# DESIGN AND CONSTRUCTION OF A COMPUTER - AIDED FREQUENCY READER (CAFR) 0-256Hz; 0-256kHz; 0-256MHz

BY

VANNI VALERIE ADERINMOLA EHI

93/4134

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SCHOOL OF ENGINEERING & ENGINEERING TECHNOLOGY
FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA
NIGER STATE, NIGERIA.

THIS PROJECT IS SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF BACHELOR OF ENGINEERING. (ELECTRICAL AND COMPUTER ENGINEERING).

MARCH 2000

# CERTIFICATION

I ,**VANNI VALERIE A.E. (93/4134)** hereby certify that this Project "Design and Construction of a Computer Aided Frequency Reader (CAFR) (0-256Hz); (0-256KHz); (0-256MHz) was executed by me. Engr. Danjuma of the Department of Electrical & Computer Engineering, School of Eng. & Eng. Technology, Minna, Niger State, ably supervised it.

| | |
|---|---|
| [illegible] | 24/03/2000 |
| **Supervisor** | **Date** |
| **Engr. Danjuma** | |
| [illegible] | 6/4/00 |
| **Head of Department** | **Date** |
| **Engr. (Dr) Y.A. Adediran** | |
| [illegible] | 6/4/2000 |
| **External Examiner** | **Date** |
| [illegible] | 24-03-00 |
| **Vanni Valerie A.E.** | **Date** |

# DEDICATION

To the El-Shaddai My Everything

## ACKNOWLEDGEMENT

This work would not be complete without acknowledging the following people and their contributions in my life and in particular to this work.

| | |
|---|---|
| My Parents | Thank you for your support. |
| My Mother | A mother in Israel, thank you for the prayers, love, time, encouragement and for the lovely genes. |
| My Siblings | Y'all are a great bunch. |
| My Pastors | Thank you for feeding me the Bread of Life. |
| HOF Campus Fellowship | My spiritual supporters thank you. |
| My Friends | Thanks for always being on stand-by. You guys are really appreciated and y'all mean a lot to me. |
| My Project Partner | For your patience. |
| My Supervisor | For your time, interest, painstaking support and encouragement. |
| My Lecturers | Each of you has contributed to making me who I am. |

## ABSTRACT

This Project is designed as a measuring device, which displays its output on a computer monitor. It is a hardware peripheral device driven by a program and interfaced with the Central Processing Unit of a computer. The range for this design is 0-256 Hz; 0-256kHz and 0-256MHz as specified.

It basically measures/reads the unknown frequency at the strobe input and converts it to a form that is readily recognized by the Computer. The Computer program written in C language operates on the input and performs a number of calculations. The output is displayed on the computer monitor for the user to see.

The frequency range is not limited i.e. infinity. Like most instruments and equipment, there is a range limitation. When compared with similar projects at this level, however, it must be noted that the Hz, kHz and MHz bands of range covered is quite wide.

The CAFR is more user friendly, faster and more accurate than conventional analogue or digital multimeters. It is in live with the current trend of computer aided tools machines and instrument.

It is designed to meet the needs of both amateur and professional scientists, engineers and anyone with a rudimental knowledge of computers.

# TABLE OF CONTENT

# CHAPTER ONE

## 1.1 INTRODUCTION

The Computer - Aided Frequency Reader (CAFR) is basically a software driven peripheral device. Its main function is to count the frequency of a received signal and display the output at the Computer terminal. It is interfaced to the Central Processing Unit (CPU) of the computer by means of a parallel port.

The CAFR employs a very basic step-by-step modular principle of operation via the software on the computer terminal, the user selects the range of the frequency to be measured. The receiving unit allows the frequency to be passed through it. The receiver then passes it on to the appropriate divider module based on the range selected by the user. The counter section of the hardware receives the selected frequency after division and counts it. The resulting data is passed to the Central Processing Unit via the interface section. The driver software handles the calculation and conversion of the received result into the correct value with appropriate range (i.e. Hz, kHz or MHz). The output can be read from the monitor of the computer terminal in digits.

The division of the frequency from MHz and kHz into Hz range is done for ease of counting and increased operation speed.

The frequency range for this particular design was chosen to be between 0-256Hz, 0-256kHz and 0-256MHz.

Based on hardware and software, the CAFR can very easily be divided into those two major sub-systems.

## A. HARDWARE SECTION

This section is basically made up of all the components, which can be touched by the hand, i.e. by the user. This section can be further broken down into the data selector, frequency divider, and frequency counter and interface unit as shown in the block diagram fig 1.1 below.

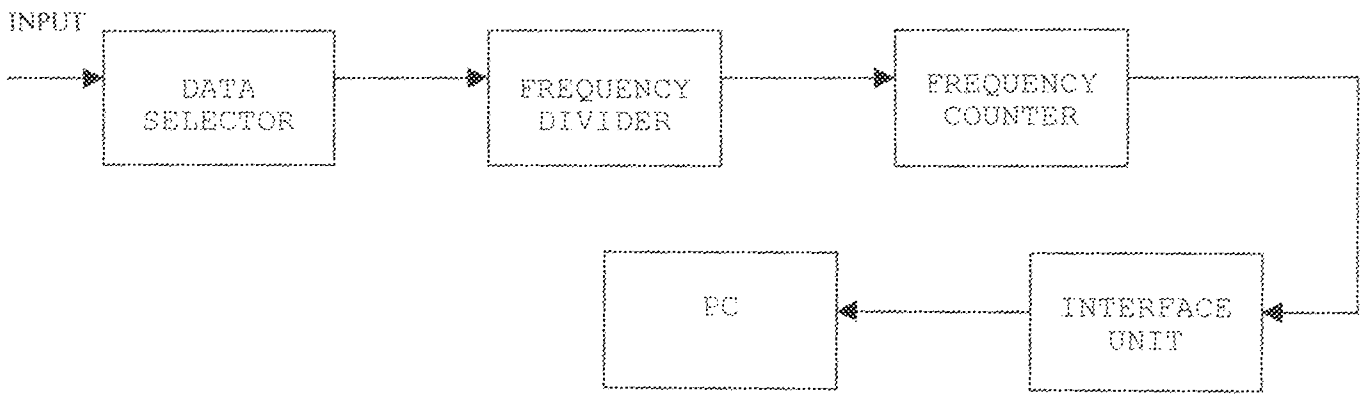


Fig. 1.1 Block Diagram Of The Computer-Aided-Frequency Reader.

### i) DATA SELECTOR

This unit is responsible for picking the signal and based on the range selected by the user, transferring the frequency to the appropriate divider circuit.

### ii) FREQUENCY DIVIDER

This module is charged with dividing the signal frequency from KHz and MHz by an appropriate number to the Hz range.

### iii) FREQUENCY COUNTER

The frequency counter executes the counting of the frequency of the received signal.

### iv) INTERFACE UNIT

This consists of the interfacing integrated chips (IC) and connectors, which link the hardware section of the CAFR to the software section via a PC. This unit serves to make the hardware and software systems sections compatible and work in tandem with each other.

v) PERSONAL COMPUTER

This is used both as an input terminal for frequency range selection and also as an output display unit where the user visually reads the frequency.

## B SOFTWARE SECTION

This is actually the set of instructions which co-ordinate the entire frequency reading operation written in Turbo C++ language. This set of instructions is also called a program. It makes use of object-oriented program, which is a powerful feature of the C++ programming language. The program itself can also be split into three subsections namely: -

i) Input - Output routine

ii) Graphical User Interface

iii) Mouse and Keyboard Routine.

## 1.2 AIMS AND OBJECTIVES

With the advent of the information superhighway the PC has become a basic building block of society in today's age. It is due to this that there has been and continues to be a need for as many user-friendly computer based/aided applications and peripheral devices as possible. The CAFR is an answer to one of the teeming needs. It essentially is a module, which meets the needs of Engineers and Scientists both professional and amateur. The CAFR enables them to accomplish one more task via their PC's. The need to leave the terminals to use an frequency reading instrument located somewhere else has with this design project become obsolete.

The CAFR in its design had the following as part of its aims.

i) Accuracy and Precision: - due to the increased number of significant digits provided by the PC, we can and have achieved better accuracy and high precision in measurement.

ii) Economy: - the lack of 7-segment displays in the design made for reduced cost. The output was displayed on the computer terminal.

iii) Increase in Range: - An increase in the range of the signal frequencies to be measured makes the design more versatile. Larger range is also due to the software programming.

iv) Efficient Power Consumption: - In the peripheral device, the fact that we did away with quite a number of components found in an ordinary digital meter. This reduction in components leads to a proportional decrease in the amount of power consumed. this makes the CAFR a power efficient device.

v) Durability: - Normal digital instruments make use of light emitting diodes (LED's) or Liquid Crystal Displays (LCD's) which are vulnerable to failure. The lack of these things in the CAFR makes it more durable.

## 1.3 LITERATURE REVIEW

Frequency expressed in Hertz is a count of the number of cycles or oscillations per unit time. A frequency of 1 Hertz (Hz) means that there is one cycle or oscillation per second. The unit is named in honor of the German Physicist Heinrich Rudolf Hertz, who first demonstrated the nature of electromagnetic wave propagation, Kilohertz (kHz) thousands of cycles per second, megahertz

(MHz) millions of cycles per second and gigahertz (GHz) billions of cycles per second are employed in describing certain high frequency phenomena, such as radio waves. Radio waves and other types of electromagnetic radiation may be characterized either by their wavelengths (ᵡ) or by their frequencies. The relationship between the frequency and wavelengths of electromagnetic waves is seen in the formula below:

$$f = \frac{1}{\lambda}$$

Where $f$ = frequency

$\lambda$ = wavelength (1)

Measurement and data processing hardware, which refers to the physical means of performing measurements, gathering and processing the data, measured and disseminating the information thus obtained in process control, research, system tests and frequency measurement. This class of hardware may include both individual devices and assemblies incorporating analog, digital and hybrid computing (processing) facilities.

At this time, a new generation of measurement information system has been developed on the basis of the above hardware offering the following advantages: -

i) Wider functional capabilities through reprogramming in the center of measurement data processing.

ii) They acquire, store and reduce measurement data for presentation of tables, charts, models and the like.

