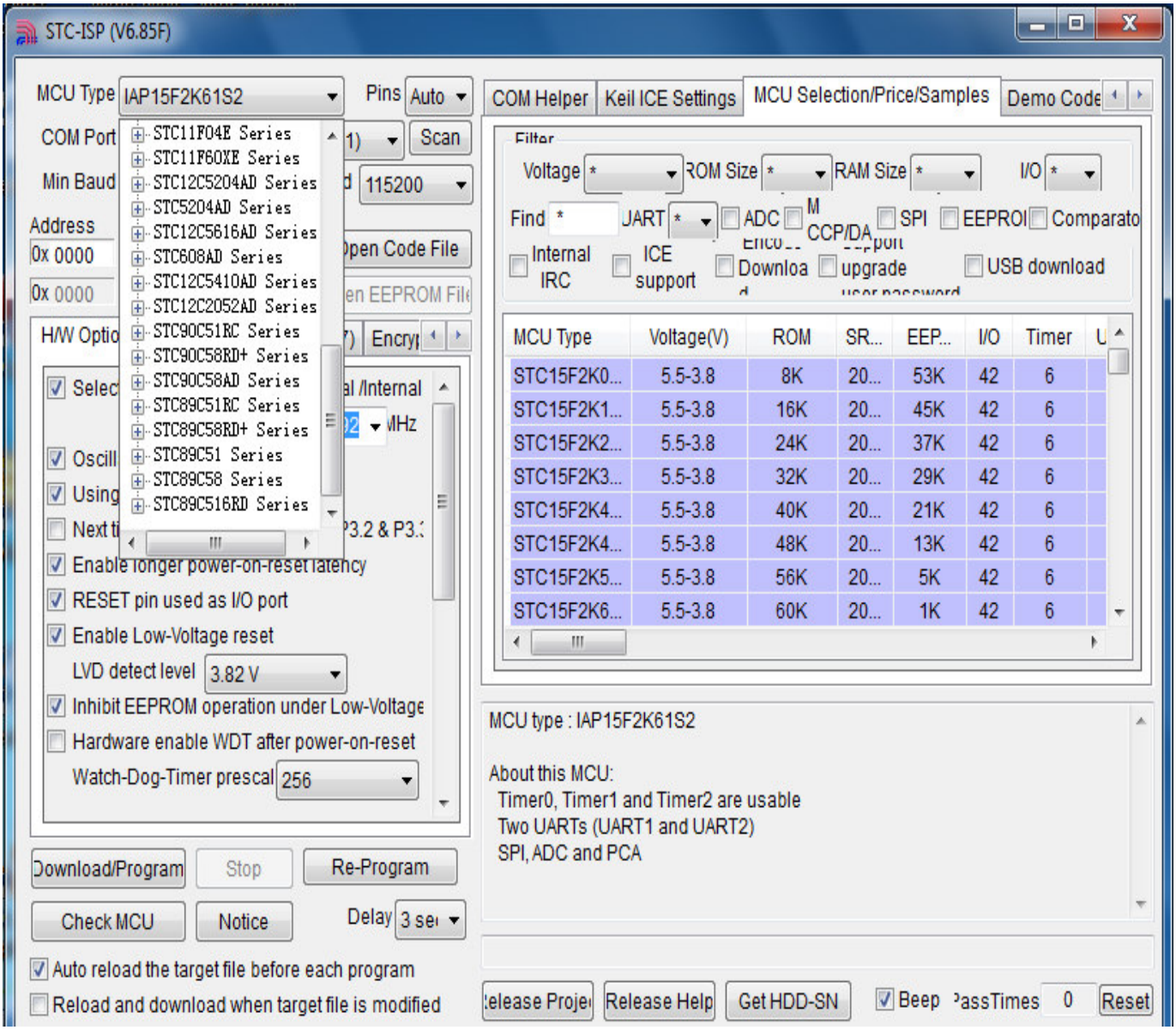

Fig 3.6: STC Software ready to transfer hex file to hardware microcontroller

3.8 OPERATING PRINCIPLES OF THE CIRCUIT

1. In this circuit the main function is carried out by the microcontroller AT89C51 whose role is to execute the Source Code saved to its memory via a Program Burner.
2. The working of the circuit is based on the definition of frequency*):*

Frequency = number of pulses/second …….. Eq.1(*Floyd, 2010)*

3 It makes use of two internal registers of the microcontroller, namely the Counter and Timer. The counter register is configured to enable the microcontroller count the number of incoming signals passing through the gate, while the timer register records the time taken. (*Floyd, 2010)*.

