MICROPROCESSOR BASED MULTIFUNCTION SIGNAL GENERATOR

By

KARIMI P. MWANGI

A THESIS SUBMITTED IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF SCIENCE

KENYATTA UNIVERSITY

OCTOBER, 2001

DECLARATION

This Thesis is my original work and has not been presented for the award of a degree at any other University.

Karimi P. Mwangi
(I56/8441/98)
Department of Physics
Kenyatta University
P. O. Box 43844
Nairobi, KENYA

Signature _____ Date 9/11/2001

This Thesis has been submitted with our approval as University Supervisors.

Dr. G. A. Ibitola
Department of Physics
Kenyatta University
P. O. Box 43844
Nairobi, KENYA

Signature _____ Date 9-11-2001

Dr. R. L. Stangl
Department of Physics
Kenyatta University
P. O. Box 43844
Nairobi, KENYA

Signature _____ Date 9-11-2001

Dr. S. Namuye
Department of Computer Science
Kenyatta University
P. O. Box 43844
Nairobi, KENYA

Signature _____ Date 9/11/2001

DEDICATION

This work is dedicated to my parents

Karimi Munu and **Beatrice Nyambura**.

Thanks for moulding me to be what I am today.

ACKNOWLEDGEMENTS

Let me take this opportunity to specifically thank all my lecturers in the department of physics, especially my supervisors Dr. G.A. Ibitola, Dr. R.L. Stangl and Dr. Sylvester Namuye, who guided me all through this research. I am also grateful to Kenyatta University for providing me with a full scholarship, which financed my studies.

Special thanks also go to Dr. Gakuru and the entire Department of Electrical Electronic Engineering (Nairobi University) staff for giving me consent to use their 8086 microprocessor kit, EPROM programmer and all other facilities in their microprocessor laboratory without interference.

My colleagues Owade, Wamwangi, Hashim were always there to encourage me and provided peer review of the work, which helped in improving it. There are some special people who constantly provided me with a lot of moral and material support. They include my siblings Njagi, Kinyua and Muriithi, my friends Kamau, Wainaina and Sophi. I also would like to express gratitude to my dad and mum for their moral and financial support.

TABLE OF CONTENTS **PAGE**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

Waveform generators are excellent tools to recreate real-world signals that offer precise simulations for innumerable challenging test conditions. Waveforms are, for example used in airbag deployment, multi-tone amplifier testing, brake simulation, digital modulation, cardiac device calibration, power disturbance harmonic generation, laser beam control and wafer cleaning drive signals among many others. The possible waveforms that can be been generated include: Sine, Cosine, Triangle, Ramp, Exponential, Gaussian, Pulse, Sinc, Hamming (Sinc), Amplitude modulation (AM), Frequency modulation (FM), Frequency shift keying (FSK), Phase shift keying (PSK), Noise-Digital, Noise-Analogue, Comb, Steps, Sweep, Cardiac and Squine.

In this research work, the generation of square, sine, triangular, and random waveforms using a microprocessor-based programmable waveform generator is presented. The 8088 microprocessor and other support chips, hardware and software designs are discussed. Analysis and experimental results based upon the system and its associated waveforms are evaluated using a PM 3384 autoranging combiscope (CRO). The function generator is a prototype of general-purpose, high performance instrument capable of generating standard waveform patterns at frequencies of exceptionally low distortion. The generator is capable of generating frequencies, selectable from 0 Hz to 100Hz and adjustable output amplitude from ± 4.8 V to ± 14.1 V peak to peak with high resolution. The system as a whole possesses some interesting advantages over the analogue and digital types of signal generators that are presently in use.

CHAPTER 1

INTRODUCTION

1.1 A General Overview

This thesis is about the design, construction and testing/ calibration of a microprocessor-based multifunction signal generator (MBM signal generator). There are already in existence today analogue and digital signal generators, which however, have some undesirable limitations. The MBM signal generator is an improved version of the IC 555 timer- based digital signal generator and has several advantages over the currently used analogue and digital signal generators.

Waveform/function generators are excellent tools to recreate real signals and offer precise simulations for innumerable challenging test conditions. They are used in six different major categories [1]:

- Acoustic and audio
- Medical
- Automotive
- Power
- Communication
- Semiconductor applications

In acoustic and audio, function generators are used to test for inter-modulation distortion, audiological deficiency, sonar simulation, geophysical contour analysis,

bearing failure assessment, amplifier linearity testing and multi-tone response evaluation.

In the medical field generators offer diverse applications such as threshold testing and monitor evaluation, patient display monitor testing and pacemaker certification among others.

In automotive systems, function generators are used for airbag deployment, electronic security orders, ignition timing signals, suspension actuation and brake simulation.

The use of function generators in power includes simulating power disturbances to activate distribution relays or circuit breakers (harmonic generation), precision phase angle control, utility meter calibration and uninterruptible power supply testing.

In communication, applications include modem testing, radio data simulation, modulator phase sensitivity measurements, pseudorandom code patterns and intermodulation distortions.

Waveform variety punctuates scientific areas of applications. Uses range from ultrasonic wafer cleaning, semiconductor device evaluation, laser beam control, to fluid flow metering.

There are already in existence a wide variety of models of analogue and digital signal generators that are capable of generating the standard waveforms. Analogue signal generators have been observed to be of high cost, low quality, lack of memory save-recall facility, low noise immunity and low signal resolution [2, 3, 4]. On the other hand

digital signal generators lack control programs and are characterized by poor responses in terms of settling times, phase shifts, amplitude and frequency [5, 6, 7]. As a microprocessor is at the heart of the MBM signal generator to be discussed in this work a brief introduction into microprocessors is given in the following section.

1.2 Microprocessors

The microprocessor is a digital device able to receive information in the digital form, process this information according to a stored program and output information in the form of digital signals [8]. It acts as a control centre for all operations and executes instructions that are contained in the memory subsystem. The basic operations of the microprocessor include:

- the transfer of data between itself and the memory section
- the manipulation of data in the memory subsystem
- the transfer of data between itself and input/output devices

In order to facilitate the transfer of data and instructions between the microprocessor, memory and input/output devices, the following major buses (groups of lines or pins sharing an information) are used:

- Address bus
- Data bus, and
- Control bus

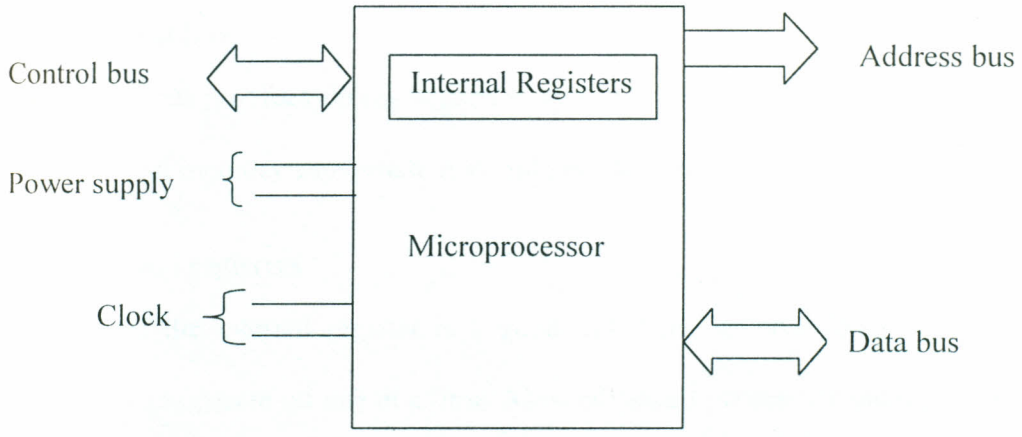


Fig. 1.1: The basic input- output signals associated with a microprocessor.

1.2.1 Data bus

Any transmission medium that has more than one outlet at each end is called a bus The processor data bus is a bundle of wires (or pins) used to send and receive digital signals. The more signals that can be sent at the same time, the more data can be transmitted in a specified interval and, therefore, the faster the bus.

1.2.2 Address bus

The address bus is the set of wires that carry the addressing information used to identify the memory location to which the data is being sent, or from which data is being retrieved. As with the data bus, each wire in an address bus carries a single bit of information. This single bit is a binary digit in the address. The more the wires (digits) used for these addresses, the greater the total number of address locations that is reliased. The size (or width) of the address bus indicates the maximum amount of memory locations that a microprocessor can address. These number of address locations which can be accessed by the processor is known as its physical address space and is often expressed in terms of Kilobytes, Megabytes or Gigabytes.

1.2.3 Control bus

The control bus provides timing signals to synchronise the flow of data between the processor and memory subsystem or peripheral devices.

1.2.4 Internal registers

The size of the internal register is a good indication of how much information the processor can operate on one at a time. Most advanced processors today, from the 386 to the Pentium use 32-bit internal registers and 32- 64 bit data buses. Some processors have an internal data bus (made up of data paths and of storage units called registers) that is different from the external data bus. The 8088 and 386SX are examples of this structure. Each chip has an internal data bus twice the width of the external bus. These designs are called hybrid designs. The 386SX, for example, can pass data around internally with a full 32-bit register size. For communications with the outside world, however, the chip is restricted to a 16-bit-wide data path. Internal registers often are larger than the data bus, and thus the chip requires two cycles to fill a register before the register can be operated on. The Pentium is an example of the opposite situation.

1.2.5 Clock System

A computer system's clock speed is measured as a frequency, usually expressed as a number of cycles per second. A crystal oscillator controls clock speeds, using a silver quartz in a small tin container. As voltage is applied to the quartz, it begins to vibrate (oscillate) at frequency rate dictated by the shape and size of the crystal [9]. The oscillations emanate from the crystal in the form of a current that alternates at the frequency of the crystal. This alternating current is the clock signal. A typical computer

system runs at millions of these cycles per second, so speed is measured in megahertz (MHz) where one hertz is equal to one cycle per second.

In building a processor, a manufacturer tests it at different speeds, temperatures, and pressures. After the processor is tested, it receives a stamp indicating the maximum safe speed at which the unit will operate under wide variations of temperatures and pressures encountered in normal operations.

The size of the data and the address buses, the types of control signals in the control bus, the power supply and clock requirements vary from one microprocessor to another microprocessor.

1.2.6 Microprocessor Applications

Microprocessors are applied in three principal ranges of operation: Control, calculation and administration [8, 9]. Many applications that previously employed hardwired logic have been made feasible by use of microprocessors. Specific applications range from the retail shop price machines to sophisticated equipment like oscilloscopes, frequency synthesizers and TV games. In the process industry, microprocessors are now used for implementing a number of process-control algorithms as economic alternatives to conventional analogue controllers. They have the advantage of being programmable and hence more flexible. In the automobile industry a few recent car models include microprocessors for ignition timing and carburetor control to improve efficiency [10, 11]. Some microprocessors are more suited for particular applications. Midrange devices such as Intel 8080, 8085 or Motorola 6800 are suitable for low cost process control applications [12, 13]. On the other hand, 16, 32 and 64-bit high-performance

microprocessors ranging from 8086/8088 to Pentium III/IV processors (running at 1.5 GHz or more) are more suited to complex signal-processing applications [14].

The final level of microprocessor usage is the multiprocessor system where several dedicated microprocessors are teamed to monitor and control a large industrial plant. For example, the continuous casting of steel slabs uses a complex control system for maintaining temperature, valve pressure flow rate within prescribed limits, at successive stages, utilizing a number of microprocessors [15, 16].

In this research work, 8088 microprocessor is used for optimum control of the generation of the standard waveforms and other pre-stored waveforms.

1.2.7 8088 Microprocessor

The 8088 combines the powerful resources of a 16-bit microprocessor internal architecture with an easy to use 8-bit bus interface. The 16-bit internal architecture provides 16-bit wide registers, and allows the use of 16-bit instructions identical to the ones found in the 16- bit 8086 microprocessor. The 20-bit address bus of the 8088-microprocessor allows it to access 1Megabyte of memory as shown in table 1.1 below.

Table 1.1: 8086/8088 Intel Processor Specifications

Processor	Std. Voltage	Internal Register Size	Data-Bus Width	Address-Bus Width	Maximum Memory	Integral Cache	Integral Math Co-processor
8088	5v	16-bit	8-bit	20-bit	1M	None	None
8086	5v	16-bit	16-bit	20-bit	1M	None	None

The data and address buses are independent, and chip designers can use whatever size they want for each. The sizes of the buses provide important information about a chip's relative power, measured in two important ways: the size of the data bus is an indication of the information-moving capability of the chip, and the size of the address bus tells about how much memory the microprocessor can address.

The 8088 microprocessor has been used in this research project for the following reasons [17]:

- The 8088 microprocessor is 100% compatible with the 16-bit 8086 CPU, i.e. all the power of the 16-bit instruction set is available in the 8-bit 8088.
- The 8088 has a separate bus interface unit called the bus interface unit (BIU) whose job is to fetch instructions from memory and pass data to and from the execution hardware to the outside world over the bus interface. Since the execution unit seldom needs to wait for the BIU to fetch the next instruction, there is less need for the BIU to fetch data quickly. Thus, the 8088 BIU allows maximum performance and processing power without high-speed memory devices in the system.
- The 5 MHz 8088 microprocessor can run at full speed using readily available Erasable Programmable Read Only Memories (EPROM's) with access time of 450ns whereas its counterparts, the 68B09 and Z80B, require wait states in their machine cycles to do the same.

The microprocessor is expected to process and output the data signals in digital forms. The format and shape of the output signal from the 8088 microprocessor based multifunction (MBM) signal generator will be governed by digital codes stored in the EPROM memory units. The following microprocessor related parameters shall be taken into consideration when using the 8088 MBM signal generator for optimal control of the generator i.e. time constants, sampling rates, resolution and accuracy [18, 19].

1.3 Computer Programming Languages

To run a program, a microprocessor must have the program codes stored in binary form in successive memory locations. There are three language levels that can be used to write a program for a microprocessor: machine language, assembly language and high-level language [20].

1.3.1 Machine Language

This is a program written as sequence of the binary codes for the instructions that a microcomputer is to execute. Small monitor programs allow the entry of these instructions and data via hex keypads or terminals by the programmer. The disadvantage of this method is that programs are hand-coded, which is a slow process and also leads to translation errors e.g., the programmer could easily misread 8 as the value B.

1.3.2 Assembly Language

Assembly language programming is done by writing machine instructions in mnemonic form, using an assembler program to convert these mnemonics into actual processor instruction codes and associated data. Mnemonics are symbols or the shortened form of the English words for the operations to be performed by instructions.

Mnemonics are used because they: -

- are more meaningful than hex or binary values.
- reduce the chances of making an error.
- are easier to remember than bit values.

Assembly language statements are usually written in a standard form that has four fields: Label field, Opcode field, Operand field and Comment field, e.g.

Label field	**Opcode field**	**Operand field**	**Comment field**
START:	MOV	AX, 00H	; clears the ax register

A label is a symbol or group of symbols used to represent an address which is not specifically known at the time the statement is written. A colon usually follows labels. Here, the label START is equal to the address of the instruction MOV AX ,00H.

The opcode field (operation code) of the instruction contains the mnemonic for the instruction to be performed.

The operand field of the statement contains the data, the memory address, the port address or the name of the register on which the instruction is to be performed.

The comment field starts with a semicolon and is not part of the machine language program. The comments in a program remind one of the function that an instruction performs in the program.