In general cases, the computing (processing) hardware used in measurement devices, automates the measurement process from start to finish, including the computation of final results. Microprocessors hold special promise in this respect. At present, microprocessors are used in different applications e.g. graphics recorders, generators, oscilloscopes, medical instruments and other instruments operating as stand-alone or part of measurement information system. Vital components like the interface buses provide a proper tie-in both inside and outside a measurement device or system.

Measurement information systems of the present age are also found in automatic process controls, automated research and testing, Computer-Aided Design (CAD), computer-Aided Manufacturing computer Integrated Manufacturing, etc.

The software used in measurement systems is based on the following principles:-

i) The measurement task on hand is formalized i.e. a list is drawn up of quantities to be measured, the data presentation format is specific and criteria are formulated for measurement data processing or mathematical equations or logic expressions.

ii) An algorithm is developed for the measurement task, designing the sequence of operators for the various functional units of the measurement system according to the scope of nature of the raw measurement data. (2)

For microcomputer measurement system, the keyboard and the mouse serve to select the quantity to be measured; the measurement procedure and the range measurement. Signals keyed in from the keyboard are converted by a code into code words, which go to the data bus. Acting according to the procedure select source, the microprocessor interprets the data coming in from the keyboard and generates control code words e.g. for a multiplexer to select the signal channel.

In recent times, frequency measuring or reading instruments have employed LED's, 7-segment displays or Liquid Crystal Displays LCD's etc which have limitations as regards their power efficiency (1).

## 1.4 PROJECT OUTLINE

Chapter One gives a broad overview of the project. It consists of an introduction and literature review. These deal with basic concepts, earlier research and modular flow of the hardware. The aims and objectives, which were set out as guidelines for design and construction, are also in this chapter.

Chapter Two deals with hardware and software design. The units of the hardware about 10 in number were treated individually and in depth to some extent. The functional routines of the software program three in number were looked into and presented.

Chapter Three: Construction, Testing and results. The hardware and software were constructed or encoded separately. This chapter gives the steps on the process, the tests and checks carried out and the results obtained at each stage.

Chapter Four: Conclusion, Recommendation, Reference and Appendix. This Chapter contains the students' view of what has been achieved and the goals met in the execution of this project. Recommendations for more challenging and technologically advanced projects with greater ease on the part of all concerned are also given. Books and other materials consulted are in the reference. Drawings, pictures, etc. are in the Appendix.

# CHAPTER TWO

## DESIGN AND ANALYSIS

This Chapter primarily deals with the two major sub-sections of this design project viz:- the hardware and software sections. Both subsections are treated on a modular basis for logical flow and ease of comprehension. Each module under both subsections is analyzed and any and all calculations required are contained therein. The values in the calculations are as approximately close to those utilized in the project as possible.

## 2.1 HARDWARE DESIGN

Design of the hardware basically is the design of the equipment involving the function of the CAFR. Hardware consists of components that can be physically handled.

### 2.1.1 POWER SUPPLY UNIT

In Nigeria, the power supplied in the mains is a 240V 50Hz sinusoidal voltage a.k.a. alternating current. Most electronic components and circuits however require low direct current voltages of about 5-15V. In this design therefore, a regulated d.c. supply is incorporated.

A Center-tap 240V rms - 12V-rms step down transformer is used to bring the mains voltage down to 12V. The reduced 12V is however still an a.c. current voltage. To convert it from a.c to d.c it undergoes a process known as RECTIFICATION. This design incorporates a full bridge Silicon Rectifier with rating 600 PRV (peak reverse voltage) 8A is used. Theoretically, the arrangement of four diodes so that the first two conduct during the positive half cycle and the other two during the negative half cycle as shown in fig 2.1 below will yield full wave rectification. Hence, to obtain an output Vo we apply the following formula;

$$V_o = \frac{2 V_s \sqrt{2}}{\pi}$$

Where Vs = supply voltage

= 12 Vrms

Vo = output voltage

$$V_o = \frac{2 \times 12 \times \sqrt{2}}{\pi}$$

$$= \frac{33.941}{\pi}$$

$$= 10.8038 \text{ V}$$

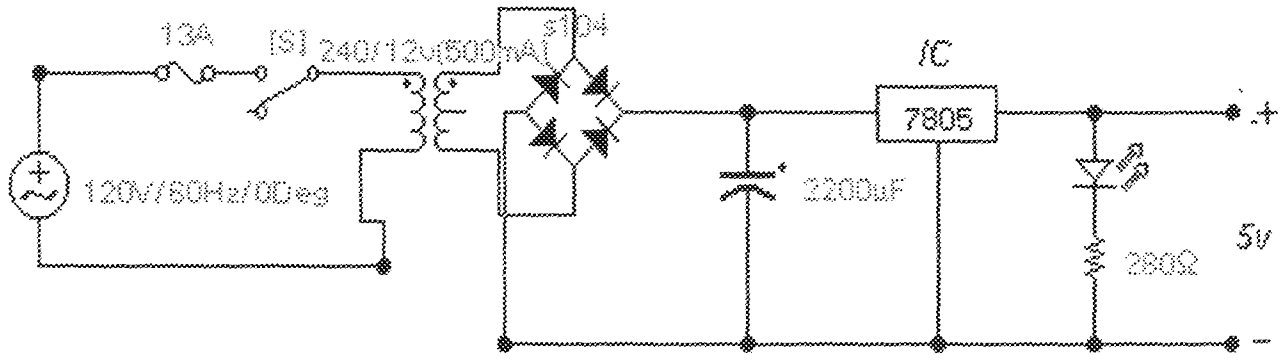


Fig 2.1 Schematic design Diagram of a Regulated Power supply Unit.

The resulting voltage after rectification still has some ripples, which cannot be tolerated by the circuit components.

**FILTRATION** is the process of removing the unwanted ripples and pulses. Filter capacitors are usually incorporated in designs as soothing circuits. These capacitors used in this design have a rating of 2200 uf 25V. This is because the working voltage of the capacitors should be twice that of the rectifier output. The output is an almost ripple-free unregulated d.c power signal.

Most of the components in this design require or recognize a logic level of 5V. The 10.8038V unregulated d.c voltage must be regulated to 5V. An IC-type regulator 7805 is used to maintain constant d.c supply. Since a.c. supply voltages are not constant and the d.c. voltage is directly proportional in Magnitude to the value of the a.c. supply, the d.c. voltage will also vary. This variation cannot be tolerated and hence the need for regulation.

A heat sink is added to diffuse any heat generated by the regulated IC during loading, as they are very heat sensitive. An LED with maximum power rating of 4W is used as a power output indicator from the 7805 regulator.

## 2.1.2 DEMULTIPLEXER UNIT

A demultiplexer takes an input and routes it to one of the several output options based on the input binary address. The other outputs are held inactive or open-circuited. A pull-up resistor is used to assert a well-defined logic level on those outputs.

The demultiplexing unit is used as the data selection module of this design. In the same manner as above, the demultiplexing unit, a CMOS IC-type Quad Bilateral Switch MC 14066 BCP has a 3-bit word as its control signal. This is useful as our range is in Hz, KHz and MHz. The Quad Bilateral Switch has four (4) internal switches and three terminals whilst the third is the control terminal. All the signal pins are active high. The three bit binary word are designated as $X_1$, $X_2$ and $X_3$ where $X_1$, is the least significant bit LSB and $X_3$ is the most significant bit MSB.

Table 2.1 below shows the pins, their designations and the corresponding ranges:

|  | X1 | X2 | X3 |
|---|---|---|---|
| **Hz** | 1 | 0 | 0 |
| **KHz** | 0 | 1 | 0 |
| **MHz** | 0 | 0 | 1 |

From the table above, a signal 100 at the control pins will route the input to the output pin for the Hz range. Signals of 010 and 001 at the control pins will route the corresponding input to the pins for kHz and MHz range respectively. Figure 2.3 below shows the internal structure of the 4066-demultiplexing unit.

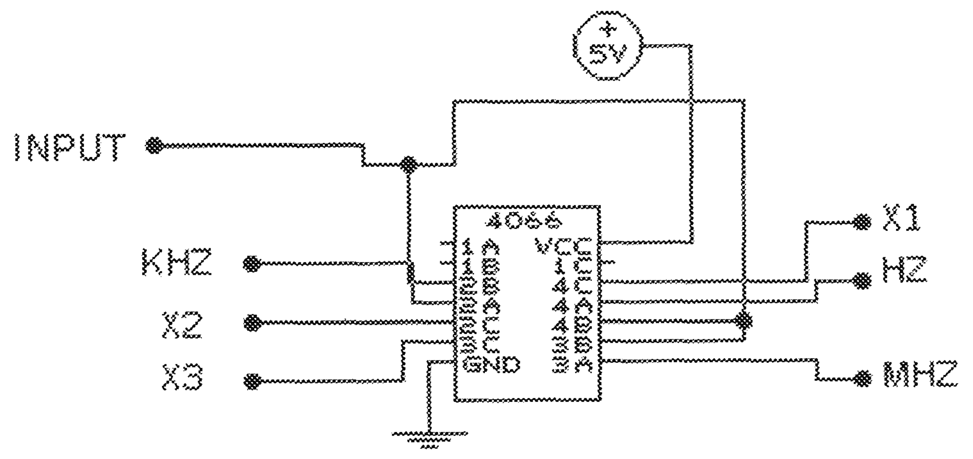


Fig. 2.3 Schematic Design Diagram of the Quad Bilateral Switch

## 2.1.3 FREQUENCY DIVIDER MODULE

The ratio of input frequency to the output frequency of a counter is called its modulo. Most dividers have modulo of integral powers of 2 e.g. 4,8,32,256 etc. For this design, the divider module can be considered in two segments each consisting of a 74LS393 and a 74LS163 each. A segment of the 393 and 163 will divide its input frequency by 1024. This is the closest to 1000, which is required to divide from kHz to Hz. If the frequency is in the megahertz range however, it passes through both segments and is thus divided through 1024 x 1024 times. This effectively converts it from MHz to Hz. Therefore, the output of the frequency divider module is in the Hz range.