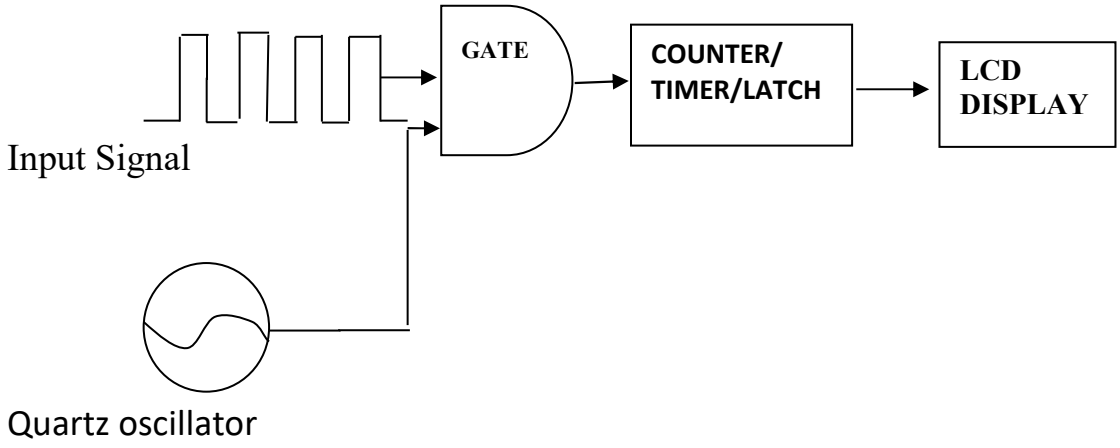

Fig. 3.7: The Operational Block Diagram of the ConstructedDFC.

4 The program code in the microcontroller harmonizes the counter and timer values to tell the signal's frequency according to eq. 1 above.

5 The gate can be seen in the operational block diagram in Fig. 3.7.

6 The input signal is enabled to pass through the microcontroller's gate by the clock pulse from the crystal oscillator, which also gives the timing for the microcontroller to execute the instructions in its memory - here, one instruction per clock pulse. (In fact, the microcontroller would not function in the absence of a clock pulse since it is the microcontroller's heartbeat.) (Moss, Ronald J.T., 2005)

7 At a preset gate time of 1 second, monitored by the timer register of the microcontroller, the number of complete pulses of the signal entering gate is counted by the counter register. For example, with the gate time of 1 second, if 1,000,000 pulses are counted then the signal's frequency is 1MHz.(Jing & Xin-Guang, 2010).

8 The frequency thus measured is transferred to the latch, and from there onto the LCD for display. The function of the latch is to hold the last posted value until when updated by the counter/timer. (*Floyd, 2010)*.

9 According to the program code, if the frequency of the input signal is between OHz and 1000 Hz, the result would be displayed on the LCD in Hz while if it is above 1000 Hz, the count would be in kHz.

3.9 TESTING THE CONSTRUCTED DFC

On the Proteus Simulator, a series of random signal frequencies were applied to the input of the DFC and its responses read off from its LCD. The responses of the DFC inside Proteus were also monitored for comparison with those of the DFC.

After construction, a hardware digital audio signals generator named "YUFAIR", rated 1MHz, was used to input various digital signal frequencies to the DFC for testing. Fig. 3.8 shows the DFC with the test signals generator.

Fig. 3.8: The DFC and the 1 MHz generator used to supply digital signals to it for testing.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 RESULTS

Tables 1 displays the results obtained when the intended counter was tested during simulation while Table 2 shows those of testing the actual construction.

Table1: Results of Testing the DCF while on Proteus Simulator.