The input program undergoes processing whereby the assembler reads the source program saved in the storage device (hard disk or a floppy disk). The assembler program generates object code and list files as the output. The object file contains the machine codes for the instruction and information about the addresses of the instructions and is the one to be loaded into the memory for execution. The list file contains the assembly language statements, the binary codes for each instruction in hexadecimal notation, the offset address for each instruction and it also indicates any typing or syntax errors made in typing the source program.

The advantages of an assembly language program are [21]:

- reduced errors
- faster translation times
- changes can be made easier and faster

The disadvantages of assembly language programming are:

- the programmer requires knowledge of the processor architecture and instruction set and thus any change in the hardware requires complete rewriting of the program.

1.3.3 High-Level Language

High-level languages use program statements that are more English-like than those of assembly language. Each high-level language statement may represent many machine code instructions. An interpreter program or a compiler program is used to translate

high-level language statements to machine codes, which can be loaded into memory and executed.

Programs written in high level languages require more memory, execute more slowly than the same programs written in assembly language program. Programs that involve a lot of hardware control such as robots and factory control systems, that must run as quickly as possible are usually written in assembly language program [22, 23]. However, complex data processing programs that manipulate massive amounts of data, such as insurance company records are best written in a high- level language.

1.4 Main memory Subsystem

Memories are a combination of registers, each storing a multi-bit word (program or data). Two distinct types of memories are used:

- Random Access Memory (RAM)
- Read Only Memory (ROM)

The information/data stored in RAM is lost when power is removed from it while that of ROM is retained. RAM are of two types, Static RAM(SRAM) and dynamic RAM(DRAM). DRAM requires periodic refreshing for maintaining the stored information while SRAM does not, which is a major advantage of the former. When program and/or data has to be stored and changed, static or dynamic RAM is used. SRAMS are preferred when the memory size is not too large. For systems requiring large memories DRAMS are used. The main reasons being of their high density, low power consumption and being inexpensive.

ROMs are of two types: masked-programmed ROM and User programmed ROM.

- Masked-programmed.

  A masked-programmed ROM has the information written into it at the time of manufacture.

- User-programmable ROM's are of two types:

  - Programmable ROM (PROM) which can be programmed by the user once only by fusing the selected silicon links.

  -Erasable programmable ROM (EPROM) which can be programmed, erased and reprogrammed by the user a number of times.

Following is a list of features that are examined when selecting a specific memory chip Capacity and organisations, Timing characteristics/AC characteristics, Physical dimensions and packaging, Cost, Reliability and Availability [24].

1.5 Objectives

The general objective of this research project is to design and construct a microprocessor- based, programmable function generator required to generate:

(i) Sine wave,

(ii) Square wave,

(iii) Triangular wave,

(iv) Ramp wave, and

(v) Other pre-stored waveforms.

The use of the 8088 microprocessor for control of the generation of the waveforms is supposed to achieve the following specific objectives:

(i) To have low distortion and stable signals;

(ii) To obtain high frequency stability;

(iii) To be flexible in waveform selection;

(iv) To have fast responses and automatic control; and

(v) To have storage facilities for useful signals.

1.6 Rationale / Justification Of The Study

Most scientific experiments, electronic testing, fault finding and process control systems require stable, accurate and reliable signals. Therefore quicker, reliable, stable and more accurate signal generation can be offered via the use of microprocessors as controllers as attempted in this research work. A microprocessor offers greater flexibility, since bits in memories replace wired connections. Thus, any modification in the hardware design to improve signal generation is replaceable simply by reprogramming the system.

The project undertaken is to produce a prototype that demonstrates the effectiveness of the programmable generator. This work has utilised the numerous advantages of microprocessor-based systems to overcome the shortcomings inherent in the generations of signals such as instability, distortion and slow responses. The sine, triangular and square waves generated can be used for experimental work in the University and College laboratory settings. The square wave that is generated can also be used for sophisticated electronic testing and fault finding in industrial setting.

1.7 Thesis Organisation

This thesis is organised as follows:

Chapter 2 is a literature review that describes the history, development, advantages limitations of various types of signal generators, and the improvements that would be made. Chapter 3 presents a brief description of the research methodology adopted in carrying out this work. In chapter 4, the hardware and software designs and construction of the MBM signal generator is described in detail. The test/calibration results and analysis of the MBM signal generator are presented in chapter 5. Chapter 6 gives the summary, recommendations and suggestions for further work.

CHAPTER 2

LITERATURE REVIEW

2.1 Signal Generators

Research has been conducted on using microprocessors in control and processing by many reseachers. McOwitti A [25] designed and fabricated a measurement system using Z80 microprocessor for maximum power measurement of a voltaic source utilizing a resistor load . Owade M. [26] designed and developed a programmable laboratory interface system with an illustrative use in resistivity temperature measurement using Intel 8085 microprocessor. Research on microprocessor applications and feasibility of using microprocessors in industrial and process control has also been widely done [27,28,29]. In this work, the 8088 microprocessor is used in the control of the generation of standard and random waveforms (signals).

The signals that are usually generated have various wave shapes, which may be a sine wave, a square-wave, or a triangular wave [30, 31, 32, 33].

The amplitudes (V_{pk}) of signals from signal generators usually range from 0 V to $\pm$ 10V, whilst their frequencies range from 0 Hz to 2 MHz [34].

Considering the foregoing discussion, a literature review on analogue and digital signal generators is presented in the following sections, after which the attractive features and advantages possessed by the MBM signal generator are presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Signal Generators

Research has been conducted on using microprocessors in control and processing by many reseachers. McOwitti A [25] designed and fabricated a measurement system using Z80 microprocessor for maximum power measurement of a voltaic source utilizing a resistor load . Owade M. [26] designed and developed a programmable laboratory interface system with an illustrative use in resistivity temperature measurement using Intel 8085 microprocessor. Research on microprocessor applications and feasibility of using microprocessors in industrial and process control has also been widely done [27,28,29]. In this work, the 8088 microprocessor is used in the control of the generation of standard and random waveforms (signals).

The signals that are usually generated have various wave shapes, which may be a sine wave, a square-wave, or a triangular wave [30, 31, 32, 33].

The amplitudes (V_{pk}) of signals from signal generators usually range from 0 V to ± 10V, whilst their frequencies range from 0 Hz to 2 MHz [34].

Considering the foregoing discussion, a literature review on analogue and digital signal generators is presented in the following sections, after which the attractive features and advantages possessed by the MBM signal generator are presented.

2.2 Analogue Signal Generators

Fig. 2.1 illustrates the basic system block design of an analogue signal generator, while the associated real circuit which generates sine, square, and triangular waveforms is presented in Fig. 2.2 [35].

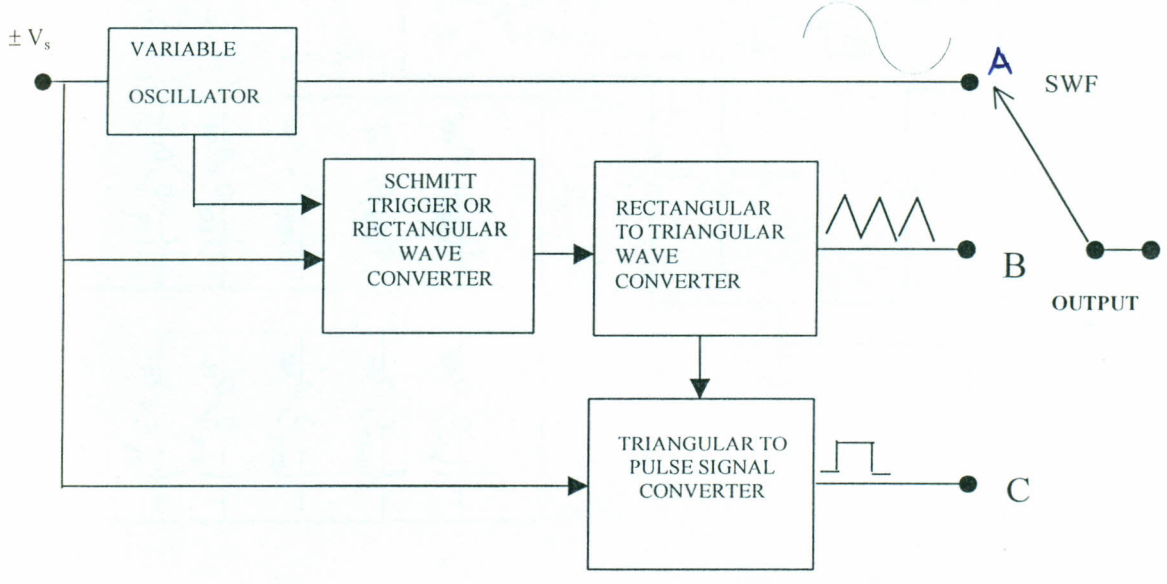


Fig.2.1: Block diagram of an analogue multifunction signal generator.

The heart of an analogue signal generator is the variable oscillator from which a sinusoidal wave can be obtained. Other signal waveforms can be realized by using suitable waveform converters. When the function select switch (SWF) is set at position A, a sinusoidal wave is obtained. When SWF is set at position B, a triangular wave is obtained at the output. SWF, being set at position C, yields a square wave or pulse signal.

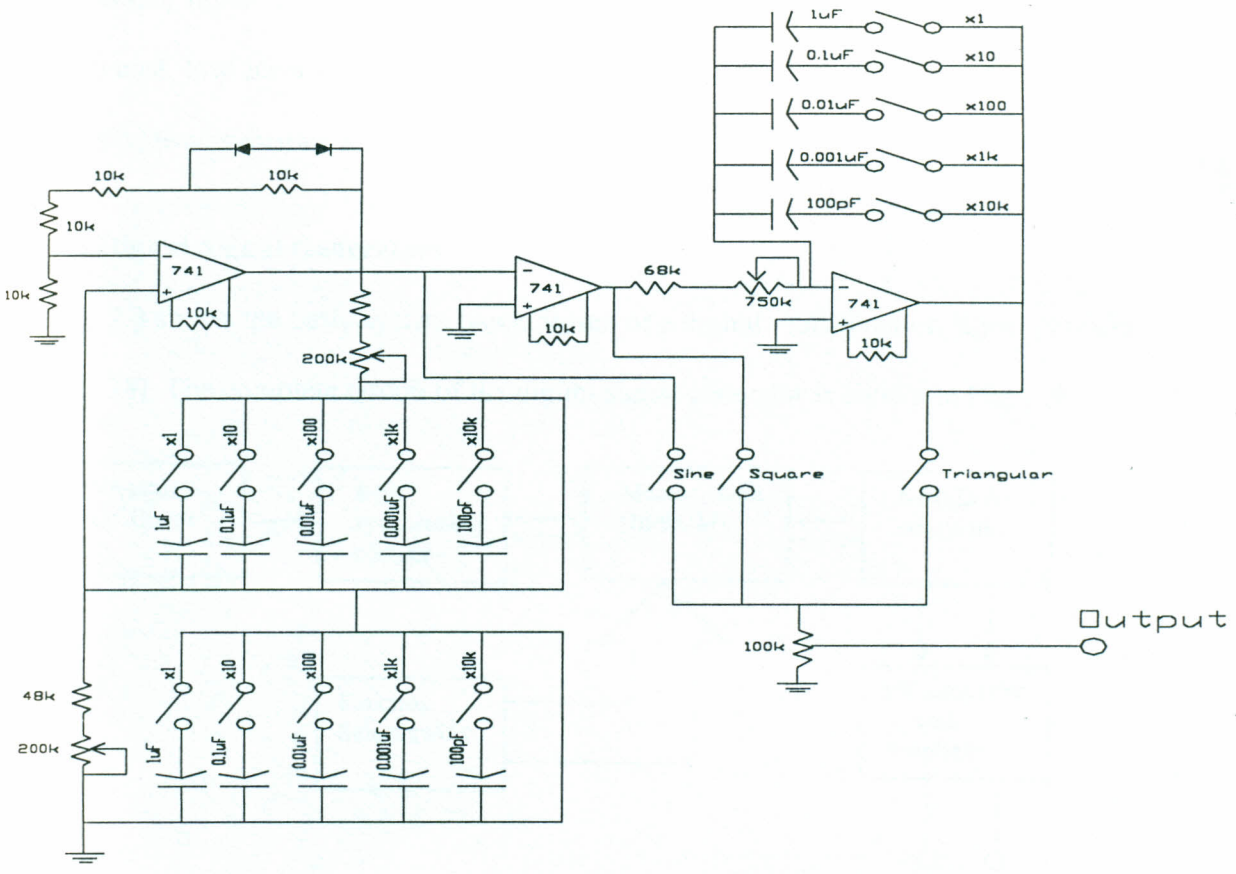


Fig.2.2: Analogue Circuit of multifunction Signal generator

The frequency of the sine wave oscillator is controlled by the selection of any one of the five capacitor values in the oscillator feedback circuit (Fig. 2.2). These capacitors produce the five frequency ranges indicated on the front panel switches. Variation of the frequency within each range is accomplished by adjusting the resistances in the feedback circuit while the desired amplitude is achieved by means of 100K potentiometer.

Analogue signal generators have been observed to possess the following drawbacks: high cost, low accuracy, lack of memory- save- recall facility, low noise immunity, low quality, low precision, and low signal resolution [36].

2.3 Digital Signal Generators

Fig. 2.3 shows the basic system block design of a digital multifunction signal generator [37, 38]. The complete circuit of the digital signal generator is shown in Fig. 2.4

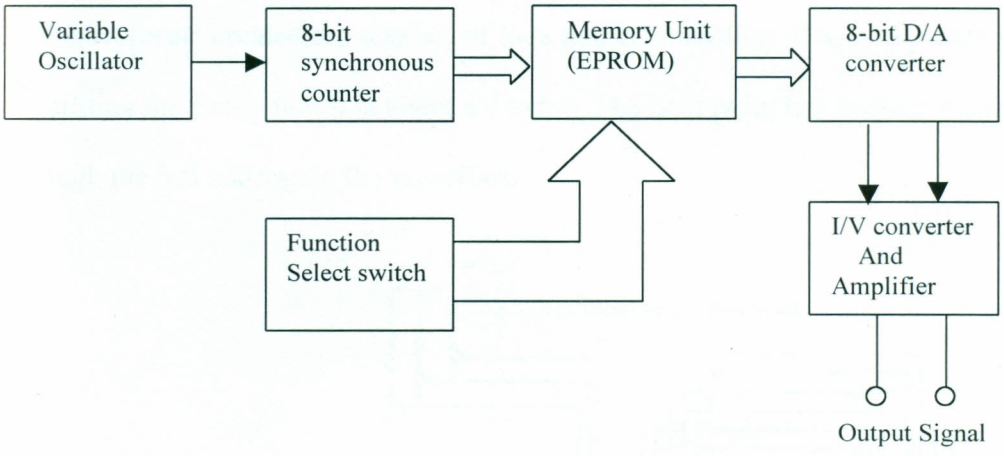


Fig.2.3: The basic block signal diagram of a digital signal generator.

The precision digital-to-analogue converter (DAC) converts the data into analogue voltage values. This series of sequential voltage levels describes the output waveform with the frequency determined by the sample clock rate divided by the number of samples in the waveform. Changing the sample clock rate from the 555 timer, achieved by varying the 100k linear potentiometer (Fig. 2.4), causes the address generator to change the speed at which the data is presented to the DAC and thereby changing the output frequency. The output from pin 3 of the 555 timer is given as a clock to the 4-bit

74163 counter cascaded to another 4-bit 74193 counter. The 8-bit counter produces the output from 00 to FFH, totalling 256 states, which stand as the lower 8- bits of the EPROM. The upper three bits include one that selects the desired functions, and one that changes the page of the EPROM.