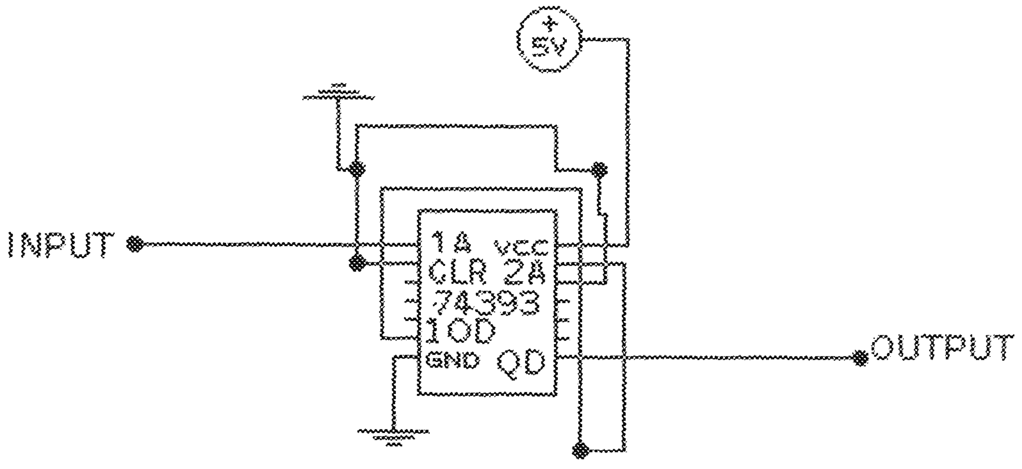


Fig 2.4 Schematic Diagram of the Dual 4-bit Binary counter

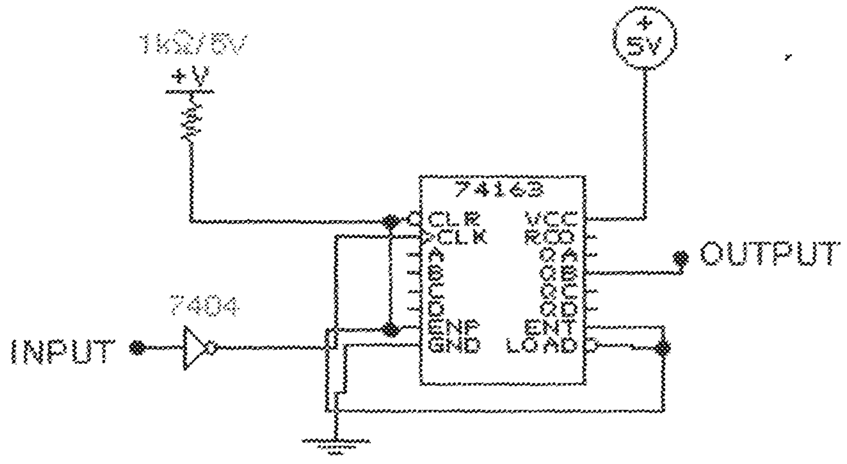


Fig 2.5 Schematic Diagram of the 4-bit Binary counter

The 7404 NOT gate was incorporated to convert the clock input of the 74LS163 from a leading edge triggered to a trailing edge triggered input. The pins that are not used are labeled NC i.e. not connected.

Now the output of 2QD is given by definition.

$$2QD \triangleq \frac{1A}{2^8}$$

$$2QD \triangleq \frac{1A}{256}$$

For the 74LS 163 the signal at output (QB) will be by definition

$$Q_B \triangleq \frac{CLK}{2^2}$$

$$Q_B \triangleq \frac{CLK}{4}$$

When the 74 LS 393 and 74LS 163 are cascaded together, the signal at the QB output will be:

$$Q_B \triangleq \frac{1A}{256 \times 4}$$

$$Q_B \triangleq \frac{1A}{1024}$$

That is any kHz frequency range at the 1A signal input will be divided by 1024.

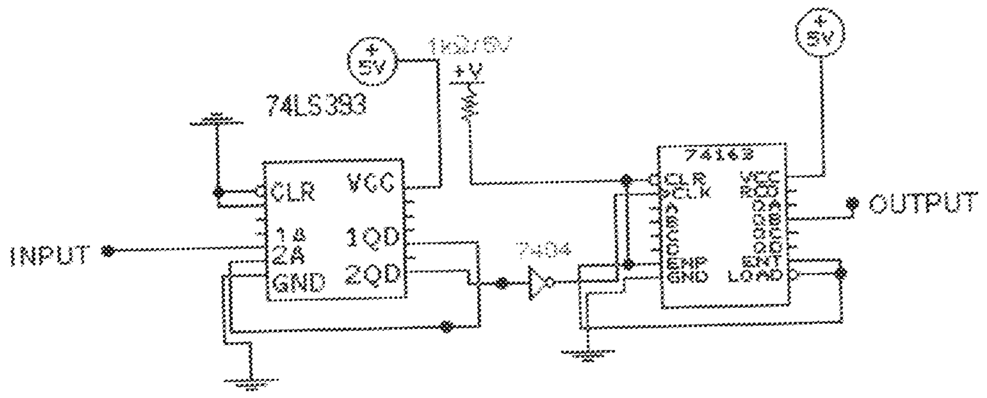


Fig. 2.6. Schematic Design Diagram of the divide by 1024 module.

To read frequency in the MHz range, we incorporate another module of the figure 2.6 in cascade with each other. We will have:

$$Q_8 \triangleq \frac{1A}{1024 \times 1024}$$

$$\triangleq \frac{1A}{1048576}$$

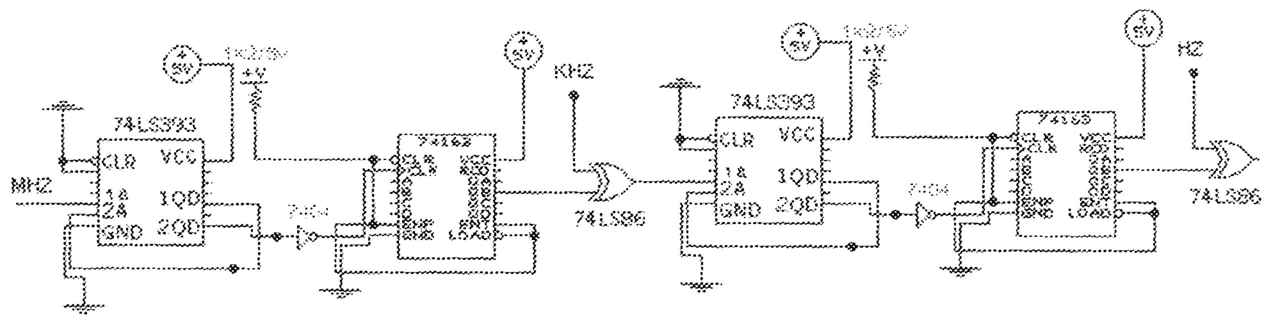


Fig 2.7 Schematic Design Diagram of the divide by 1024 x 1024 Module

## 2.1.4 MONOSTABLE MULTIVIBRATOR

A 555 timer was used to generate a five-second pulse. It gives an output only when triggered by a high to low trigger input. The window pulse was generated by connecting discrete components in form of resistors and capacitors around it to obtain the actual 5-second window. The measurement read will be the number of cycles in 5 seconds to obtain frequency in number of cycles per second, the software is implemented. The formula to obtain values of the surrounding discrete components is as follows:

$$T = 1.1\,RC$$

Where $T = 5$ seconds

5 = 1.1 RC

C = 100 uf

5 = 1.1 (100 x 10.6) R

$$R = \frac{5}{1.1\ (100 \times 10.6)}$$

= 45454.54

= 45.45 k

Due to availability constraints, we had to use 30 k and 1.95k connected in series to give 40.95k. This is however still below the value required and would not give a pulse of 5 seconds. To obtain approximately 5 seconds, a trimmer resistor with value range of 0-10k was incorporated to give a total range of 40.95 - 50.95k. The trimmer was set to give a total resistance value of 45.5k across the 3 resistors.

Figure 2.8 below shows the internal structure and schematic diagram of the NE 555 timer and peripheral components.

Note that RT = 20 % R

= 0.2 x 45.45

= 9.09k = 10k

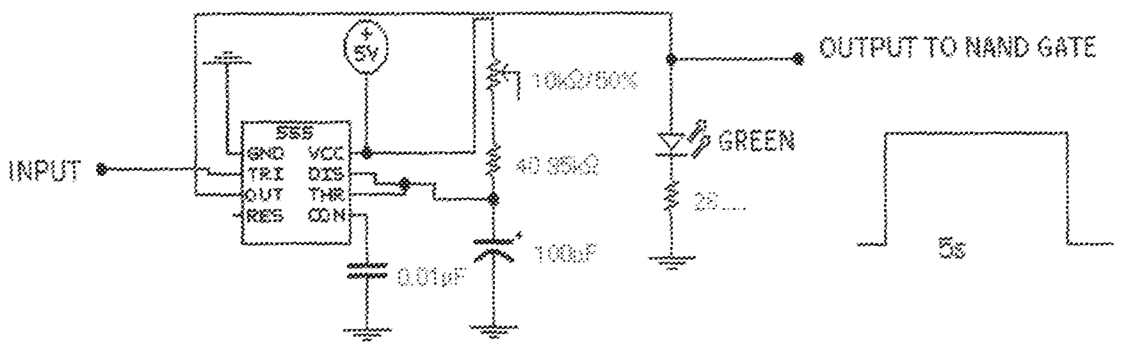


fig 2.8 Monostable Multivibrator

## 2.1.5 8-bit WORD COUNTING UNIT 74LS393

This design for the CAFR employs a transistor logic low power Schottky dual 4-bit binary counter i.e. it uses TTL. To fulfill the design requirements we convert the IC from 4-bit to 8-bit by tapping the output from pin 6 i.e. 1QD signal to pin 13 the second clock input. The clear pins 1CLR and 2CLR are connected to the parallel port through a buffer to be cleared and uncleared by the software program as needed.

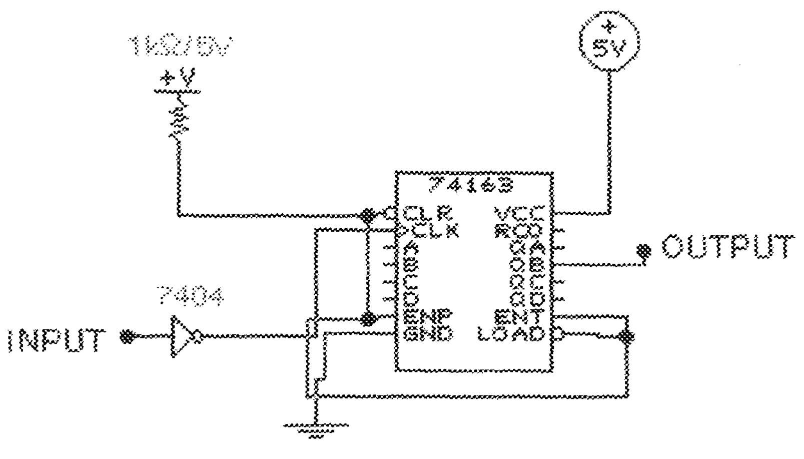


Fig 2.6 Schematic Diagram Of The 8-Bit Word Counting Unit.