S/No.	Input Signal Frequency (Hz)	Frequency Displayed by DFC (Hz)	Count Delay by DFC (sec)
1	50.00	50.00	0 4
2	100.00	100.00	0 4
3	200.00	199.00	0 2
4	250.00	246.00	0 4
5	300.00	298.00	0 3
6	350.00	347.00	0 4
7	400.00	397.00	0 4
8	1.00 K	985.00	0 2
9	2.00 K	1.974 K	0 4
10	5.00 K	4.944 K	0 2
11	10.00 K	10.094 K	0 3
12	15.00 K	15.126 K	0 4
13	20.00k	20.168 k	0 2
14	25.00k	25.210 k	0 4
15	50.00k	50.420 k	0 4
16	100.00k	100.040 k	0 3
17	200.00k	201.675 k	0 4

Table 2: Results of Testing the DFC with a hardware “Yufair” Digital Audio Generator

S/No	Input signal Freq. (Hz)	Freq. Counted By DFC (Hz)	Time delay Before DFC Display count (second)
1	10.00	11.00	0 1
2	15.00	17.00	0 1
3	20.00	22.00	0 1
4	30.00	32.00	0 1
5	40.00	42.00	0 1
6	50.00	52.00	0 1
9	80.00	81.00	0 1
10	90.00	92.00	0 1
11	100.00	102.00	0 1
12	100.00	108.00	0 1
13	150.00	162.00	0 1
14	200.00	213.00	0 1
15	300.00	310.00	0 1
16	400.00	409.00	0 1
17	500.00	498.00	0 1
18	800.00	792.00	0 1
19	10.00 k	10.809 k	0 1
20	15.00 k	15.716 k	0 1
21	20.00 k	20.803 k	0 3
22	50.00 k	49.131 k	0 3
23	100.00 k	107.80 k	0 2
24	150.00 k	150.69 k	0 2
25	200.00 k	200.44 k	0 2
26	250.00 k	205.15 k	0 2
27	300.00 k	300.37 k	0 2
28	350.00 k	350.29 k	0 2
29	400.00 k	400.65 k	0 3
30	450.00 k	-	-
31	500.00 k	-	-

4.2 DISCUSSION

When on the simulator, the differences between the counts of the proposed frequency counter and those of the in-built frequency meter were negligible. Likewise after construction, the differences between the counts of the DFC and those of the hardware signals generator were little .However, these differences could be attributed to the following two factors:

i. Possible imperfections in the values of the circuit components that were used to producethe DFC.

ii. Stray capacitances, resistances, impedances (etc.) that may have occurred during the soldering of the components.

The delay between when the input signal is applied to the DFC and output count is shown on its LCD is between 01 and 04 seconds, both during simulation and after construction, which is a fair performance compared to the response times of common measuring devices such as a voltmeter or ammeter.

4.3 CONTRIBUTION TO KNOWLEDGE

This research has succeeded in producing a portable, hand-held version of a digital frequency counter, a departure from the relatively bulky sizes of most of the available DFCs. This is consonant with the drive in today's world which is towards simple circuits that perform complex functions, such as is obtainable in our modern day GSM telephone handsets. Hitherto, electronics was at the mercy of complex and large circuits that performed simple, peculiar functions as with the radio receivers and calculator machines of 1960s.

Secondly, in this work one is enlightened on how to use a software circuit simulator to design, simulate and implement an electronic circuit, especially one involving a microcontroller.

Thirdly, the DFC produced here can be used to measure frequencies of up to 400 kHz in digital form since frequency counters are not common instruments.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECCOMMENDATION

5.1 SUMMARY

Virtually any electronic device which exhibits **"**intelligence**"** surely has one or more microcontrollers embedded and working quietly and diligently inside it. This is attributed to the fact that a microcontroller can perform various tasks and only needs to be appropriately programmed for a specified function and then provided with the right operating environment and peripherals.

This condition facilitated the implementation of this project whereby the microcontroller AT89C51 was used to carry out the task of determining the frequencies of pulse signals.

5.2 CONCLUSION

The digital frequency counter intended in this research was designed, simulated and constructed and can measure pulse signals up to 400 kHz, well above the audio range of 20Hz to 20 KHz. And its response time for displaying the count is from 1 to 4 seconds.

5.3 RECCOMMENDATION

The results obtained from testing the constructed device were impressive except that the values being displayed by the DFC were a little less or more than the actual values input to it. For instance, when a frequency of 250 Hz was selected from the scale of the hardware 'Yufair' signals generator, the DFC showed a count of 246.00 Hz on its LCD. The reason for this difference could not be established precisely up till the end of this work. Thus it is hoped that the cause would be discovered later and then the performance of this DFC improved upon.