When the output of the first counter becomes 1111, the carry is produced which is given to the count-up pin 5 of 74193. After the carry is produced, the first counter is reset and it again starts counting from zero.

The waveforms created are a series of data points consisting of x- and y-axis values. In describing the first point, 0 is given a y value. The next point has another y and so on, through the last address in the waveform.

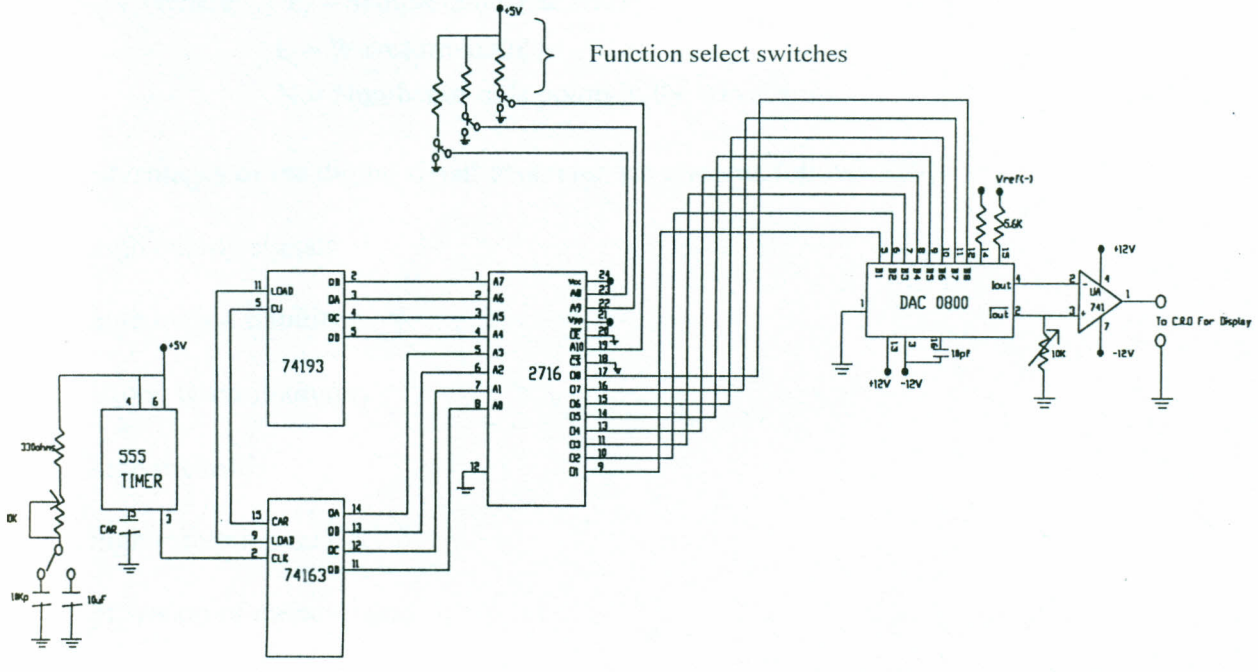


Fig.2.4: The circuit diagram of digital signal generator.

These series of points make the wave shape, which can be stored in large waveform memories typical of digital generators. All the data points in the specified waveform memory locations make up one waveform cycle. The waveform generator will output all the points in the waveform at the sample clock rate specified. The resulting frequency is equal to the sample clock rate divided by the number of data points in the waveform. If multiple cycles of the waveforms are entered into the same waveform memory location, the output frequency will increase proportional to the number of cycles in memory. The waveform frequency $F_{w\,is}$ given by:

$$F_w = \left(\frac{F_s}{L}\right) x\, N$$

Where F_s = Sample clock rate (Hz)
L = Waveform points
N = Number of data points in the waveform (2.1)

The advantages of the digital signal generator are given as follows [39]:

- high quality signals
- high output stability
- strong noise immunity
- high precision
- high resolution and
- provision of memory unit.

However, the demerits of the digital signal generator are [40]:

- lack of logic control circuitry
- lack of control programs
- poor responses in terms of settling time, phase shifts, amplitude and frequency.

The following improvements were to be made on the digital signal generator so as to obtain a microprocessor based signal generator:

(i) Provision of an expanded erasable programmable read-only-memory (EPROM) for greater signal storage.

(ii) Provision of a wide range of frequencies based on the Microprocessors capability.

(iii) Interfacing the signal generator to a microprocessor system for controls.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 An Overview

The methodology adopted in this research work is illustrated by means of a flow chart given in Fig. 3.1. The appropriate hardware components for the system design were selected on the basis of the following factors:

- operating temperature
- frequency response
- current rating and
- voltage rating.

The hardware components include: the 8088 CPU, 14.31818 MHz crystal oscillator, 8284A clock generator, 2.2μF capacitor, 10KΩ resistor, 74LS373 latch, 74LS138 decoder, 7404 quad inverter chip, two 2716 (4K) EPROM, 6116 2K RAM, 8255A programmable peripheral interface, DAC 0800 (digital to analogue converter), 741 operational amplifier, selector switches, reset switch and 10KΩ potentiometer. The circuit designed was simulated using the Electronic Workbench software. Thereafter, the hardware design circuit was implemented and analysed for performance. The design and implementation of the software involved the problem analysis, program design, program coding, assembling, testing and debugging and documentation [41]. To integrate the system software with the system hardware the program was burnt into the erasable programmable read only memory.

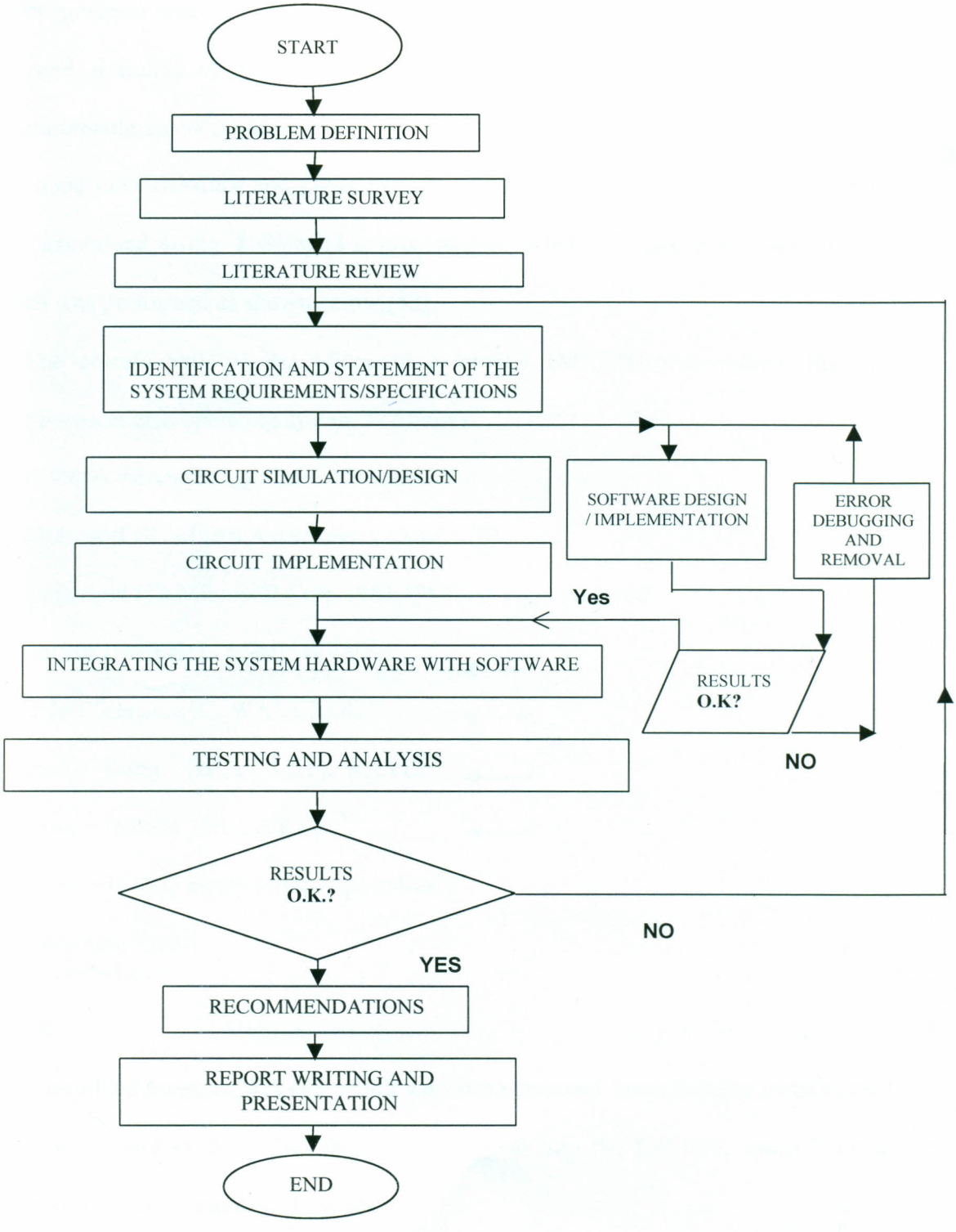


Fig.3.1: Flow chart of the research methodology

In the problem analysis, the requirements of the program were clearly defined. This included defining of the segment registers to be used, port addresses of the programmable input/output port and the memory map nomenclature. Program coding was done with an editor and given WAVEF.ASM file name after which the program was assembled using 8086/8088-microprocessor Microsoft assembler version 5.1. which was performed as shown below [42].

- The correct path of the Microsoft assembler (MASM) was entered into the Microsoft disk operating system as follows:

```
C:\8086>MASM ←┘
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981,1988. All right reserved
Source filename [. ASM]: WAVEF ←┘
Object filename [C: WAVEF.OBJ]: ←┘
Source listing  [NULL.LST]: WAVEF ←┘
Cross reference [NUL.CRF]: ←┘
47696 + 364363 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\8086>
```

The assembled program WAVEF3.OBJ was then converted into a suitable format (Intel Hex format or Hex ABS file) that could be burnt into the EPROM, using Paragon LOD186 Loader- Version 4.0H as follows:

C:\8086>LOD186 ←┘

Paragon LOD186 Loader- Version 4.0h

Copyright (C) 1983-1986 Micro-tech Research Inc.

ALL RIGHTS RESERVED.

Object/Command File [.OBJ]:WAVEF ←┘

Output Object File [C: WAVEF.ABS]: ←┘

Map Filename [C:NULL.MAP]: ←┘

**LOAD COMPLETE

3.2 Programming the erasable programmable read only memory (EPROM 2716)

The WAVEF.ABS file was loaded into the erasable programmable read only memory (EPROM 2716) with the help of EPROM programmer Omnipro2 as follows [43].

- The pins were checked for proper contact with the socket in the Omnipro2 EPROM programmer.
- The EPROM was checked whether it was inserted in the correct orientation in the Omni-pro2 EPROM programmer.
- The EPROM was then checked whether it was blank/erased. A blank EPROM has all its bits set to logical 1(i.e. the content of all the memory locations are FFH). After a blank check failure, the EPROM was erased using EPROM eraser, which exposes ultraviolet light through the glass window of the EPROM.

- After the success of the above check, the appropriate address of the random access memory (RAM) of Omnipro II corresponding to the physical address of the fabricated hardware of the 8088 microprocessor controlled signal generator was given and data (WAVEF.ABS file) written to it from the personal computer.
- The EPROM was then brought in the program mode and all the data stored in the RAM locations were transferred in the EPROM.
- After the chip was programmed, it was brought in the verify mode whereby all the programmed contents in the EPROM were compared with the RAM contents. The verify signal was displayed if the comparison was correct.

The program was then integrated with the hardware for purpose of testing. The troubleshooting of the hardware was carried with the help of logic probe, logic pulsars and cathode ray oscilloscope. The software errors encountered during testing were debugged and the whole process repeated until it was ascertained that the program was free from errors. The final program was integrated with the hardware and the whole system tested and analysed for overall performance.

3.3 Testing and Analysis

The integrated system was evaluated using the following procedures.

3.3.1 Technique No. 1 (Direct Testing Method)

This method of testing and analyzing the output signals of the MBM signal generator directly does not require any electronic device to be connected between the signal

generator and the test equipment but analyzed by connecting it directly to the test equipment [44].

In this case, the microprocessor based signal generator was connected to a cathode ray oscilloscope (CRO) and a logic analyzer as shown in Fig. 3.2. This technique was adopted for testing the generated signal waveforms and in the calibration of the MBM signal generator.

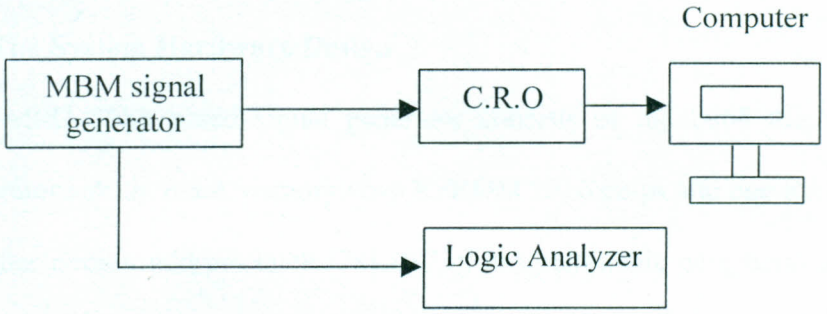


Fig. 3.2: Schematic diagram of system evaluation: Technique No. 1

3.3.2 Technique No. 2 (Indirect Testing Method)

Finally, this method used in evaluating the performance of the MBM signal generator involves the connection of an electronic device under test in between the signal generator and the CRO, and then analyzing the output signal waveforms (See Fig. 3.3).

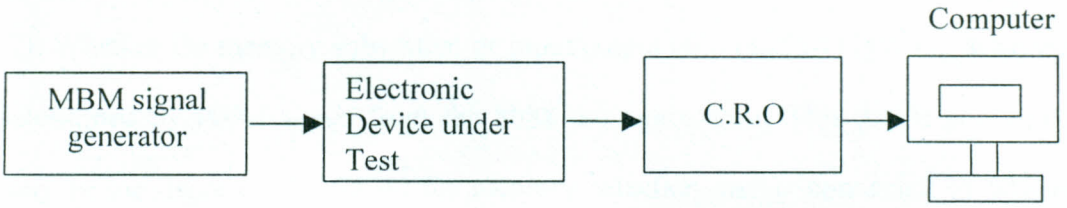


Fig. 3.3: Schematic diagram of system evaluation: Technique No. 2.

CHAPTER 4

THE SYSTEM DESIGNS

The system design consists of two parts, namely: the hardware and the software designs. This chapter describes the various components of the two designs, their operation and performance.

4.1 The System Hardware Design

The MPU 8088 based signal generator consists of the 8088 microprocessor, clock generator circuit, main memory (two EPROM 2716 chips and one RAM 6116 AP chip), decoder circuit, address latch (74LS373), programmable peripheral interface (8255A), selector switches, digital to analogue converter (0800 DAC) and current to voltage converter (741 operational amplifier).