## 2.1.6 TRANSPARENT D-LATCH 74LS373

The transparent D-latch is a TTL low power Schottky octal 3-state IC. It serves as the load to the 393. It transfers the input from the counter to the Q output only when the enable G input is high. The G input is strobed by the software. The logic state present in the input just before the enable goes low will be latched to the output. The output control is grounded low since it is active low.

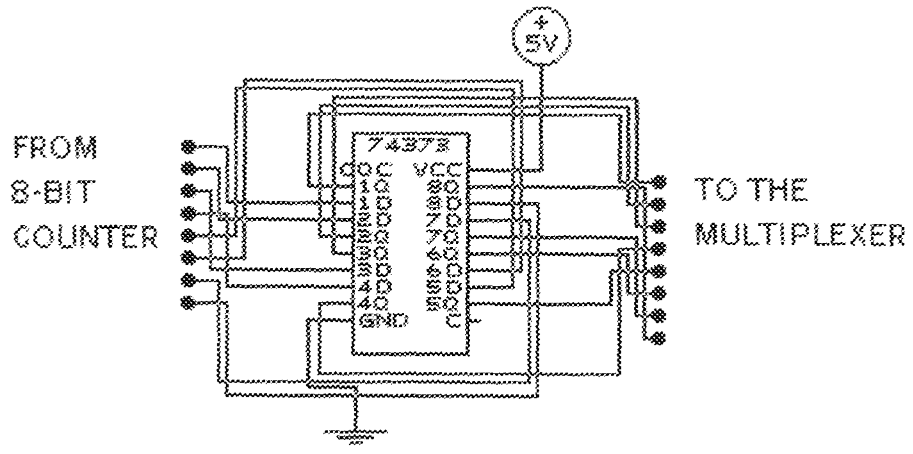


Fig 2.7 A Schematic Diagram Of The D-Latch.

## 2.1.7 MULTIPLEXER UNIT 74LS157

The multiplexer unit is an IC TTL low power Schottky Quad 2 input multiplexer non-inverting. This has an 8-bit input from the latch and a 4-bit output to the buffer. The enable pin G is active low i.e. only when it is low will the input be sent on to the output pins.

Hence, the output pin G is grounded. Since the output is 4-bits we have the input 8-bits, split into A & B i.e. the higher and lower nibbles of 4-bits each. Each nibble can be output by strobing the switch over pin 1 (the A/B signal pin) high and low. A low strobe at the pin gives the signal A with the input A going to the output Y1 as the lower nibble. A high strobe (B) allows the input B to go through and come out as the output Y2, i.e. the higher nibble. A 74LS157 was used due to the fact that the parallel port connector available has an input of 6 pins only. The software will apply an OR operation to merge the lower and higher nibbles to obtain the original 8-bit WORD.

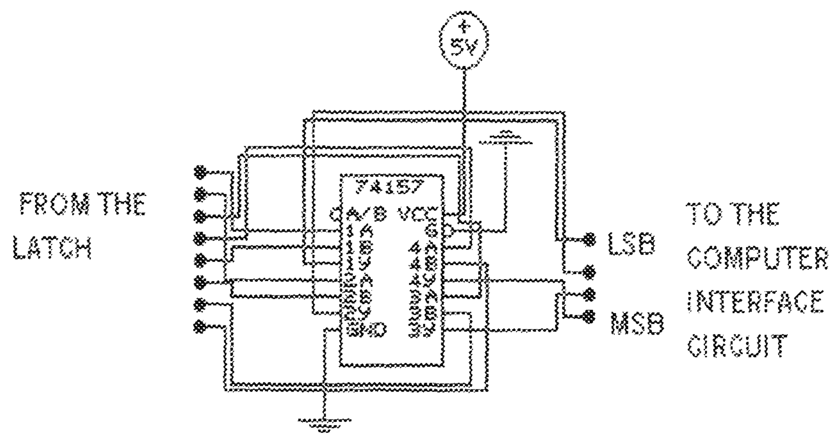


Figure 2.8 Schematic Diagram Of The 74LS157 Multiplexer.

## 2.1.8 BUFFER

A buffer is an intermediate repository of data - a reserved portion of memory in which data is temporarily held pending an opportunity to complete its transfer to or from a storage device or another location in memory. Some devices, such as printers or the adapters supporting them, commonly have their own buffers.

For this design 2 units of 74LS367 were used as buffers. The 74LS367 is an IC-TTL low power Schottky hexadecimal buffer. The buffer acts as a bus driver with a non-inverting 3-state output. The input from the multiplexer is sent to the output, which is the parallel port of the CPU, in exactly the same format in which it was received. It also transfers the data signals to operate different components from the software via the parallel port of the buffer. During both operations it performs two operations namely;

1) SOURCE OPERATION: - the current from the device i.e. the multiplexer is too low, the buffer increases it to a level useable by the CPU i.e., it sources or raises the current during interfacing to an adequate level.

2) SINK OPERATION:- In this case, the current received from the peripheral components may be so high as to be damaging or incompatible with the current to make it compatible with the PC requirements and specifications.

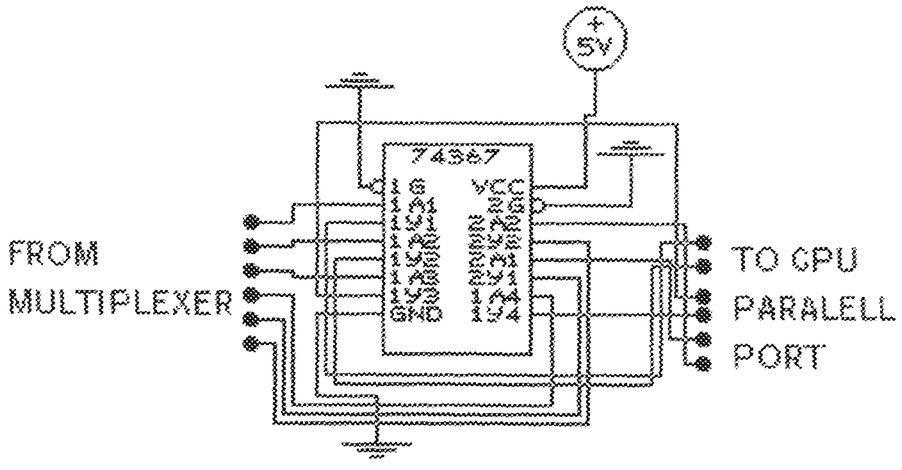


Figure 2.9 A Schematic Diagram Of The 74LS 367 Buffer.

## 2.1.9 LIGHT EMITTING DIODE (LED) DISPLAY

In LED's, a voltage applied to the semiconductor junction results in the emission of light energy. LED's are used in numerical displays such as those on electronic digital watches and pocket calculators.

In this design, the incorporated LED's serve as a manual output indicator from the buffer. 4 LED's were connected to the output pins of the buffer. They basically count he output of the buffer using the first 4 bits as the 4 least significant bits i.e. the lower nibble and the next four as the 4 most significant bits or the higher nibble. It also serves as a trouble-shooting guide as even before the interfacing is carried out, one can determine the output frequency of the hardware section. The 4 LED's in use are taken to ground through individual resistors prevent large current which may damage the LED's.

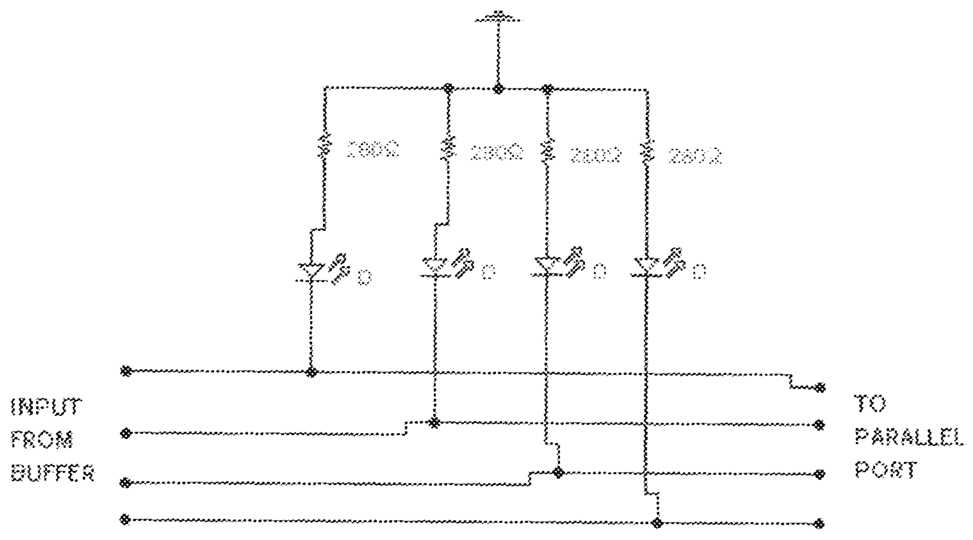


Figure 2.10 LED Display Unit.

### 2.1.10 CPU PARALLEL PORT

To interface between the hardware and the Central Processing Unit we used a parallel port. The CPU recognizes and accesses its various input and output ports using a unique address code. For the parallel printer port the address ranges from 378H - 37FH. It has 25 pins and which can be divided into 3 types i.e. the status, output and control ports. In this design project we use only the output and status in our design.

The output has address of 3BCH or 378H or 278H with decimal equivalent 986,888 or 632. Status port address is 3B DH, 379H or 278H or 957,889 and 633 in decimal. Figure 2 (a), (b) and (c) below illustrate the port addresses and pin numbers and the manner in which signals are arranged in register like format.