REFERENCES:

Fu, B. H., Liu, K. X., & Ma, J. C. (2014). Design and Construction of Oilfield Digitization System Logs. *Applied Mechanics and Materials*, *687-691*, 2569–2572. http://doi.org/10.4028/www.scientific.net/AMM.687-691.2569

Jing, Z., & Xin-Guang, L. (2010). The Front Design and Implement of Direct Digital Frequency Synthesizer Based on FPGA. *Electrical and Control Engineering (ICECE), 2010 International Conference on*, 4816–4819. http://doi.org/10.1109/iCECE.2010.1165

Langlois, J. M. P., & Al-Khalili, D. (2002). A new approach to the design of low power direct digital frequency synthesizers. *Proceedings of the 2002 IEEE International Frequency Control Symposium and PDA Exhibition (Cat. No.02CH37234)*, 654–661. http://doi.org/10.1109/FREQ.2002.1075963

Margolin, L. G., & Rider, W. J. (2005). The design and construction of implicit LES models. *International Journal for Numerical Methods in Fluids*, *47*(10-11), 1173–1179. http://doi.org/10.1002/fld.862

Shen, G. S. G., Xie, K. X. K., Lu, L. L. L., Xu, Q. X. Q., Wu, X. W. X., Luo, E., & Chang, Z. C. Z. (2009). Design and Application of a Digital Filter of Mains Frequency. *2009 WRI World Congress on Computer Science and Information Engineering*, *7*, 325–327. http://doi.org/10.1109/CSIE.2009.121

Turner, M. D., Capron, L., Laurence, R. L., & Conner, W. C. (2001). The design and construction of a frequency response apparatus to investigate diffusion in zeolites. *Review of Scientific Instruments*, *72*(12), 4424–4433. http://doi.org/10.1063/1.1408931

Weng, J. H., Yang, C. Y., & Jhu, Y. L. (2011). A low-power direct digital frequency synthesizer

using an analogue-sine-conversion technique. *Proceedings of the International Symposium on Low Power Electronics and Design*, 193–197. http://doi.org/10.1109/ISLPED.2011.5993635

Seward, N.B., (2008). *Microcontrollers: Designs and Applications*. 5thed. New York U.S.A. Prentice Hall Ltd. Pgs 200-210.

Choudhary, H. (2016). *Basics of Using a Frequency Counter*. Retrieved from:

http://www.radio-electronics.com/info/t_and_m/frequency_counter/how-to-use-using-frequency-counter.php (10/02/2015 10:52:00)

AllDatasheets (2016). *Electronics Components Datasheet Search*.Retrieved From http://www.alldatasheet.com/view.jsp?Searchword=AT89C51 (03/05/2015 18:05:00)

Floyd, T. I. (2009). *Digital Fundamentals*. 10thed. London. Pearson Education Ltd., pgs. 724-736.

Moss, K., Ronald J.T. (2005): *Design of Digital Systems*. 8thed. Prentice Hall Ltd, Washington D.C., pgs 63, 91, 93.

Poole, I.(2015). *Frequency, Accuracy and Resolution*.

Retrieved from http://radio-electronics.com (10/02/15 15:08:50)