4.1.1 MBM Signal Generator Circuit Description, Operation and Construction

The system block diagram of the MBM signal generator is illustrated in Fig. 4.1. On powering up the system the reset circuit initialises the system after which the wave generation software is loaded from the memory subsystem (EPROM No.1 and EPROM No.2). Whether the memory subsystem or input/output chip (8255A) is being accessed is determined by IO/$\overline{M}$ signal from the 8088 microprocessor. This pin is connected directly to the decoder (74LS138) for memory selection and is connected to 8255A programmable peripheral interface active low chip-enable pin through a NOT gate for inverting the IO signal. The EPROM being accessed is dictated by the chip enable

signal, which depends on the decoding circuitry (74LS138 decoder). The 8255A chip has its ports configured as follows:

Port A as input port for waveform selection, Port B as an output port for outputting the digital word corresponding to desired wave function and port C as an input port for frequency selection. The digital-to-analogue converter (DAC 0800) converts the digital signal from port B of the 8255A PPI to a corresponding analogue signal equivalent. The current output from the DAC 0800 is then applied to a current-to-voltage converter (OPAMP 741) and the peak-to-peak amplitude of the analogue signal is obtained or adjusted by means of a 10 KΩ potentiometer. The final analogue output signal is then fed into a CRO for display and analysis.

The system components were assembled and mounted in their respective positions on the breadboard. Using international electronics data sheets [45,46,47], the pin-out diagrams of ICs and the terminal configurations of the other parts used were obtained. Pin-to-pin, terminal-to-terminal and all the required interconnections were carried out. The joints and contacts were soldered wherever necessary. The complete circuit diagram of the MBM signal generator is displayed in Fig. 4.2. The various components of the system are described in the following subsections.

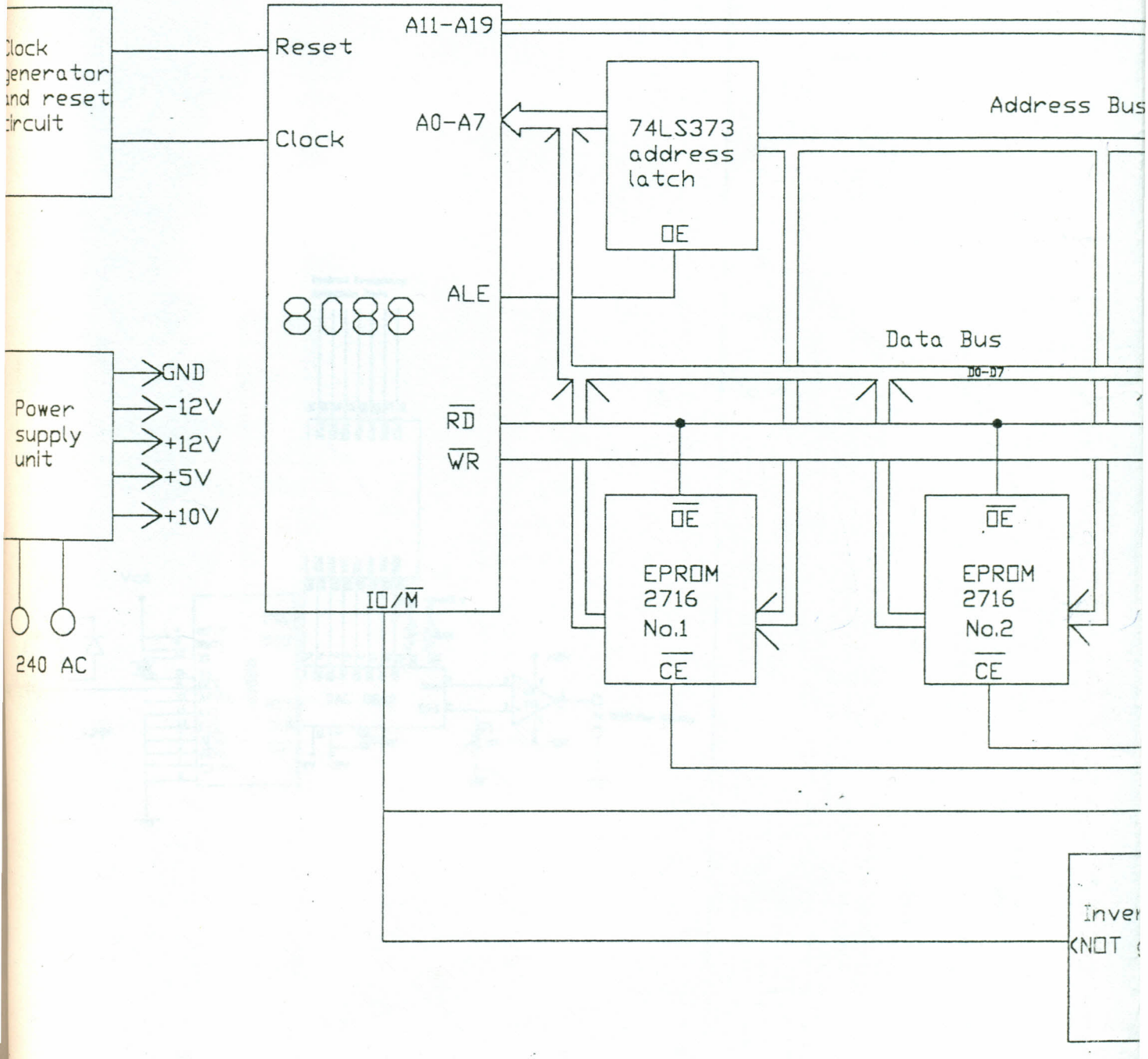
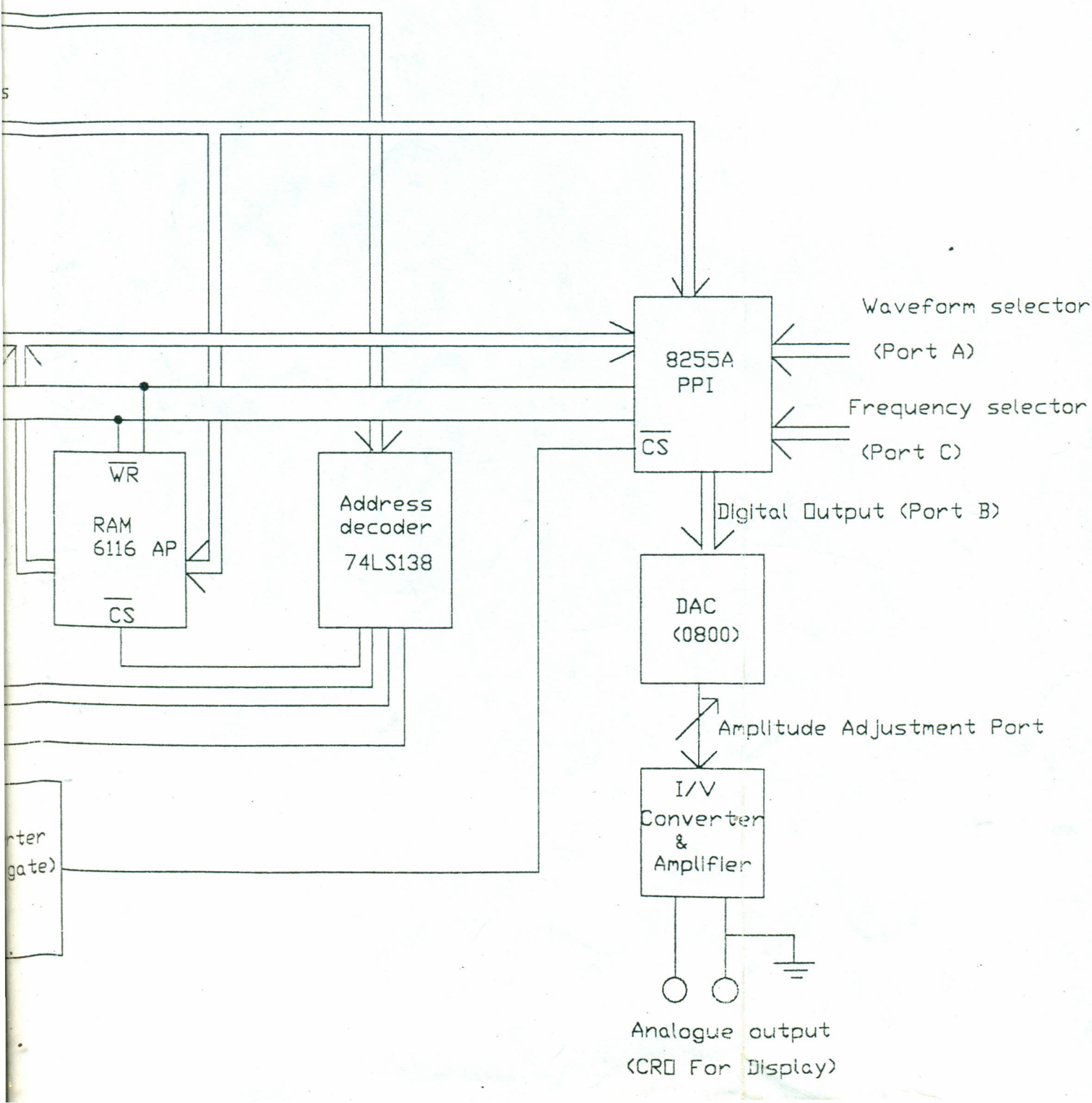


Fig. 4.1: The System Block Diagram Of The MCM Signal Generator.


8255A
PPI
CS
Waveform selector
(Port A)
Frequency selector
(Port C)
Digital Output (Port B)
WR
RAM
6116 AP
CS
Address
decoder
74LS138
DAC
(0800)
Amplitude Adjustment Port
I/V
Converter
&
Amplifier
Analogue output
(CRO For Display)

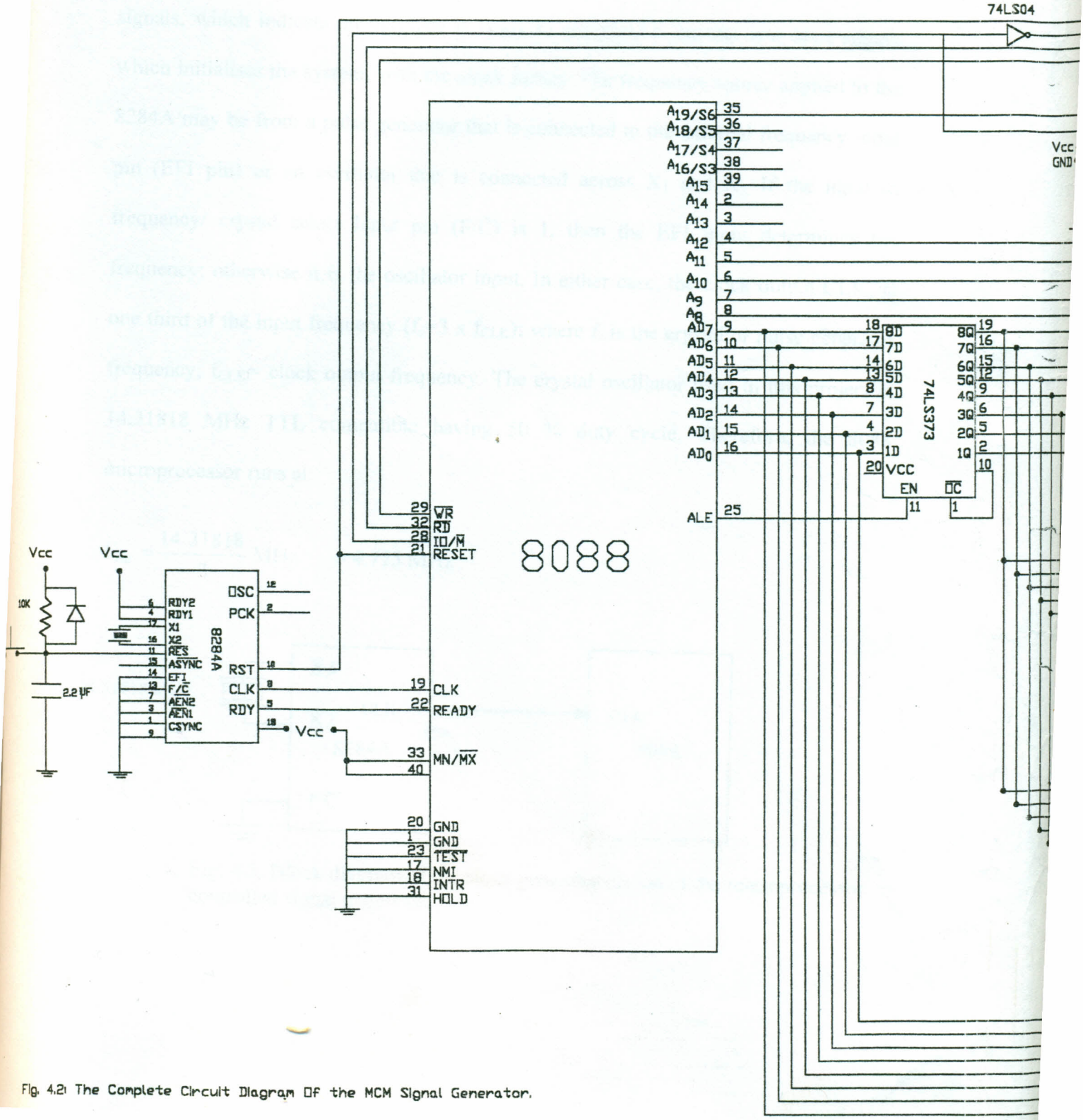
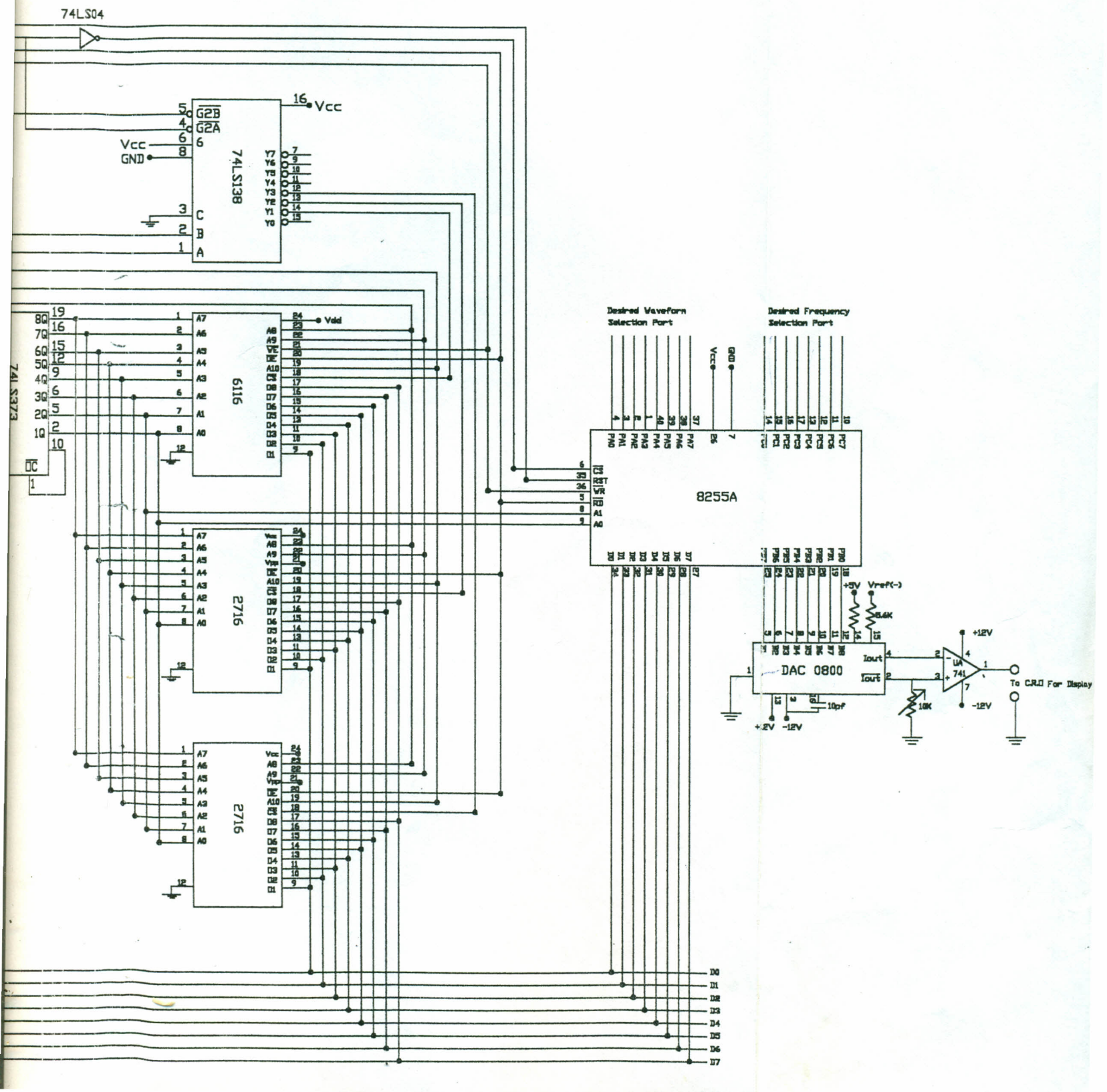


Fig. 4.21 The Complete Circuit Diagram Of the MCM Signal Generator.