PORT ADDRESS

Hex = Dec

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Pin9 | Pin8 | Pin 7 | Pin 6 | Pin 5 | Pin 4 | Pin 3 | Pin 2 |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

3BC = 956

378 = 888

278 = 632

PORT ADDRESS

| Hex | | Dec | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3BD | = | 957 | Pin11 | Pin10 | Pin12 | Pin13 | Pin8 | Not used | | |
| 379 | = | 889 | Busy | Acknowledge | Out of paper | Select | Error | Not used | | |
| 278 | = | 633 | | | | | | | | |

PORT ADDRESS

| Hex | | Dec | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 37A | = | 958 | Not used | | | | Pin17 | Pin16 | Pin14 | Pin 1 |
| 3BE | = | 890 | Not used | | | | Not selected | Initialize | Auto linefeed | Strobe |
| 27A | = | 634 | | | | | | | | |

## 2.1.11 PRINCIPLE OF OPERATION OF THE HARDWARE UNIT

The quad bilateral switch receives the frequency to be measured. The address of the channel to be used is determined by the binary word $X_1$, $X_2$ , $X_3$ which is controlled by the user. Assuming a frequency in the MHz range is applied the address 001 is activated in the $X_1$, $X_2$ , $X_3$ and is passed on and the MHz channel is opened. The frequency is passed onto the first set of frequency dividers made up of one 74LS 393 and one 74 LS 163. The 74LS 393 divides by $2^8$ i.e. 256 and 74LS 163 by $2^2 = 4$. The whole divider section therefore divides it through by 1024. Having been divided by 1024, the frequency is passed to another divider module to be further divided by

1024 into the Hz range. The result is passed through XOR gates at each dividing stage to avoid passing more than one output at a time as this can short circuit the system.

The divided frequency result is passed to the counter via a NAND gate, which also receives the 5-second window pulse triggered by the 555-timer circuit. This opens the NAND gate and the frequency result is allowed through for 5 seconds. The counter receives the frequency and passes it on to the latch and further on to the multiplexer. From the multiplexer, it is passed on to through the buffer to the CPU via the parallel port.

The software program does the conversion and displays the result.

## 2.2 SOFTWARE DESIGN

The design of the software program to drive the hardware can be divided into three different parts. The user interacts with the graphics he sees on the computer monitor using the mouse and keyboard.

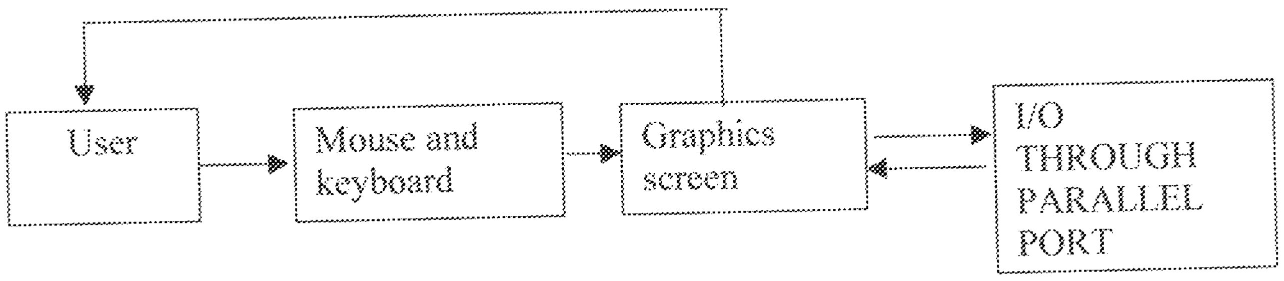


Fig. 2.1.1 Interaction Between The Programming Section And The User.

### 2.2.1 GRAPHICAL USER INTERFACE

For the screen design, we declared a Guiobject with data members topleft, Caption, Border Style, Backcolor, Height Width, Forecolor and State. It has 4 constructors and 1 destructor and has member functions Draw (), within () and Mouse Event (). Guiobject has 3 subclasses; Button, Window and Label to enable object oriented programming which is a feature of C++. Rangewindow, a subclass of window subclass takes care of the clicking of the Hz, KHz and MHz and the Digital and Dial Buttons.

The first screen is a WELCOME screen followed by the MENU and finally the CAFR screen. The screen is seen in picture of the appendix.

## 2.2.2 MOUSE AND KEYBOARD ROUTINE

The first thing to do was declare the mouseclass and the following data members UNION REGS, Mouse Manager Left and Right. It has 1 constructor and 1 destructor. The member functions include Register ( ), hide ( ), show ( ) and Event ( ). REGS UNION had in and out as instances. Ax was used to set function members for interrupt routine bx return as the mouse button status and cx and dx returns the x and y coordinate of the mouse pointer.

## 2.2.3 INPUT AND OUTPUT ROUTINE

Input and Output routine involves the enabling of the IC's, receiving data by use of address of the parallel port. The address, which controls the IC's from the parallel port, is 378H while for the data reception unit the address is 379H. Pins 15, 13, 12 and 10 serve as input pins while 4,5,8,9,3,6 and 7 are control pins.

The graphics and mouse routine was saved as Gui.h. It is included as a preprocesor directive in a good feature of C++ known as information binding. Instances of Window Label, Button and Mouse will be declared before the main function. The graphics function will be declared and initialized before the main function and then called at the main function.

# CHAPTER THREE

## CONSTRUCTION, TESTING AND RESULTS

In the construction of the computer aided frequency reader (CAFR) at each stage, the design specifications of each component were followed strictly, except in cases of non-availability of components. Such cases made slight changes in design specifications inevitable. As the project is software driven one, testing and results revolved both in software and hardware areas of the project.

Hardware construction includes a prototype on the temporary location of the breadboard and then proper construction of the completed design on the vero board. Testing was carried out at both stages and results taken and analyzed to ensure they were within design error range.

Software while not involving construction curtailed a number of debugging and test run sequences before it was found to be error free. The constants, variables, functions etc to be used were specified before programming using turbo $C^{++}$ language.

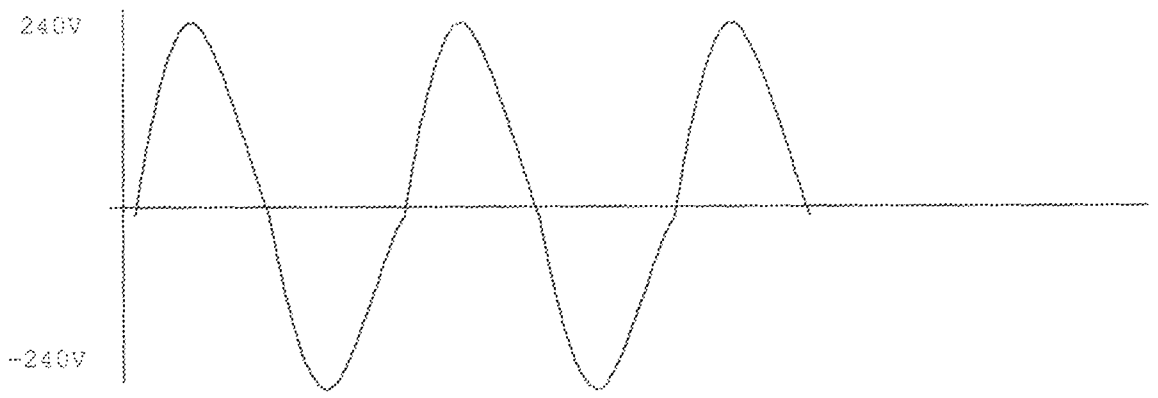
## 3.1 HARDWARE CONSTRUCTION, TESTING AND RESULT

The construction itself is the connection of the different components according to the design was done initially on a breadboard. Testing and result taking was of the main circuit was done on the breadboard and later on the vero board after transfers.

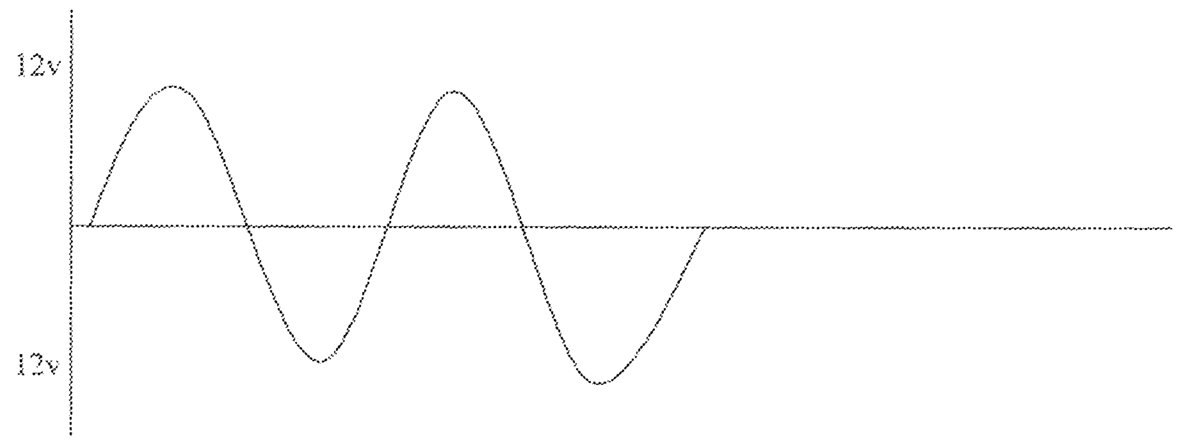
## 3.1.1 POWER SUPPLY UNIT.

The power supply unit consist of a 240by 12V center tapped transformer, 600 PRV 6A rated bridge rectifier, a 1K resistor, LED, fuse holder, ON-OFF switch, mains socket. The transformer primary terminal is soldered to fuse holder and terminal of mains socket. The other terminal of fuse holder and terminal of mains socket. The other terminal of fuse holder soldered to switch terminals and the other terminal to the mains socket. When connected to an ac main supply, the transformer output was measured to be 12V when the yellow and black terminals were tested. When both yellow terminals were tested, the output was 24Volts, which was normal.

The 12V terminal was then soldered to the vero board. The ac-input terminal to the bridge rectifier was soldered so as to make contact with the output of the transformer. The rectifier output was measured at 1.5Vdc with the evidence of some ripples see fig. 3.2(a). To eliminate the ripples, a filtering rectifier 2200µf 25V was soldered to the output of the rectifier connecting the appropriate positive digital multimeter is shown in figure 3.2 (b).