APPENDIX

Source Code –the Program for the AT89C51

The following is the program written for the AT89C51 microcontroller on KEIL µVISION3 software (a C Program Editor), for it to function as a digital frequency counter (DFC).

```
#include<reg52.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <string.h>
#include <ctype.h>
#include <Lib_Uart.h>
#include <Lib_Delay.h>

#define LCD_STYLE
#define LCD_PRINTF
#include <lcd_4.h>

////////////////////////////////////////////////////////////////
      ///////
volatile float Frequency;
volatile float Timer_Count;
volatile unsigned char Counter;
unsigned short Timer1_overflow_counts;
unsigned short Timer0_overflow_counts;
////////////////////////////////////////////////////////////////
//                                        converts HEX to decimal in base ten
      //
////////////////////////////////////////////////////////////////
volatileintHex_To_Int(intHex,char bits)
{
int Hex_2_Int;
char byte;
      Hex_2_Int=0;

for(byte=0;byte<bits;byte++)
      {
if(Hex&(0x0001<<byte)) Hex_2_Int+=1*(int)(pow(2,byte));
else Hex_2_Int+=0*(pow(2,byte));
      }
```

```
return Hex_2_Int;
}
////////////////////////////////////////////////////////
//                    Timer 0 overflow interrupt routine
      //
////////////////////////////////////////////////////////
void ISR_Timer0(void) interrupt 1
{
      Timer0_overflow_counts=Timer0_overflow_counts+1;
      TF0=0;                  //clear timer0 overflow flag
      TH0=0x00;              //reset the timer values
      TL0=0x00;
}
////////////////////////////////////////////////////////
//                          Timer 1 overflow interrupt routine
       //
//            The frequency of the signal is calculated as thus;          //
//            (TH0x256)+TL0 +(Timer0 _overflow_counts*65536).
       //
////////////////////////////////////////////////////////
void ISR_ex0(void) interrupt 3              //interrupt no. 1 for Timer 0
{
      TF1=0;       //clear timer1 over flow flag
      TH1=57;                 //Reload timer values
      TL1=176;
      Timer1_overflow_counts++;
if(Timer1_overflow_counts==20)                              //(40x25)ms
       ==1000ms==1s
      {
            TR0 = 0;             //stop timer0
Timer_Count=(float)Hex_To_Int(TH0,8);                  //convert timer0 values to
       decimal
            Frequency=(float)Hex_To_Int(TL0,8);

            Frequency=(float)((256*Timer_Count)+Frequency);          //compute
       the frequency of the signal
            Frequency+=(float)(Timer0_overflow_counts*65536);
            Timer0_overflow_counts=0;

      TH0=0x00;                          //reset timer0 values for another
       count cycles
      TL0=0x00;
      TR0 = 1;             //start timer
      Timer1_overflow_counts=0;                      //reset
       Timer1_overflow_counts
      }
return;
```

```
}


////////////////////////////////////////////////////////////////////
      ///////
//                                                  BODY OF THE MAIN
      PROGRAMME
      //
////////////////////////////////////////////////////////////////////
      ////////


void main(void)
{
         //intialise global variables
      Timer1_overflow_counts=0;
      Timer0_overflow_counts=0;
      Frequency=0;
      Counter=0;
      ////////////////////////////////////////////
      P3=0xff;//set port 3 as input
      IE = 0x8A;              // enable timer0 and timer1 interrupts respectively
      IP=0x08;                    //set timer1 priority as 1

      TMOD=0x15;                                  //configure timer0 and timer1
      TH1=0x00;                                     //here, timer0 is configure for 16
       bit counter with external clock source(T0),while timer1
      TL1=0x00;                                    //is configures as a 16 bit timer.
      TH0=0x00;                                     //reset all timer values
      TL0=0x00;
      TR0 = 0;              //start timer0
      TR1 = 1;              //start timer1

Uart_Init(9600);

Lcd_Init();       //  intialise the lcd module
      // clear lcd.
Out_Path=Lcd;          //  write the out put to lcd.
Delay_ms(100);
Lcd_Write(1,1,"    WELCOME ");              //print this at startup
Lcd_Write(2,1,"  Please Wait... ");
Delay_ms(100);
Lcd_Cmd(_LCD_CLEAR);      // clear lcd.
Lcd_Write(1,1,"SIGNAL FREQUENCY ");              //print this at startup

      ////////////////////////////////////////////////////////////
      /////
```

```
while(1)
      {                         //Out_Path=Lcd;
                  //Lcd_Cmd(_LCD_CLEAR);
Lcd_Cursor(2,4);
if(Frequency<1000)

printf(" %4.3f Hz   ",Frequency);//print the signal frequency with auto range
////                                    Out_Path=Serial_Com_Port;
////                                          printf(" %4.3f Hz   ",Frequency);
////                                          Out_Path=Lcd;
////                                          Delay_ms(100);

//                          else if(Frequency>=1000&&Frequency<1000000)
      //if signal above 1000, print in kilo hertz
//                  {
//                                    printf("%4.3f KHz  ",Frequency/1000);
//                                          Out_Path=Serial_Com_Port;
//                                          printf("%4.3f KHz  ",Frequency/1000);
//                                          Out_Path=Lcd;
//                                          Delay_ms(100);
//            }
//                          else if(Frequency>=1000000)
//                          {
//                   printf("%4.3f MHz  ",Frequency/1000000);
//
//                                          Out_Path=Serial_Com_Port;
//                                          printf("%4.3f MHz  ",Frequency/1000000);
//                                          Out_Path=Lcd;                   //if signal above
      1000000, print in Mega hertz
//                Delay_ms(100);
//                          }
}                          }
```