74LS04
74LS138
Vcc
GND
74LS373
6116
2716
2716
Vdd
Desired Waveform Selection Port
Desired Frequency Selection Port
8255A
DAC 0800
Iout
Iout
+5V
Vref(-)
8.6K
10K
10pf
+12V
-12V
+12V
-12V
UA 741
To CRO For Display
D0
D1
D2
D3
D4
D5
D6
D7

4.1.2 8284A Clock Generator

The 8088 requires a clock signal with fast rise and fall times (10ns maximum) between low and high voltages. This is provided by Intel 8284A clock generator and driver.

This generator supplies a train of pulses at a constant frequency and synchronizes ready signals, which indicate an interface is ready to complete a transfer and reset signals which initialises the system, with the clock pulses. The frequency source applied to the 8284A may be from a pulse generator that is connected to the external frequency input pin (EFI pin) or an oscillator that is connected across X_1 and X_2. If the input to frequency/ crystal select input pin ($F/\overline{C}$) is 1, then the EFI input determines the frequency; otherwise it is the oscillator input. In either case, the clock output CLK, is one third of the input frequency ($f_c=3 \times f_{CLK}$); where f_c is the crystal or pulse generator frequency; f_{CLK}= clock output frequency. The crystal oscillator used in this project is 14.31818 MHz TTL compatible having 50 % duty cycle. Therefore, the 8088 microprocessor runs at:

$$F_{clk} = \frac{14.31818}{3} \text{ MHz} \quad = 4.773 \text{ MHz}$$

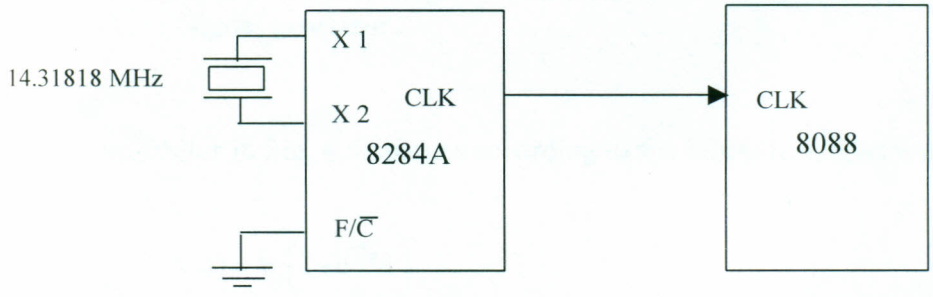


Fig. 4.3: Block diagram of the clock generator circuit of the microprocessor controlled signal generator.

4.1.3 Reset

Reset is for inputting a system-reset pulse. A switch that allows the operator to reinitialize the system manually generates the reset pulse in this system. The 8088 require an active high reset after which the components go to their "turn on" state. For this microprocessor, this turn on state clears the flags, Queue, instruction pointer, data segment register, stack segment register, extra segment register and sets the physical address of the code segment to FFFF0H. The 8088 microprocessor begins to execute instructions from this point and hence an inter-segment direct jump (JMP) instruction has been put in this location to load the wave generation software for this microprocessor controlled signal generator.

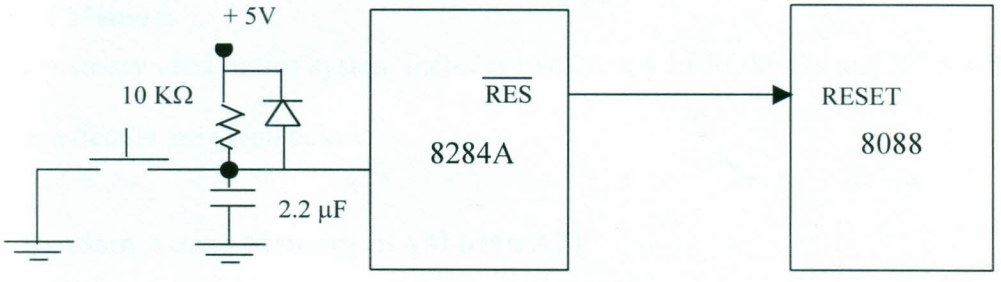


Fig. 4.4: Block diagram of the reset circuit of the microprocessor controlled signal generator.

The capacitor in Fig. 4.4 charges according to the formula in Equation 4.1 [48]:

$$V(t) = V_o\left(1 - e^{-t/RC}\right) \qquad (4.1$$

From the A.C. characteristics of the 8088 microprocessor $V_{LOW(max)} = 0.8$ V and time it takes the microprocessor to read the RESET input after power up = 50 μs. Substituting for these values in equation 4.1, we get:

$$0.8 = 5\left(1 - e^{-50 \times 10^{-6} / \tau}\right) \tag{4.2}$$

From which the minimum time constant required is

$$\tau = 0.28\text{ms} \tag{4.3}$$

which could then be chosen from this value upwards such that V(t) approaches zero. Assuming a capacitance of 2.2 μF, the minimum resistance would be 128 Ω. It is therefore clear that the choice of R (10KΩ) and C (2.2 μF) gives a time constant which is well above the minimum required time constant.

4.1.4 Memory

The memory used in this system includes two 2K x 8 EPROM ICs and 2K x 8 RAM IC whose details are given below.

a) Random Access Memory (RAM 6116 AP)

The RAM 6116AP is a static read/write memory a low type complementary metal oxide semiconductor organised as 2048 words x 8 bits with access time 120ns and voltage supply of +5v dc. The address pins of the 6116AP are directly connected to $A_0 - A_{10}$ on the demultiplexed address bus while the data pins are connected to the time multiplexed data bus on the 8088 microprocessor.

The read enable $\overline{RD}$ and the write enable $\overline{WE}$ signals from the microprocessor are directly connected to the 6116 AP RAM to enable read and write operations respectively.

The RAM used in this system is for stack operations and therefore stack segment is mapped onto it.

b) Erasable Programmable Read Only Memory (EPROM 2716)

The 2716 EPROM, 2048 memory locations erasable programmable read only memory used in this project, is an example of metal oxide semiconductor with access time of 450ns. EPROM 2716 is designed to work with microprocessor buses. Address lines ($A_0 - A_{10}$) of the EPROM are connected to the demultiplexed address bus ($A_0 - A_{10}$) of the 8088 microprocessor. The time multiplexed address bus/data bus ($A_0 - A_7$) of the 8088 microprocessor is connected to the data output of the EPROM. The 8088 read control line drives the output enable $\overline{OE}$ of the 2716 and enables the output data onto the data bus from it with the proper timing to prevent bus contention problem. Generally, the 11-bit address is submitted to an address decoding circuitry and a data word from the cell array is delivered at the data output pins. Because no on chip fuse destruction takes place, there is little risk of damaging the silicon by applying bad programming signals.

c) Memory Map

The 8088 microprocessor is directly mapped to location FFFF0H when power is first applied and the first instructions (bootstrap loader) are kept in this location onwards. In this research work, two EPROM ICs are used with the first one mapped from location

FF000H~ FF7FFH and the second one mapped from location FF800H~FFFFFH. The RAM IC used (6116 AP) has been mapped to location FE800H~FE7FFH (see Table 4.1 for memory map nomenclature).

Table 4.1 Memory map nomenclature of the MBM signal generator.

ADDRESS	MEMORY	TYPE USED
00000~FE7FFH	Unused	-
FE800~FEFFFH	RAM	6116
FF000~FF7FFH	EPROM	2716
FF800~FFFFFH	EPROM	2716

d) Memory Address Decoding Circuitry

The chip select $\overline{CS}$ inputs of the RAM and EPROM are driven by the transistor-transistor logic (TTL) decoder (74LS138) which maps the RAM 6116AP to FE800H ~ FEFFFH memory locations. It maps the two EPROMs to address space (FF000H ~ FFFFFH) addressable by the 8088 microprocessor. The chip select lines of RAM and EPROMs, driven by the decoder, are brought to logic 0 when either of them is selected by the microprocessor. See Fig. 4.5.

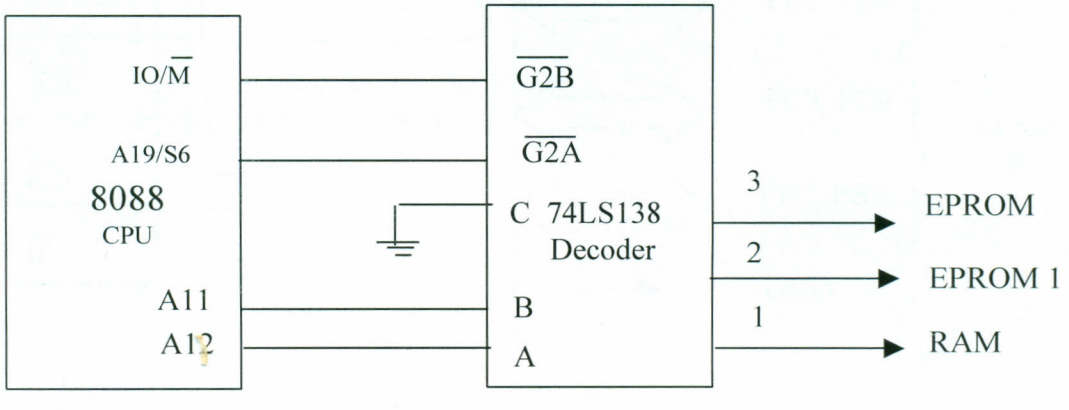


Fig. 4.5: The decoder configuration

4.1.5 Programmable Peripheral Interface (8255A PPI)

Intel's 8255A is a programmable peripheral interface fabricated using NMOS technology and is contained in a 40-pin dual inline package. It is a parallel interface where data/bits (D7_D0) can simultaneously be transferred over separate lines. 8255A PPI contains a control register and three separately 8-bit addressable input/output ports. The ports are labeled A, B and C. Whether or not an 8255A is being accessed is determined by the signal on the $\overline{CS}$ pin and the direction of access is according to the read $\overline{RD}$ and write $\overline{WR}$ signals. The bits in the three ports are attached to the pins that may be connected to the input/output device. These bits are divided into groups A and B, with group A consisting of bits in port A (PA7_PA0) and the four most significant bits (MSBs) of port C (PC7_PC4) and group B consisting of port B (PB7_PB0) and the four least significant bits (LSBs) of port C (PC3_PC0). Figure 4.6 gives the various groups of the PPI.

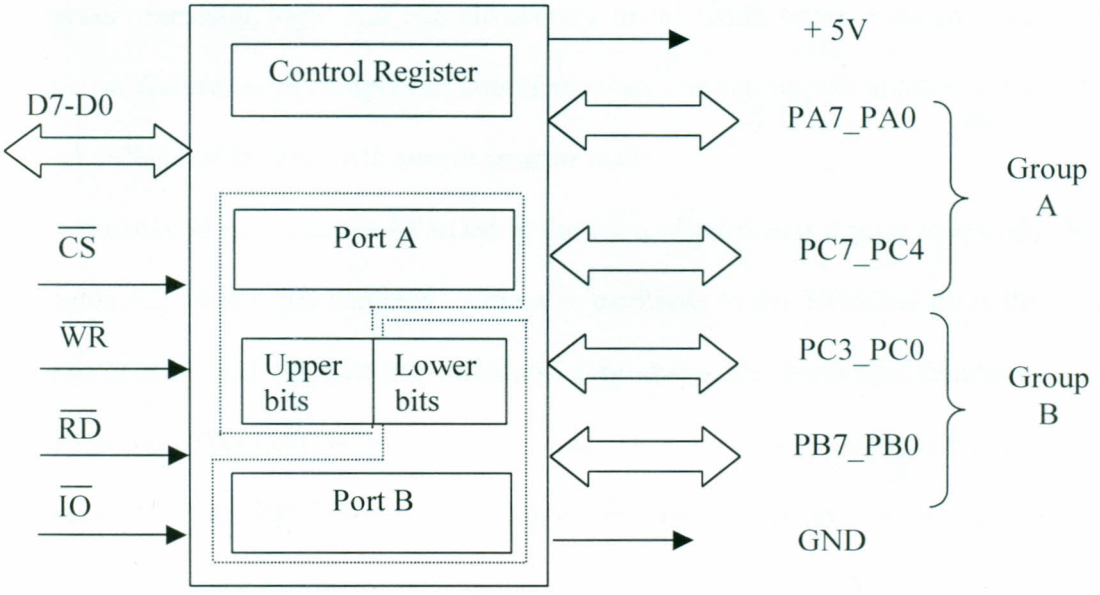


Fig. 4.6: Block diagram of the 8255A PPI

In this research project, the 8255A has been used in mode 0, a basic input/output mode in which ports A and B can be individually configured as input or output, port C upper or lower bits can also be individually configured as input or output. Therefore port A has been configured as an input port for waveform selection, port B as an output port for outputting the waveform digital codes and port C as an input port for frequency selection. To achieve this, 10011001B(99H) is written on to the control register of the 8255A by the control program (See appendix A).

4.1.6 Digital to Analogue Converter (DAC 0800)

The DAC 0800 is a monolithic 8-bit high-speed current output digital-to-analog converter (DAC) featuring typical settling times of 100ns, full scale error of ± lsb, non linearity over temperature ± 0.1%, high output compliance –10V to + 18V, wide power supply range ±4.5 to 18V, complementary current outputs and interfaces directly with transistor transistor logic and complementary metal oxide semiconductor. The DAC 0800 also features high compliance complementary current outputs to allow differential output voltage of 20 Vpp with simple resistor loads.

Each function is stored in the EPROM in the form of particular digital codes (B7_B0), see table 4.2. When the complete address is available to the EPROM from the 8088 microprocessor and the selector switches, it produces the particular function in the digital output. This DAC is used to convert the digital codes into analogue form that can be displayed in the PM 3384a-autoranging combiscope. The DAC was wired as shown in Figure 4.7 below and its associated operation is shown in Table 4.2.