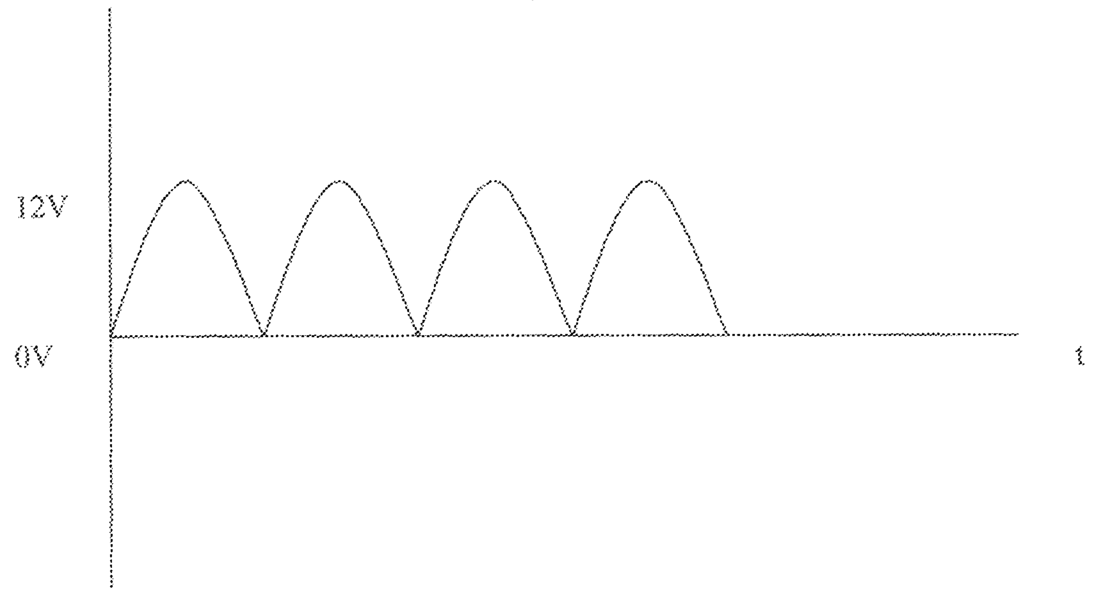
3.1(a) a/c mains input to transformer.



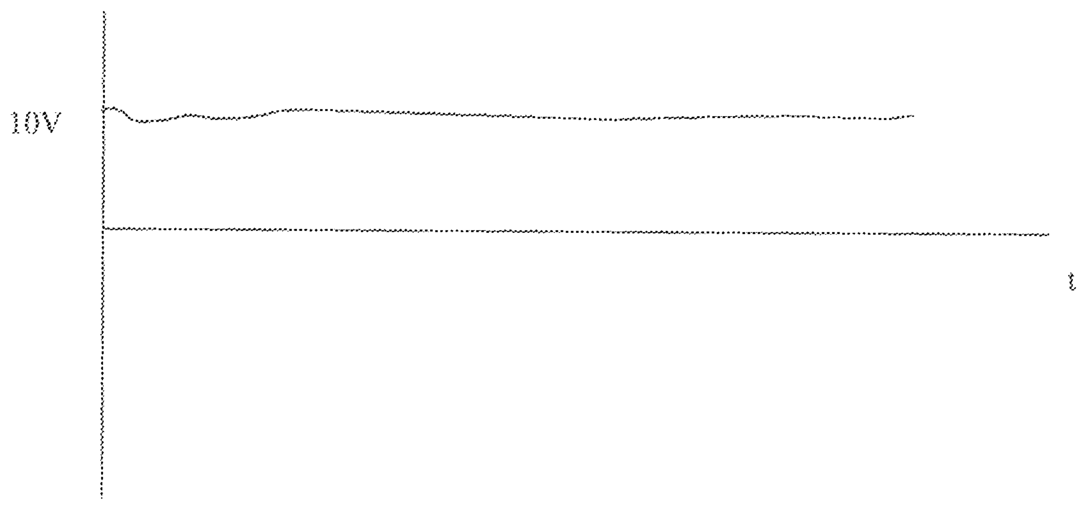
3.1(B) A/C OUTPUT OF TRANSFORMER.

The voltage after rectification was passed to a regulator to achieve the required voltage level recognized by the components as logic level high.

The 7805 regulator was screwed to a heat sink. Both heat sink and regulator terminals were soldered to the vero board. The regulator output was measured at 4.95V, which was within range of the required 5 Volts. An LED was connected to the output to primarily serve as a power on indicator and also for trouble shooting.



3.2(a) full wave bridge rectifier output.



3.2(b) filter capacitor output.

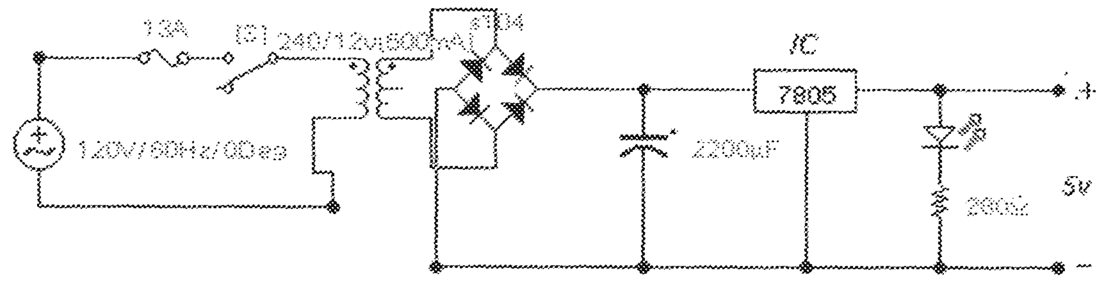


Fig. 3.3 schematic diagram of power supply

### 3.1.2 HARDWARE ON BREADBOARD

Construction included layout and mounting of IC's, discrete components and conducting jumper wires on the breadboard. This is merely a prototype jumper, which allows for alterations, as the connections are not permanent. Three interlocked breadboards were interlocked and used as a single unit. The jumper wires which we employed were used in accordance with color-code i.e. Vcc and positive using RED. Negative or grounded BLACK and white and blue for connecting signal terminals of IC's to one another. Picture in the appendix shows greater detail.

A frequency generator for testing was also mounted on the breadboard for ease of accessibility due to constant use. The frequency generator components are as follows NE 555 timer trimmer resistor, I μF 50V capacitor and a 10k resistor. This gave a frequency range of 5kHz to 15kHz. An LED indicates power output.

Pull-up resistors were constructed by mounting a resistor from Vcc to a conducting slot and pull down from ground to all IC.s to protect them from transient and splices which occur when reading MHz range frequencies.

Inserting 74393 and 74165 along with jumper wires and inserting suppressor capacitors mounted frequency divider modules. The required 5-second pulse window was generated by a 555 timer, which was tested for accuracy using a stopwatch and green LED, which was set to light up during open window intervals. The LED thus gave an approximately 5 second duration light.

The bilateral switch and exclusive OR (XOR) gates were connected to the frequency divider section. The output of the divider and 5 second generating timer output were connected to the NAND gate. The red LED at the NAND gate output indicates frequency division is taking place.

The NAND gate output was connected to the counter 74393 and an LED connected to the Laciest significant Bit (LSB) down through the Most Significant Bit (MSB) for testing and it was found that the ON and OFF states differ for the different signals plus. The multiplexer and buffer were mounted and tested 4 RED LED's were connected at the output of the buffer and then to the parallel post as shown in the schematic. Pulling all active high devices high and vice versa then tested the whole circuit.

## 3.1.3 HARDWARE ON VERO BOARD

Almost like the breadboard but a permanent layout of resistors, capacitors and IC sockets on the insulated side. All soldering was done on the conducting side using transformer wires. The XOR gate, NOT gate, bilateral switch and frequency dividers were soldered to a 9 x 6 cm Vero board while the multiplexer, buffers, latch NAND gate, counter and 555 timer went on a 14 x 6cm Vero board. The protective capacitors were soldered directly to the Vcc and ground terminals of the IC sockets. IC sockets were used to prevent damage of IC`s due to heat during soldering. Maintainability also played a role in use of the sockets. Continuity was checked using a digital multimeter.

A soldering map was drawn for use as an aid to increase speed and also for trouble shooting. The following are some soldering precautions.

i) All excess soldering was removed

ii) All solder bridges (pieces of solder crossing copper traces) formed were broken using the soldering iron.

iii) All connections were made shiny instead of dull

iv) Care was taken to ensure smooth puddles were formed rather than a round ball.

## 3.2 RESULTS

From the frequency generator, the output generated by adjusting the trimmer resistor is 5.26 $KH_z$

$$\text{Output will be} = \frac{5.26}{1024}$$

$$= 5.137\text{Hz}$$

The 5 seconds pulse was triggered and an output of 1011 was read from the LED's. The A/B signal pin of the multiplexer was pulled high to get an output bit pattern of 0001.

The 2 nibbles of 4 bits were then joined together to form a byte of 6 bits 00011011

$0\,0\,0\,1\,1\,0\,1\,1_2 = 0 + 0 + 0 + 2^4 + 2^3 + 0\,2^1 + 2^0$

$= 0 + 0 + 0 + 16 + 8 + 0 + 2 + 1$

$= 27$ cycles /5 sec.

In Hz which is cycles / sec

$$\text{Frequency} = \frac{27}{5}$$

$= 5.4$Hz

To convert to kHz we multiply by 1024

Frequency $= 5.4 \times 1024$

$= 5529.6$

$= 5.5296 \times 10^3$ Hz

$= 5.5296$kHz

Original input $= 5.26$kHz

Output at LED $= 5.5296$

Error $= 5.5296 - 5.26$

$= 0.2696$

$$\text{Percentage error} = \frac{0.2696}{5.26} \times 100$$

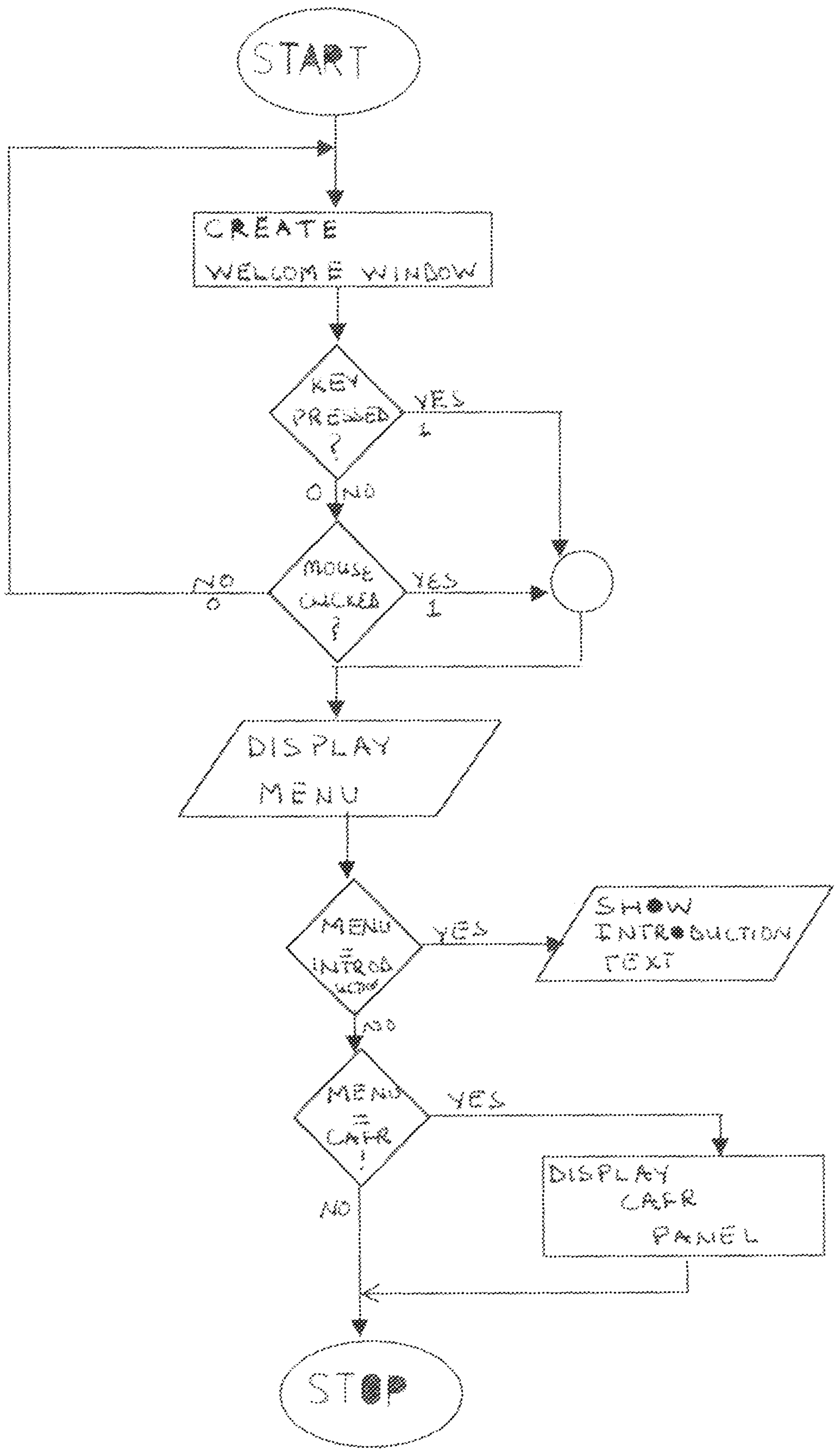
$= 0.0512 \times 100$

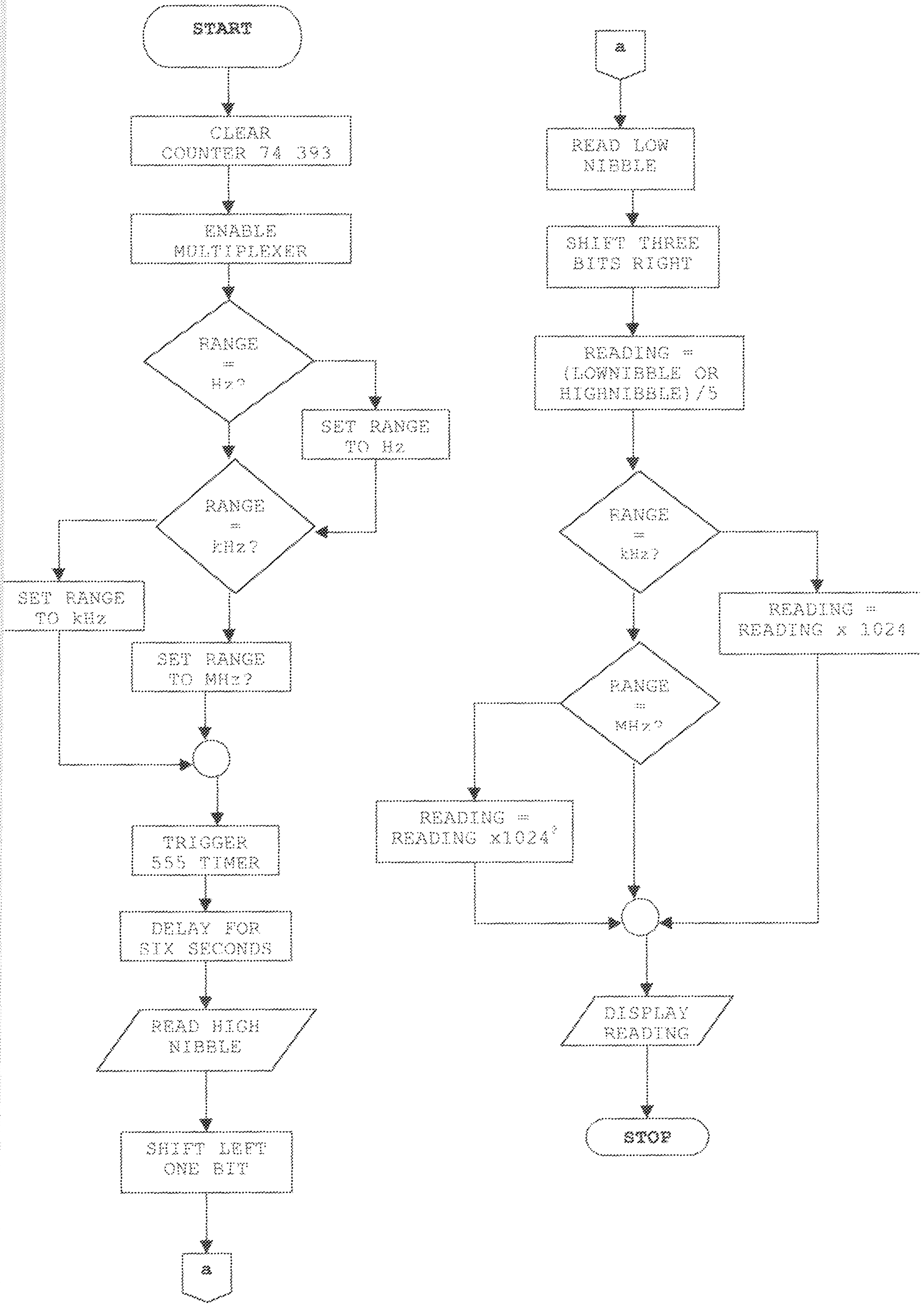
$= 5.12\%$

The above percentage error is still well within the approval limits for experimental error. Similar tests in the Hz and kHz range gave result, which were also satisfactory.

## 3.2 SOFTWARE ENCODING, TESTING AND RESULTS

The flowchart of the graphical user interface (Gui) and the input/output routine is shown figures 3.4 and 3.5 below.




START
CLEAR COUNTER 74 393
ENABLE MULTIPLEXER
RANGE = Hz?
SET RANGE TO Hz
RANGE = kHz?
SET RANGE TO kHz
SET RANGE TO MHz?
TRIGGER 555 TIMER
DELAY FOR SIX SECONDS
READ HIGH NIBBLE
SHIFT LEFT ONE BIT
a
a
READ LOW NIBBLE
SHIFT THREE BITS RIGHT
READING = (LOWNIBBLE OR HIGHNIBBLE)/5
RANGE = kHz?
READING = READING x 1024
RANGE = MHz?
READING = READING x1024²
DISPLAY READING
STOP

The encoding of the programme follows the sequence of the flowcharts in the figures. The programme, which services as t he device driver written in Turbo $C^{H}$ ver.1.00, is written below. It makes use of the object oriculed-programming feature of $C^{++}$

Complying and linking the source code to the object code tested the program. No error was encountered and the numbers of warnings given were minimal (7). Running of the object code yielded zero run-time errors and the output on the computer monitor was O $H_z$ Appendix is a picture of the output on the screen.

## 3.2 HARDWARE – CPU INTERFACING

The male to male parallel D-converter was connected to the parallel post of the CPU and the parallel part of the CAFR. The two systems were powered on and the device driver 04 in the CPU was run to display the screen of the CAFR. The CAFR probe was then connected to the frequency generator. To test it, the DIGITAL and kHz ranges were clicked along with the READ button. On clicking READ, the 4 LED's start blinking and when the counting is done, the screen displays the frequency value read.

In actual test, the digital multimeter measured the generated frequency as 7.55kHz while the CAFR read 7.528 kHz from both LED and PC screen. The margin of error is as can be seen and calculated very low.

## 3.3 CONSTRUCTION OF CASING

The entire circuit system was housed in a wooden casing for protection. The casing was constructed from ½" wood and plywood. Tools used include drilling machine, electronic jigsaw, plane, files measuring tape etc. materials employed include araldite, wood glue (white) putty, sandpaper etc. The plywood makes up the top section while the ½ inch wood was used for the base with dimensions shown in figures 3. The super glue held the plywood in place before applying wood glue. 3mm holes were drilled into the plywood while 5mm holes were drilled into the base of the ½ inch wood.

The casing was perforated at the sides to allow for ventilation thus avoiding damage due to heat as heat is dissipated quickly through the holes.

# CHAPTER FOUR

## CONCLUSION

The design and construction of a Computer-Aided Frequency Reader sounds and may to many people seem an easy task. This however as we found was not the case. The preconceived notion that technological advancement, availability of components etc would ease the execution was dashed when the project commenced. This was due to lack of nominal values of the components required by design specification.

The retailers of these components also were found in most cases to have little or no knowledge of the nominal values of these components if and when they are available. Thus the problem of identifying components which could be used in their stead lead to delay and design modifications.

As students, we also faced the inevitable crunch of financial constraint but as engineers in the making, we applied our ingenuity and made "judicious use of what we h ad to get what we wanted".