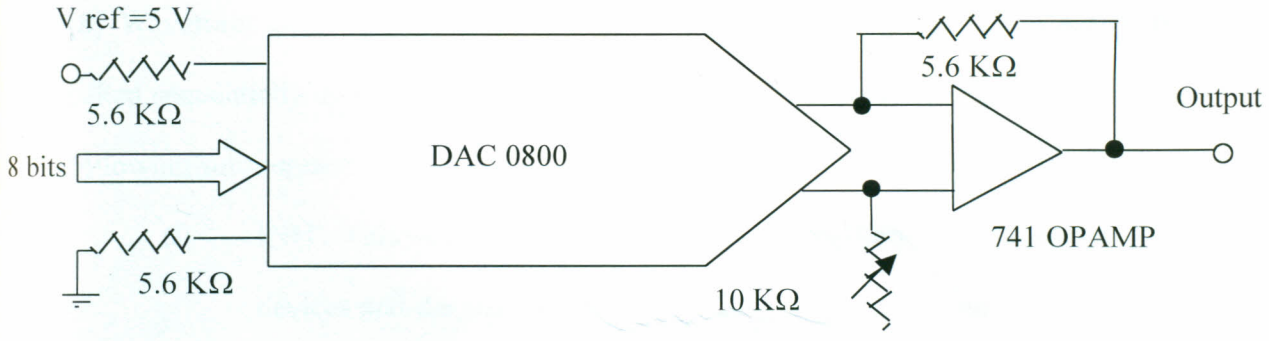


Fig. 4.7: DAC 0800 wiring configuration

Table 4.2: Symmetrical Offset Binary Operation Of The DAC.

Bits	B_7 B_6 B_5 B_4 B_3 B_2 B_1	Hex
Positive full scale (+lsb)	1 1 1 1 1 1 1 1	FFH
Positive scale (–lsb)	1 1 1 1 1 1 1 0	FEH
Zero scale (+)	1 0 0 0 0 0 0 0	80H
Zero scale (-)	0 1 1 1 1 1 1 1	7FH
Negative full scale (+lsb)	0 0 0 0 0 0 0 1	01H
Negative full scale	0 0 0 0 0 0 0 0	00H

4.3 The System Software Design

The software designed to run in this system consists of two parts: bootstrap loader and the system control software (waveform generation software).

(a) Bootstrap loader: the bootstrap loader initialises the code segment to start at memory location FF000H (the first memory location of the cascaded EPROMs) where the main program resides (waveform generation software).

(b) Waveform generation software: consists of several subroutines, which are to be called sequentially as required for execution. The main system software consists of the following subroutines:

(i) INIT: This subroutine initialises segment registers, programmable port devices and the stack pointer. The data segment was initialised to start at memory location FF000H and the stack segment to start at the first memory location of the RAM (FE800H) with stack pointer loaded with 100H as an offset to the stack segment. The binary number 10011001 was loaded to the 8255A PPI control register to configure the ports as follows:

Port A as input port for waveform selection, Port B as output port for outputting the selected digital codes for the different wave-functions and port C as an input port for selecting the desired frequency.

(ii) POLL: This subroutine selects the desired waveform out of the six to give the appropriate signal function.

(iii) DELAY 1: This subroutine selects the desired signal frequency out of the eight different frequencies.

Algorithms for system software for the above modules (see Fig. 4.8) have been developed and coded in 8086/8088-assembler language and the control program listing is presented in Appendix A.

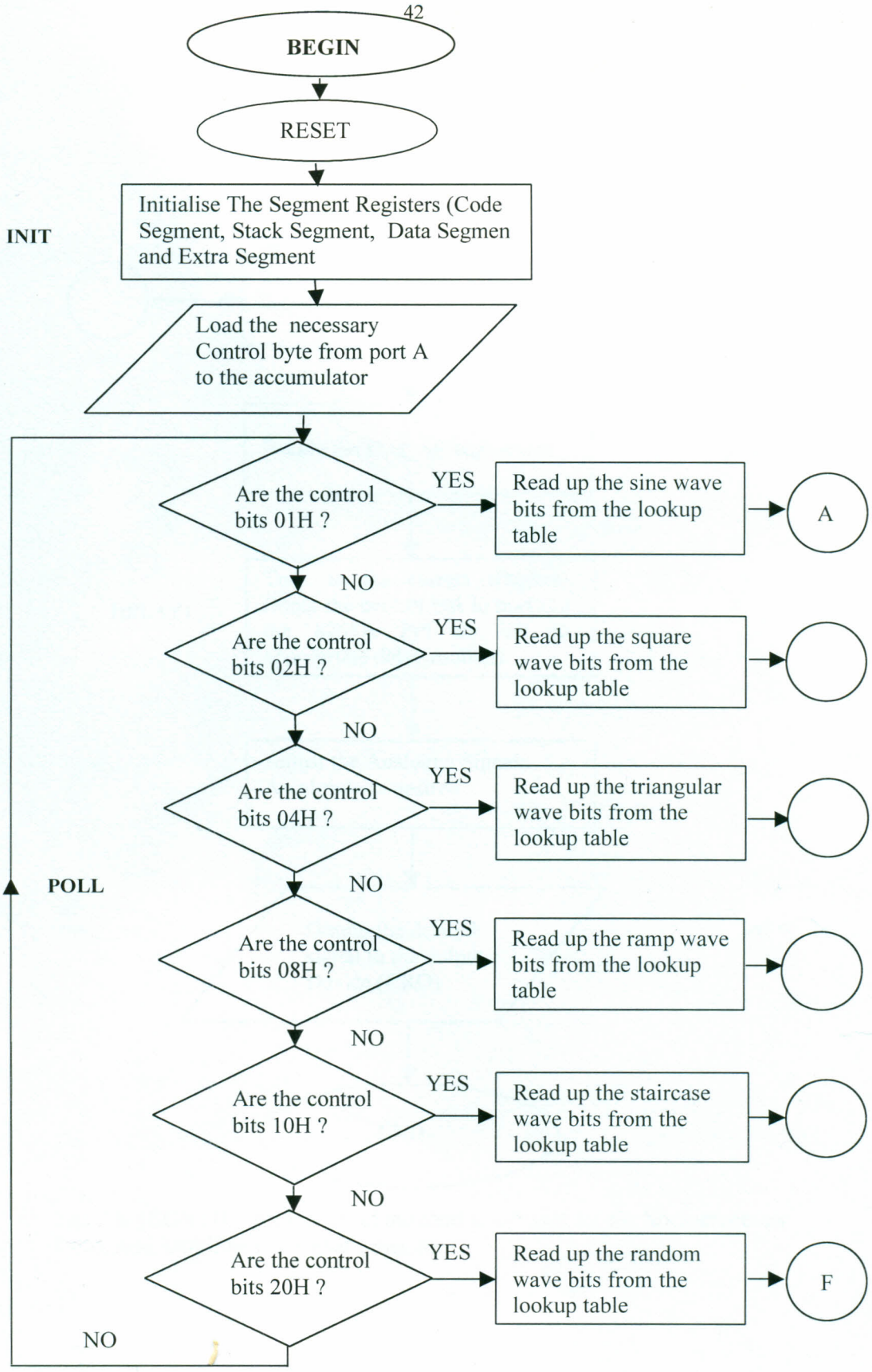


Fig. 4.8: Flow chart of the control software for the Microprocessor Controlled Multifunction signal generator.

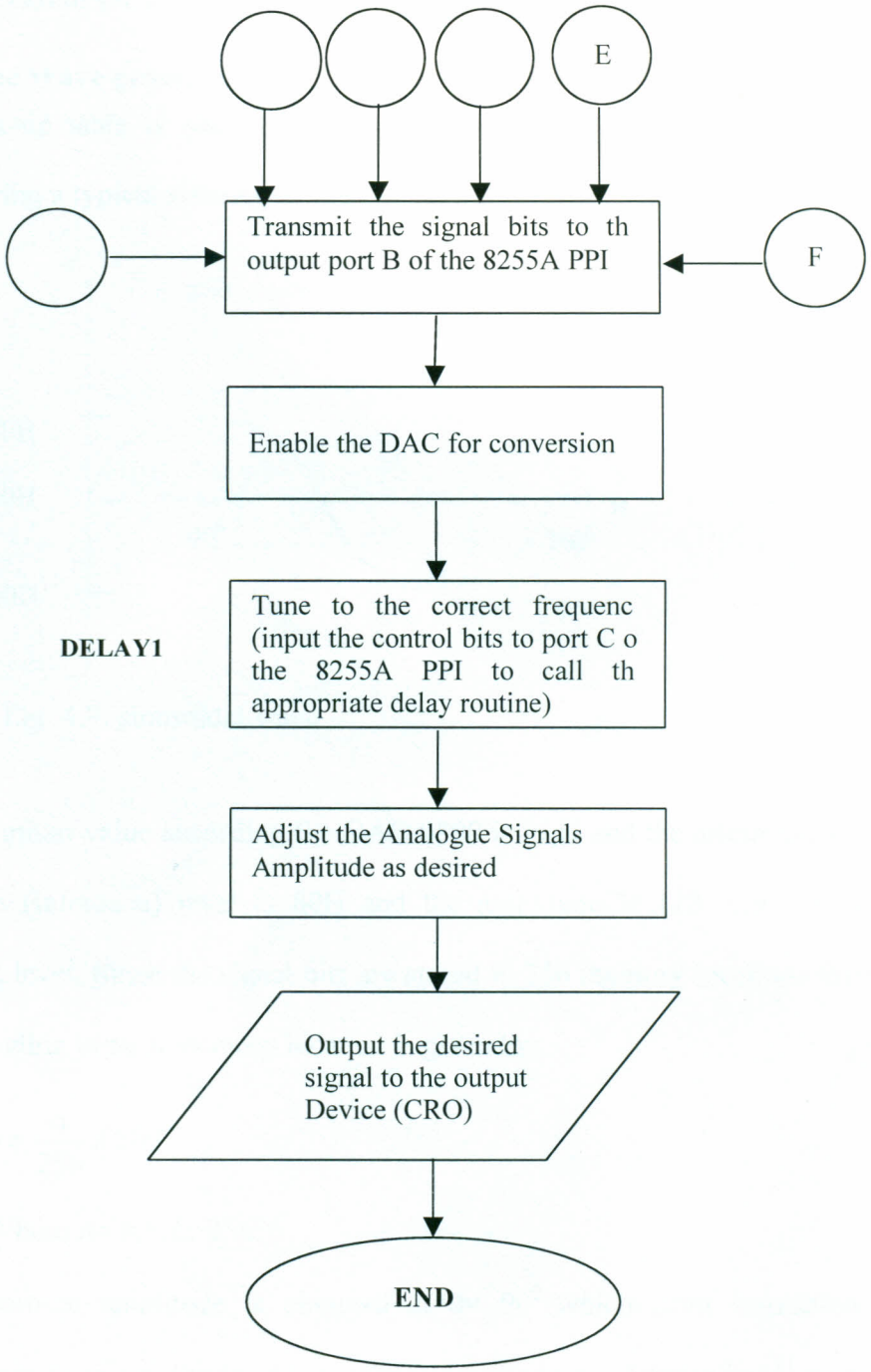


Fig. 4.8 (CONTD.): Flow chart of the control software for the Microprocessor Controlled Multifunction signal generator.

4.4 Generation Of The Signal Waveform Bits

4.4.1 Sine Wave generation

The look-up table is obtained through an analysis, which is described as follows. Considering a typical sinusoidal wave as shown in Figure 4.9

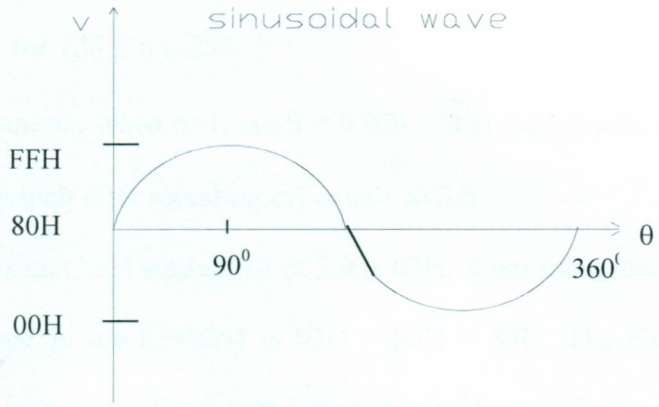


Fig. 4.9: sinusoidal wave

The maximum value according the DAC 0800 is FFH and the minimum value is 00H. The zero (reference) level is 80H and the amplitude is 128 units above the zero reference level. Since the signal bits are stored in 256 memory locations then the angle corresponding to each memory location is given by:

$$\theta = \frac{n}{256} \times 360$$

Where n= 0,1,2.. 256

The maximum amplitude is obtained at $\theta = 90^0$ which from inspection is 1 and corresponds to an amplitude of 128 units above the zero reference level according to the configuration of the digital to analogue converter. Therefore each digital word corresponding to each angle is given by:

$$\sin\left(\frac{n}{256} \text{ x } 360\right) \text{ x } 128 \text{ units in Hex form + 80H}$$

for $0 \leq n \leq 128$ and

$$80H - \sin\left(\frac{n}{256} \text{ x } 360\right) \text{ x } 128 \text{ units in binary form}$$

for $128 \leq n \leq 256$

For example, when n=1, $\sin \theta = 0.02450$ and the decimal equivalent is 0.0245^0 x 128 = 3.136 which after rounding off equals to 3.0.

The hexadecimal equivalent of 3.0 is 03H. Since the reference point is 80H, the value to be stored in the EPROM is 03H + 80H = 83H. The digital equivalent for the other angles were determined in the same way and stored in the look-up Table 4.3

4.4.2 Triangular Wave generation

The triangular wave (Figure 4.10) is represented by the functions below:

$$y = \frac{2}{\pi}t, \quad y = -\frac{2}{\pi}t + c, \; y = \frac{2}{\pi}t - c \quad \text{Where c is a constant.}$$

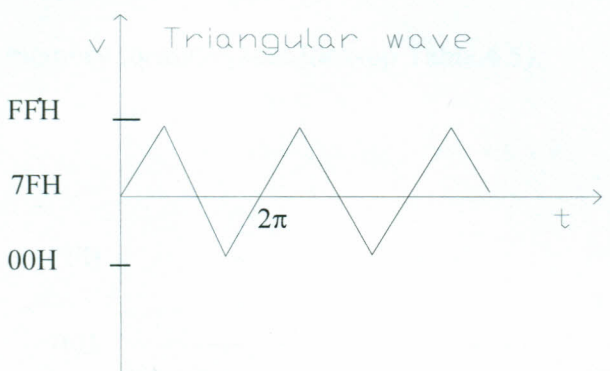


Fig. 4.10: Triangular wave.

The amplitude is 128 units above the reference level that corresponds to $\frac{\pi}{2}$ rad

The general formula for determining the amplitudes is

$$\text{amplitudes} = \frac{n}{256} x2\pi \quad \text{where} \quad n = 0, 1, 2..256$$

which after conversion into units above the reference level is given by $2n$

The byte to be stored in the EPROM is then determined by:

2n (in Hex format) + 7FH for $0 \leq n \leq 128$,

FFH – 2n (in Hex format) for $128 \leq n \leq 192$ and

00H – 2n (in Hex format) for $192 \leq n \leq 256$

after which look-up Table 4.4 was prepared.

4.4.3 Square Wave generation

$y = A, y = -A$ where A and - A represents
FFH and 00H (the amplitudes of the square wave signal) respectively.

which is illustrated in Figure 4.11. The bytes stored in the EPROM to realise one cycle of the square wave were FFH for the first 128 memory locations and 00H for the next 128 memory locations (see look-up Table 4.5).

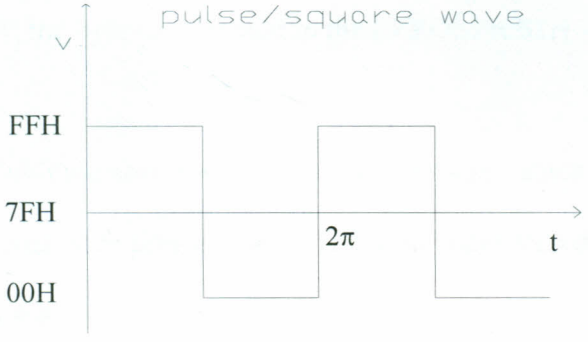


Fig. 4.11: Square wave

4.3.4 Ramp Wave generation

The ramp waveform was realized from the formula $y = \frac{1}{2}\pi t$ (Fig. 4.12)

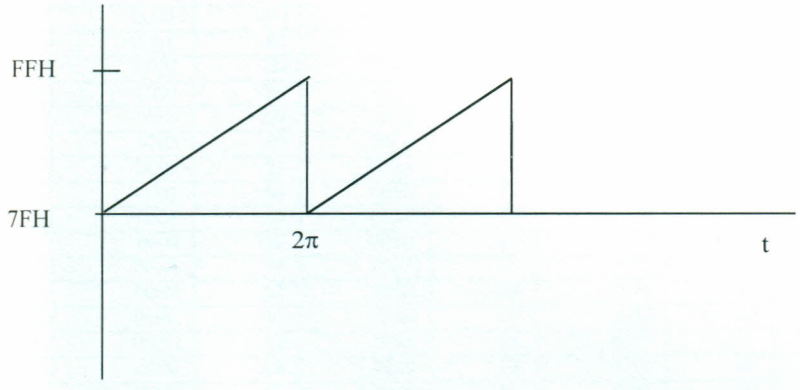


Fig. 4.12: Ramp Waveform

The general formula for determining the amplitudes is:

$$\text{amplitudes (in rad)} = \frac{n}{256} \text{x} 2\pi \quad \text{where} \quad n = 0, 1, 2..256$$

The general formula for determining the amplitudes in decimal is:

$$\text{Amplitude in decimal} = \frac{n}{2} \text{ and hence the value to be stored in the EPROM is}$$

$$\frac{n}{2}\text{(in Hex)} + 7\text{FH}, \quad \text{where} \quad n = 0, 1, 2..256$$

For example:

For n=1, the byte to be stored in the EPROM is 01H + 7FH =80H (see look-up Table 4.6)

Along with the above waveforms, the look-up Tables for staircase and random waveforms were prepared and the digital codes stored in the EPROM, look-up Tables 4.7 and 4.8 respectively.

Table 4.3: Software-Look up Table For Sine wave

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0000	80	0034	FB	0068	C6	009C	2D	00D0	0B
0001	83	0035	FC	0069	C3	009D	2B	00D1	0C
0002	86	0036	FC	006A	C1	009E	29	00D2	0D
0003	89	0037	FD	006B	BE	009F	26	00D3	0F
0004	8D	0038	FE	006C	BB	00A0	24	00D4	10
0005	90	0039	FE	006D	B8	00A1	22	00D5	11
0006	93	003A	FF	006E	B5	00A2	20	00D6	14
0007	96	003B	FF	006F	B3	00A3	1E	00D7	15
0008	99	003C	FF	0070	B0	00A4	1C	00D8	17
0009	9C	003D	FF	0071	AD	00A5	1A	00D9	19
000A	9F	003E	FF	0072	AA	00A6	18	00DA	1B
000B	A2	003F	FF	0073	A7	00A7	16	00DB	1D
000C	A5	0040	FF	0074	A4	00A8	14	00DC	1E
000D	A8	0041	FF	0075	A1	00A9	13	00DD	21
000E	AB	0042	FF	0076	9E	00AA	11	00DE	23
000F	AE	0043	FF	0077	9B	00AB	10	00DF	25
0010	B1	0044	FF	0078	98	00AC	0E	00E0	28
0011	B4	0045	FF	0079	94	00AD	0D	00E1	2A
0012	B7	0046	FE	007A	91	00AE	0B	00E2	2C
0013	BA	0047	FE	007B	8D	00AF	0A	00E3	2F
0014	BD	0048	FD	007C	8B	00B0	09	00E4	31
0015	BF	0049	FD	007D	88	00B1	08	00E5	33
0016	C2	004A	FC	007E	85	00B2	07	00E6	36
0017	C5	004B	FB	007F	82	00B3	06	00E7	39
0018	C7	004C	FA	0080	7E	00B4	05	00E8	3B
0019	CA	004D	F9	0081	7B	00B5	04	00E9	3E
001A	CD	004E	F8	0082	78	00B6	03	00EA	41
001B	CF	004F	F7	0083	75	00B7	03	00EB	43
001C	D1	0050	F6	0084	72	00B8	02	00EC	46
001D	D4	0051	F5	0085	6F	00B9	02	00ED	49
001E	D6	0052	F3	0086	6C	00BA	01	00EE	4C
001F	D9	0053	F2	0087	69	00BB	01	00EF	4F
0020	DB	0054	F0	0088	66	00BC	00	00F0	51
0021	DD	0055	EE	0089	63	00BD	00	00F1	55
0022	DF	0056	ED	008A	5F	00BE	00	00F2	58
0023	E1	0057	EC	008B	5C	00BF	00	00F3	5B
0024	E3	0058	EA	008C	59	00C0	00	00F4	5E
0025	E5	0059	E8	008D	56	00C1	00	00F5	61
0026	E7	005A	E6	008E	53	00C2	00	00F6	64
0027	E9	005B	E4	008F	50	00C3	01	00F7	67
0028	EB	005C	E2	0090	4D	00C4	01	00F8	6A
0029	EC	005D	E0	0091	4B	00C5	01	00F9	6E
002A	EE	005E	DE	0092	48	00C6	02	00FA	70
002B	F0	005F	DC	0093	45	00C7	02	00FB	73
002C	F1	0060	DA	0094	42	00C8	03	00FC	77
002D	F3	0061	D7	0095	3F	00C9	04	00FD	7A
002E	F4	0062	D5	0096	3D	00CA	04	00FE	7D
002F	F5	0063	D3	0097	3A	00CB	05	00FF	80
0030	F6	0064	D0	0098	37	00CC	06		
0031	F8	0065	CE	0099	35	00CD	07		
0032	F9	0066	CB	009A	32	00CE	08		
0033	FA	0067	C9	009B	30	00CF	09		

Table 4.4: Software-Look up Table For Triangular wave

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0200	7F	0234	E7	0268	AF	029C	47	02D0	21
0201	81	0235	E9	0269	AD	029D	45	02D1	23
0202	83	0236	EB	026A	AB	029E	43	02D2	25
0203	85	0237	ED	026B	A9	029F	41	02D3	27
0204	87	0238	EF	026C	A7	02A0	3F	02D4	29
0205	89	0239	F1	026D	A5	02A1	3D	02D5	2B
0206	8B	023A	F3	026E	A3	02A2	3B	02D6	2D
0207	8D	023B	F5	026F	A1	02A3	39	02D7	2F
0208	8F	023C	F7	0270	9F	02A4	37	02D8	31
0209	91	023D	F9	0271	9D	02A5	35	02D9	33
020A	93	023E	FB	0272	9B	02A6	33	02DA	35
020B	95	023F	FD	0273	99	02A7	31	02DB	37
020C	97	0240	FF	0274	97	02A8	2F	02DC	39
020D	99	0241	FD	0275	95	02A9	2D	02DD	3B
020E	9B	0242	FB	0276	93	02AA	2B	02DE	3D
020F	9D	0243	F9	0277	91	02AB	29	02DF	3F
0210	9F	0244	F7	0278	8F	02AC	27	02E0	41
0211	A1	0245	F5	0279	8D	02AD	25	02E1	43
0212	A3	0246	F3	027A	8B	02AE	23	02E2	45
0213	A5	0247	F1	027B	89	02AF	21	02E3	47
0214	A7	0248	EF	027C	87	02B0	1F	02E4	49
0215	A9	0249	ED	027D	85	02B1	1D	02E5	4B
0216	AB	024A	EB	027E	83	02B2	1B	02E6	4D
0217	AD	024B	E9	027F	81	02B3	19	02E7	4F
0218	AF	024C	E7	0280	7F	02B4	17	02E8	51
0219	B1	024D	E5	0281	7D	02B5	15	02E9	53
021A	B3	024E	E3	0282	7B	02B6	13	02EA	55
021B	B5	024F	E1	0283	79	02B7	11	02EB	57
021C	B7	0250	DF	0284	77	02B8	0F	02EC	59
021D	B9	0251	DD	0285	75	02B9	0D	02ED	5B
021E	BB	0252	DB	0286	73	02BA	0B	02EE	5D
021F	BD	0253	D9	0287	71	02BB	09	02EF	5F
0220	BF	0254	D7	0288	6F	02BC	07	02F0	61
0221	C1	0255	D5	0289	6D	02BD	05	02F1	63
0222	C3	0256	D3	028A	6B	02BE	03	02F2	65
0223	C5	0257	D1	028B	69	02BF	01	02F3	67
0224	C7	0258	CF	028C	67	02C0	01	02F4	69
0225	C9	0259	CD	028D	65	02C1	03	02F5	6B
0226	CB	025A	CB	028E	63	02C2	05	02F6	6D
0227	CD	025B	C9	028F	61	02C3	07	02F7	6F
0228	CF	025C	C7	0290	5F	02C4	09	02F8	71
0229	D1	025D	C5	0291	5D	02C5	0B	02F9	73
022A	D3	025E	C3	0292	5B	02C6	0D	02FA	75
022B	D5	025F	C1	0293	59	02C7	0F	02FB	77
022C	D7	0260	BF	0294	57	02C8	11	02FC	79
022D	D9	0261	BD	0295	55	02C9	13	02FD	7B
022E	DB	0262	BB	0296	53	02CA	15	02FE	7D
023F	DD	0263	B9	0297	51	02CB	17	02FF	7F
0230	DF	0264	B7	0298	4F	02CC	19		
0231	E1	0265	B5	0299	4D	02CD	1B		
0232	E3	0266	B3	029A	4B	02CE	1D		
0233	E5	0267	B1	029B	49	02CF	1F		

Table 4.5: Software Look -up Table For Square wave

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0300	FF	0334	FF	0368	FF	039C	00	03D0	00
0301	FF	0335	FF	0369	FF	039D	00	03D1	00
0302	FF	0336	FF	036A	FF	039E	00	03D2	00
0303	FF	0337	FF	036B	FF	039F	00	03D3	00
0304	FF	0338	FF	036C	FF	03A0	00	03D4	00
0305	FF	0339	FF	036D	FF	03A1	00	03D5	00
0306	FF	033A	FF	036E	FF	03A2	00	03D6	00
0307	FF	033B	FF	036F	FF	03A3	00	03D7	00
0308	FF	033C	FF	0370	FF	03A4	00	03D8	00
0309	FF	033D	FF	0371	FF	03A5	00	03D9	00
030A	FF	033E	FF	0372	FF	03A6	00	03DA	00
03 0B	FF	033F	FF	0373	FF	03A7	00	03DB	00
030C	FF	0340	FF	0374	FF	03A8	00	03DC	00
030D	FF	0341	FF	0375	FF	03A9	00	03DD	00
030E	FF	0342	FF	0376	FF	03AA	00	03DE	00
030F	FF	0343	FF	0377	FF	03AB	00	03DF	00
0310	FF	0344	FF	0378	FF	03AC	00	03E0	00
0311	FF	0345	FF	0379	FF	03AD	00	03E1	00
0312	FF	0346	FF	037A	FF	03AE	00	03E2	00
0313	FF	0347	FF	037B	FF	03AF	00	03E3	00
0314	FF	0348	FF	037C	FF	03B0	00	03E4	00
0315	FF	0349	FF	037D	FF	03B1	00	03E5	00
0316	FF	034A	FF	037E	FF	03B2	00	03E6	00
0317	FF	034B	FF	037F	FF	03B3	00	03E7	00
0318	FF	034C	FF	0380	00	03B4	00	03E8	00
0319	FF	034D	FF	0381	00	03B5	00	03E9	00
031A	FF	034E	FF	0382	00	03B6	00	03EA	00
031B	FF	034F	FF	0383	00	03B7	00	03EB	00
031C	FF	0350	FF	0384	00	03B8	00	03EC	00
031D	FF	0351	FF	0385	00	03B9	00	03ED	00
031E	FF	0352	FF	0386	00	03BA	00	03EE	00
031F	FF	0353	FF	0387	00	03BB	00	03EF	00
0320	FF	0354	FF	0388	00	03BC	00	03F0	00
0321	FF	0355	FF	0389	00	03BD	00	03F1	00
0322	FF	0356	FF	038A	00	03BE	00	03F1	00
0323	FF	0357	FF	038B	00	03BF	00	03F2	00
0324	FF	0358	FF	038C	00	03C0	00	03F3	00
0325	FF	0359	FF	038D	00	03C1	00	03F4	00
0326	FF	035A	FF	038E	00	03C2	00	03F5	00
0327	FF	035B	FF	038F	00	03C3	00	03F6	00
0328	FF	035C	FF	0390	00	03C4	00	03F7	00
0329	FF	035D	FF	0391	00	03C5	00	03F8	00
032A	FF	035E	FF	0392	00	03C6	00	03F9	00
032B	FF	035F	FF	0393	00	03C7	00	03FA	00
032C	FF	0360	FF	0394	00	03C8	00	03FB	00
032D	FF	0361	FF	0395	00	03C9	00	03FC	00
032E	FF	0362	FF	0396	00	03CA	00	03FD	00
032F	FF	0363	FF	0397	00	03CB	00	03FE	00
0330	FF	0364	FF	0398	00	03CC	00	03FF	00
0331	FF	0365	FF	0399	00	03CD	00		
0332	FF	0366	FF	039A	00	03CE	00		
0333	FF	0367	FF	039B	00	03CF	00		