Therefore, with a reasonable degree of economy and efficiency, the aims outlined earlier were achieved and the project dream is now a reality. The frequency can be read on the computer terminal of branded system such as the IBM, DELL. COMPAQ and their compatibles.

## 4.1 RECOMMENDATION

Due to high cost of components, and therefore projects, a lot of students were limited in topics to work on. It is recommended that the organized private sector and other interested parties take active parts in finding final year projects like this.

## 4.2 REFERENCES

1. MICROSOFT INC. 1997 ENCARTA ENCYCLOPEDIA MICROSOFT INC. USA VOL.2

2. MC GRAW HILL 1982 ENCYCLOPEDIA OF SCIENCE & TECH. ACADEMIC PRESS USA

3. BELONE SCHILLING 1979 ELECTRONIC CIRCUITS DISCRETE & INTEGRATED MC GRAW HILL USA

4. BRAMER BRAIN AND SUSAN 1995 C$^{++}$ FOR ENGINEERS ARNOLD PUBLISHERS U.K

5. DUSHIME BASIC METEOROLOGY & ELECTRICAL MEASURE-MENT/ MIT PUBLISHERS USSR

6. GREGORY B.A. 1981 AN INTRODUCTION TO ELECTRICAL INSTRU-MENTATION AND MEASUREMENT SYSTEMS. MACMILLAN U.K

7. HALL DOUGLAS V. 1983 MICROPROCESSORS & DIGITAL SYSTEM. MC GRAW HILL U.S.A

8. THERAJA BL & AK 1990 A TEXTBOOK OF ELECTRICAL TECHNOLOGY. INDIA

# APPENDIX 1

Below is the device driver written in C++:

```
#include "gui1.h"
#include <graphics.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include<stdlib.h>
#include <iostream.h>
#include <conio.h>
enum{hz,khz,mhz};
int p378,p379;
int SelectedRange=hz;
double Reading;
char ReadS[20];
int TimeToGo=0;
unsigned char lownibble,highnibble; int l,h;
Window * WindowsManager, *Display,*ProgressBar,*Range,*Range1;
Label *label1,*label2,*TitleBar;
Button *Hz,*Khz,*Mhz,*Dial,*Digital,*Read,*Power,*Quit,*Help;
mouse *Mouse;
Window* Menu;
Button* CAFR;
Button* Introduction;
Button* QuitButton;

class RangeWindow:public Window{
        public:
        RangeWindow (int x,int y,int h,int w):Window (x,y,h,w) {}
        void virtual MouseEvent(int Event,int X,int Y)
        {
        int handled=0;
        if (Within(X,Y)){
                Collection Control;
                Control=Controls;
                while (Control.Current!=NULL){
                        if (Control.Current->Within(X,Y)){
           Control.Current->MouseEvent(CLICK,X,Y);
                                Control.Current->State=DOWN;Control.Current->Draw();
                                handled=1; Focus=Control.Current;
           if (Control.Next!=NULL) Control=*Control.Next;
                                else Control.Current=NULL;
                        }
                        else {
                                Control.Current->State=UP;Control.Current->Draw();
                                if (Control.Next!=NULL) Control=*Control.Next;
                                else Control.Current=NULL;
                        }
                }
                if (handled){
                        switch (Event){
                        case (DOWN):State=DOWN;Draw();break;
                        case (CLICK):State=UP;Draw();break;}
                }
        }
        }
};
```

```
void init(void)
{
 WindowsManager=new Window(0,0,479,639);WindowsManager->Draw();
 Display=new Window(15,60,309,609);Display->BorderStyle=single;Display-
>BackColor=GREEN;Display->Draw();
 ProgressBar=new Window(15,373,15,609);ProgressBar->BackColor=YELLOW;ProgressBar-
>Draw();
 TitleBar=new Label(3,3,18,633,"Frequency Reader");TitleBar->BackColor=BLUE;TitleBar-
>ForeColor=WHITE;TitleBar->BorderStyle=threeD;TitleBar->Draw();

 Quit=new Button(3,25,18,311,"Quit");Quit->BackColor=BROWN;Quit-
>Draw();WindowsManager->AddControl(Quit);
 Help=new Button(315,25,18,320,"Help");Help->BackColor=BROWN;Help-
>Draw();WindowsManager->AddControl(Help);

 label1=new Label(15,415,15,100,"Freq. Range ");label1->Draw();
 label2=new Label(15,445,15,100,"Display Mode");label2->Draw();

 Range=new RangeWindow(120,410,25,500);Range->BorderStyle=single;Range-
>Draw();WindowsManager->AddControl(Range);
 Hz=new Button(123,413,18,165,"Hz");Hz->Draw();Range->AddControl(Hz);
 Khz=new Button(288,413,18,165,"Khz");Khz->Draw();Range->AddControl(Khz);
 Mhz=new Button(453,413,18,165,"Mhz");Mhz->Draw();Range->AddControl(Mhz);

 Range1=new RangeWindow(120,440,25,250);Range1->BorderStyle=single;Range1-
>Draw();WindowsManager->AddControl(Range1);
 Digital=new Button(123,443,18,123,"Digital");Digital->Draw();Range1->AddControl(Digital);
 Dial=new Button(246,443,18,120,"Dial");Dial->Draw();Range1->AddControl(Dial);

 Power=new Button(505,440,25,116,"Power");Power->ForeColor=RED;WindowsManager-
>AddControl(Power);Power->Draw();
 Read=new Button(375,440,25,129,"Read");WindowsManager->AddControl(Read);Read-
>Draw();

 Mouse=new mouse;
 Mouse->Register(WindowsManager);
}
void ReadClick(void)
{       p378=8;outport(0x378,p378);//clear counter '393
        p378=0;outport(0x378,p378);
        p378=16;outport(0x378,p378);//enable mux '373

        //select range
        switch(SelectedRange){
        case (hz):p378=0x50;break;
        case(khz):p378=0x90;break;
        case(mhz):p378=0x18;break;
        }
        outport(0x378,p378);

        //trigger 555 timer
        p378=p378|4;outport(0x378,p378);
        p378=p378&251;outport(0x378,p378);

        delay(7000);

        lownibble=inport(0x379);i=lownibble;
        lowuibble=lownibble & 0x78;      l=lownibble;
        lownibble>>=3;          l=lownibble;

        p378=p378|0x20;outport(0x378,p378);
        highnibble=inport(0x379);h=highnibble,
        highnibble&=0x78:       h=highnibble;
        highnibble<<=1;         h=highnibble;

        Reading=lownibble|highnibble;

        Reading/=5;
        switch(SelectedRange){
        case(khz):Reading=Reading*1024;break;
        case(mhz):Reading=Reading*1048576;break;
        }
  Display->Draw();
  //printf("\nReaDING=%d",Reading);

oid DrawIntro(Window* Disp)

        setcolor(RED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
        outtextxy(Disp->TopLeft.x+6,Disp->TopLeft.y+50,"WELCOME TO THE
OMPUTER");
        outtextxy(Disp->TopLeft.x+6,Disp->TopLeft.y+110,"AIDED FREQUENCY
EADER");
        outtextxy(Disp->TopLeft.x+240,Disp->TopLeft.y+160,"(CAFR)");
        settextstyle(SANS_SERIF_FONT,HORIZ_DIR,3);
        outtextxy(Disp->TopLeft.x+14,Disp->TopLeft.y+230,"Range 0 to 255 Hz");
        outtextxy(Disp->TopLeft.x+14,Disp->TopLeft.y+260,"Range 0 to 255 KHz");
        outtextxy(Disp->TopLeft.x+14,Disp->TopLeft.y+290,"Range 0 to 255 MHz");
        outtextxy(Disp->TopLeft.x+14,Disp->TopLeft.y+450,"CopyRight Yanni and Emebel");
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
}
void DrawDisplay(Window* Disp)
{
        setcolor(RED);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,10);
  gcvt(Reading,5,ReadS);
  strcat(ReadS,"Hz");
        outtextxy(Disp->TopLeft.x+80,Disp->TopLeft.y+100,ReadS);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

}
void HelpClick(void)
{
        delete Help;
}
void HzClick(void) {SelectedRange=hz;}  //all the click functions are Event handlers
void KhzClick(void) {SelectedRange=khz;}
void MhzClick(void) {SelectedRange=mhz;}
void QuitClick(void) {TimeToGo=1;}
void IntroClick(void) {TimeToGo=1;}
void CAFRClick(void){
        delete Mouse;
        delete QuitButton;
        delete Introduction;
        delete CAFR;
        delete Menu;
        init();
        Display->SetPaintProc(DrawDisplay);
        Help->SetClickFunction(HelpClick);
        Hz->SetClickFunction(HzClick);
        Khz->SetClickFunction(KhzClick);
        Mhz->SetClickFunction(MhzClick);
        Read->SetClickFunction(ReadClick);
        Quit->SetClickFunction(QuitClick);
        do Mouse->Event();while(!TimeToGo);
}
void IntroductionClick(void)
{}

main ()
{  int graphmode,graphdriver=DETECT;
         initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
         Window* Intro=new Window(0,0,479,639);
         Intro->SetPaintProc(DrawIntro);
         Intro->Draw();
    Mouse=new mouse;
         Mouse->Register(Intro);
         Button* IntroButton=new Button(410,460,15,220,"Click here to continue");Intro-
>AddControl(IntroButton);IntroButton->Draw();
         IntroButton->SetClickFunction(IntroClick);
         do Mouse->Event();while(!TimeToGo);
         delete Mouse;
         delete IntroButton;
         delete Intro;
         TimeToGo=0;

         Menu=new Window(0,0,479,639);
         Menu->Draw();
         Mouse=new mouse;
         Mouse->Register(Menu);
         CAFR=new Button(210,150,15,220,"CAFR");Menu->AddControl(CAFR);CAFR-
 >Draw();
         CAFR->SetClickFunction(CAFRClick);
         Introduction=new Button(210,300,15,220,"Introduction");Menu-
 >AddControl(Introduction);Introduction->Draw();
         Introduction->SetClickFunction(IntroductionClick);
         QuitButton=new Button(210,450,15,220,"Quit");Menu-
 >AddControl(QuitButton);QuitButton->Draw();
         QuitButton->SetClickFunction(QuitClick);
         do Mouse->Event();while(!TimeToGo);

         TimeToGo=0;


         closegraph();
```