Table 4.6: Software Look-up Table For Ramp

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0100	7F	0134	99	0168	B3	019C	CD	01D0	E7
0101	80	0135	9A	0169	B4	019D	CE	01D1	E8
0102	80	0136	9A	016A	B4	019E	CE	01D2	E8
0103	81	0137	9B	016B	B5	019F	CF	01D3	E9
0104	81	0138	9B	016C	B5	01A0	CF	01D4	E9
0105	82	0139	9C	016D	B6	01A1	D0	01D5	EA
0106	82	013A	9C	016E	B6	01A2	D0	01D6	EA
0107	83	013B	9D	016F	B7	01A3	D1	01D7	EB
0108	83	013C	9D	0170	B7	01A4	D1	01D8	EB
0109	84	013D	9E	0171	B8	01A5	D2	01D9	EC
010A	84	013E	9E	0172	B8	01A6	D2	01DA	EC
010B	85	013F	9F	0173	B9	01A7	D3	01DB	ED
010C	85	0140	9F	0174	B9	01A8	D3	01DC	ED
010D	86	0141	A0	0175	BA	01A9	D4	01DD	EE
010E	86	0142	A0	0176	BA	01AA	D4	01DE	EE
010F	87	0143	A1	0177	BB	01AB	D5	01DF	EF
0110	87	0144	A1	0178	BB	01AC	D5	01E0	EF
0111	88	0145	A2	0179	BC	01AD	D6	01E1	F0
0112	88	0146	A2	017A	BC	01AE	D6	01E2	F0
0113	89	0147	A3	017B	BD	01AF	D7	01E3	F1
0114	89	0148	A3	017C	BD	01B0	D7	01E4	F1
0115	8A	0149	A4	017D	BE	01B1	D8	01E5	F2
0116	8A	014A	A4	017E	BE	01B2	D8	01E6	F2
0117	8B	014B	A5	017F	BF	01B3	D9	01E7	F3
0118	8B	014C	A5	0180	BF	01B4	D9	01E8	F3
0119	8C	014D	A6	0181	C0	01B5	DA	01E9	F4
011A	8C	014E	A6	0182	C0	01B6	DA	01EA	F4
011B	8D	014F	A7	0183	C1	01B7	DB	01EB	F5
011C	8D	0150	A7	0184	C1	01B8	DB	01EC	F5
011D	8E	0151	A8	0185	C2	01B9	DC	01ED	F6
011E	8E	0152	A8	0186	C2	01BA	DC	01EE	F6
011F	8F	0153	A9	0187	C3	01BB	DD	01EF	F7
0120	8F	0154	A9	0188	C3	01BC	DD	01F0	F7
0121	90	0155	AA	0189	C4	01BD	DE	01F1	F8
0122	90	0156	AA	018A	C4	01BE	DE	01F2	F8
0123	91	0157	AB	018B	C5	01BF	DF	01F3	F9
0124	91	0158	AB	018C	C5	01C0	DF	01F4	F9
0125	92	0159	AC	018D	C6	01C1	E0	01F5	FA
0126	92	015A	AC	018E	C6	01C2	E0	01F6	FA
0127	93	015B	AD	018F	C7	01C3	E1	01F7	FB
0128	93	015C	AD	0190	C7	01C4	E1	01F8	FB
0129	94	015D	AE	0191	C8	01C5	E2	01F9	FC
012A	94	015E	AE	0192	C8	01C6	E2	01FA	FC
012B	95	015F	AF	0193	C9	01C7	E3	01FB	FD
012C	95	0160	AF	0194	C9	01C8	E3	01FC	FD
012D	96	0161	B0	0195	CA	01C9	E4	01FD	FE
012E	96	0162	B0	0196	CA	01CA	E4	01FE	FE
012F	97	0163	B1	0197	CB	01CB	E5	01FF	7F
0130	97	0164	B1	0198	CB	01CC	E5		
0131	98	0165	B2	0199	CC	01CD	E6		
0132	98	0166	B2	019A	CC	01CE	E6		
0133	99	0167	B3	019B	CD	01CF	E7		

Table 4.7: Software Look-up Table For Staircase

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0400	8F	0434	9F	0468	BF	049C	CF	04D0	EF
0401	8F	0435	9F	0469	BF	049D	CF	04D1	EF
0402	8F	0436	9F	046A	BF	049E	CF	04D2	EF
0403	8F	0437	9F	046B	BF	049F	CF	04D3	EF
0404	8F	0438	9F	046C	BF	04A0	DF	04D4	EF
0405	8F	0439	9F	046D	BF	04A1	DF	04D5	EF
0406	8F	043A	9F	046E	BF	04A2	DF	04D6	EF
0407	8F	043B	9F	046F	BF	04A3	DF	04D7	EF
0408	8F	043C	9F	0470	BF	04A4	DF	04D8	EF
0409	8F	043D	9F	0471	BF	04A5	DF	04D9	EF
040A	8F	043E	9F	0472	BF	04A6	DF	04DA	EF
040B	8F	043F	9F	0473	BF	04A7	DF	04DB	EF
040C	8F	0440	AF	0474	BF	04A8	DF	04DC	EF
040D	8F	0441	AF	0475	BF	04A9	DF	04DD	EF
040E	8F	0442	AF	0476	BF	04AA	DF	04DE	EF
040F	8F	0443	AF	0477	BF	04AB	DF	04DF	EF
0410	8F	0444	AF	0478	BF	04AC	DF	04E0	FF
0411	8F	0445	AF	0479	BF	04AD	DF	04E1	FF
0412	8F	0446	AF	047A	BF	04AE	DF	04E2	FF
0413	8F	0447	AF	047B	BF	04AF	DF	04E3	FF
0414	8F	0448	AF	047C	BF	04B0	DF	04E4	FF
0415	8F	0449	AF	047D	BF	04B1	DF	04E5	FF
0416	8F	044A	AF	047E	BF	04B2	DF	04E6	FF
0417	8F	044B	AF	047F	BF	04B3	DF	04E7	FF
0418	8F	044C	AF	0480	CF	04B4	DF	04E8	FF
0419	8F	044D	AF	0481	CF	04B5	DF	04E9	FF
041A	8F	044E	AF	0482	CF	04B6	DF	04EA	FF
041B	8F	044F	AF	0483	CF	04B7	DF	04EB	FF
041C	8F	0450	AF	0484	CF	04B8	DF	04EC	FF
041D	8F	0451	AF	0485	CF	04B9	DF	04ED	FF
041E	8F	0452	AF	0486	CF	04BA	DF	04EE	FF
041F	8F	0453	AF	0487	CF	04BB	DF	04EF	FF
0420	9F	0454	AF	0488	CF	04BC	DF	04F0	FF
0421	9F	0455	AF	0489	CF	04BD	DF	04F1	FF
0422	9F	0456	AF	048A	CF	04BE	DF	04F2	FF
0423	9F	0457	AF	048B	CF	04BF	DF	04F3	FF
0424	9F	0458	AF	048C	CF	04C0	EF	04F4	FF
0425	9F	0459	AF	048D	CF	04C1	EF	04F5	FF
0426	9F	045A	AF	048E	CF	04C2	EF	04F6	FF
0427	9F	045B	AF	048F	CF	04C3	EF	04F7	FF
0428	9F	045C	AF	0490	CF	04C4	EF	04F8	FF
0429	9F	045D	AF	0491	CF	04C5	EF	04F9	FF
042A	9F	045E	AF	0492	CF	04C6	EF	04FA	FF
042B	9F	045F	AF	0493	CF	04C7	EF	04FB	FF
042C	9F	0460	BF	0494	CF	04C8	EF	04FC	FF
042D	9F	0461	BF	0495	CF	04C9	EF	04FD	FF
042E	9F	0462	BF	0496	CF	04CA	EF	04FE	FF
042F	9F	0463	BF	0497	CF	04CB	EF	04FF	FF
0430	9F	0464	BF	0498	CF	04CC	EF		
0431	9F	0465	BF	0499	CF	04CD	EF		
0432	9F	0466	BF	049A	CF	04CE	EF		
0433	9F	0467	BF	049B	CF	04CF	EF		

Table 4.8: Software Look-up Table For Random Wave

Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code	Memory Address	Hex code
0500	7F	0529	9F	0560	AF	0597	CF	05D4	CF
0501	00	052A	9F	0561	AF	0598	CF	05D5	CF
0502	00	052B	9F	0562	AF	0599	CF	05D6	CF
0503	00	052C	9F	0563	AF	05A0	CF	05D7	CF
0504	00	052D	9F	0564	AF	05A1	CF	05D8	CF
0505	00	052E	9F	0565	AF	05A2	CF	05D9	CF
0506	00	052F	9F	0566	AF	05A3	CF	05DA	CF
0507	00	0530	9F	0567	AF	05A4	CF	05DB	CF
0508	00	0531	9F	0568	AF	05A5	CF	05DC	CF
0509	00	0532	9F	0569	AF	05A6	CF	05DD	CF
050A	00	0533	9F	056A	AF	05A7	CF	05DE	CF
050B	00	0534	9F	056B	AF	0A58	CF	05DF	CF
050C	00	0535	9F	056C	AF	05A9	CF	05E0	CF
050D	00	0536	9F	056D	AF	05AA	CF	05E1	CF
050F	00	0537	9F	056E	AF	05AB	CF	05E2	CF
0510	00	0538	9F	056F	AF	05AC	CF	05E3	CF
0511	00	0539	9F	0570	AF	05AD	CF	0E54	CF
0512	00	053A	9F	0571	AF	05AE	CF	05E5	CF
0513	00	053B	9F	0572	AF	05AF	CF	05E6	CF
0514	00	053C	9F	0573	AF	05B0	CF	05E7	CF
0515	00	053D	9F	0574	AF	05B1	CF	05E8	CF
0516	00	053E	9F	0575	AF	05B2	CF	0E59	CF
0517	00	053F	9F	0576	AF	05B3	CF	05EA	CF
0518	00	0540	AF	0577	AF	05B4	CF	05EB	CF
0519	00	0541	AF	0578	AF	05B5	CF	05EC	CF
051A	00	0542	AF	0579	AF	05B6	CF	05ED	CF
051B	00	0543	AF	057A	AF	05B7	CF	05EE	CF
051C	00	0544	AF	057B	AF	05B8	CF	05EF	CF
051D	00	0545	AF	057C	AF	05B9	CF	05F0	CF
051E	00	0546	AF	057D	AF	05BA	CF	05F1	CF
051F	00	0547	AF	057E	AF	05BB	CF	05F2	CF
0520	9F	0548	AF	057F	BF	05BC	CF	05F3	CF
0521	9F	0549	AF	0580	CF	05BD	CF	05F4	CF
0522	9F	054A	AF	0581	CF	05BE	CF	05F5	CF
0523	9F	054B	AF	0582	CF	05BF	CF	05F6	CF
0524	9F	054C	AF	0583	CF	05C0	CF	05F7	CF
0525	9F	054D	AF	0584	CF	05C1	CF	05F8	CF
0526	9F	054E	AF	0585	CF	05C2	CF	05F9	CF
0527	9F	054F	AF	0586	CF	05C3	CF	05FA	CF
0528	9F	0550	AF	0587	CF	05C4	CF	05FB	CF
0529	9F	0551	AF	0588	CF	05C5	CF	05FC	CF
052A	9F	0552	AF	0589	CF	05C6	CF	05FD	CF
052B	9F	0553	AF	058A	CF	05C7	CF	05FE	FF
052C	9F	0554	AF	058B	CF	05C8	CF	05FE	FF
052D	9F	0555	AF	058C	CF	05C9	CF		
052E	9F	0556	AF	058D	CF	05CA	CF		
052F	9F	0557	AF	058E	CF	05CB	CF		
0521	9F	0558	AF	058F	CF	05CC	CF		
0522	9F	0559	AF	0590	CF	05CD	CF		
0523	9F	055A	AF	0591	CF	05CE	CF		
0524	9F	055B	AF	0592	CF	05CF	CF		

4.5 Frequency Generation

Changing the rate at which the desired waveform bits were sent to the DAC was used in generating the different frequencies. Since each instruction requires different execution times, then all the instructions contributing to the delay were taken into account (see Table 4.9) and also Appendix A for the wave generation software.

Table 4.9: Software generation of frequency of 1Hz

Label	Instruction	Number of clock cycles
NEXT :	MOV AL,BYTE PTR CS:[SI]	10
	OUT PPIB,AL	10
	PUSH CX	20
	CALL DELAY1	19
	POP CX	16
	INC SI	2
	LOOP NEXT	15
DELAY1:	IN AL,PPIC	10
	CMP AL, 01H	4
	JE FREQ1	16
FREQ1:	MOV CX, 363	8
DELAY2:	PUSH CX	20
	POP CX	16
	LOOP DELAY2	15
	RET	15

To generate higher frequencies, the number of loops were decreased accordingly.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 A Working System And Analysis Of Waveforms Generated

The results presented in this research work can be divided into:

- A working hardware and software and
- Analysis of the waveforms generated

The waveforms generated are shown in Figure 5.1 through Figure 5.5

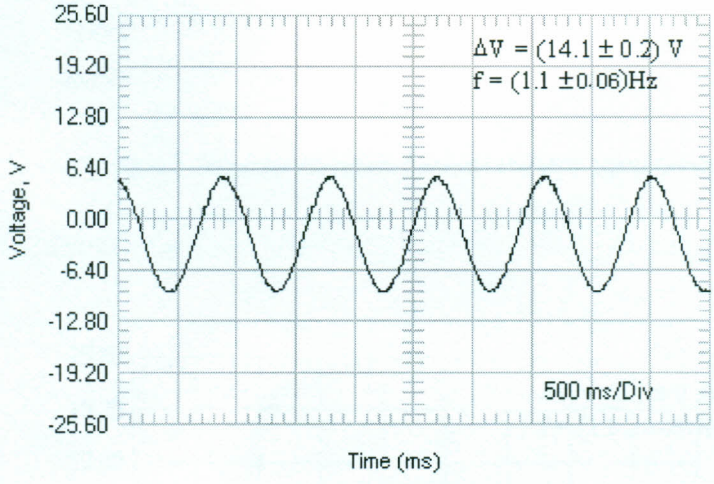


Fig. 5.1: Sine Waveform Generated by MCM Signal Generator

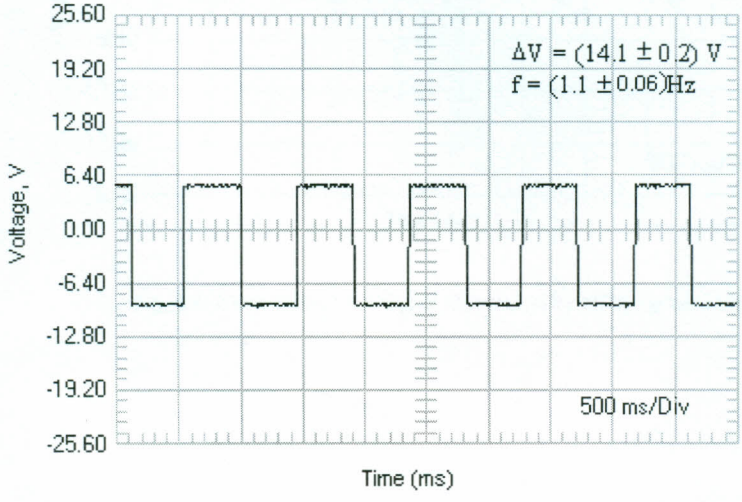


Fig. 5.2: Square Waveform Generated by MCM Signal Generator

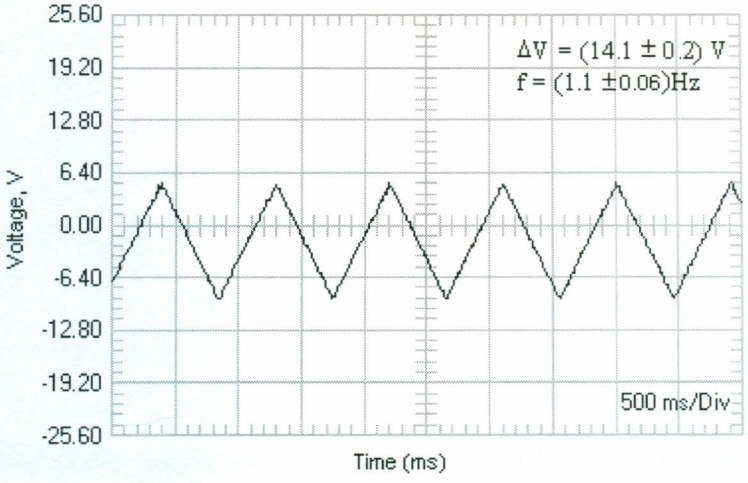


Fig. 5.3: Triangular Waveform Generated by MCM Signal Generator

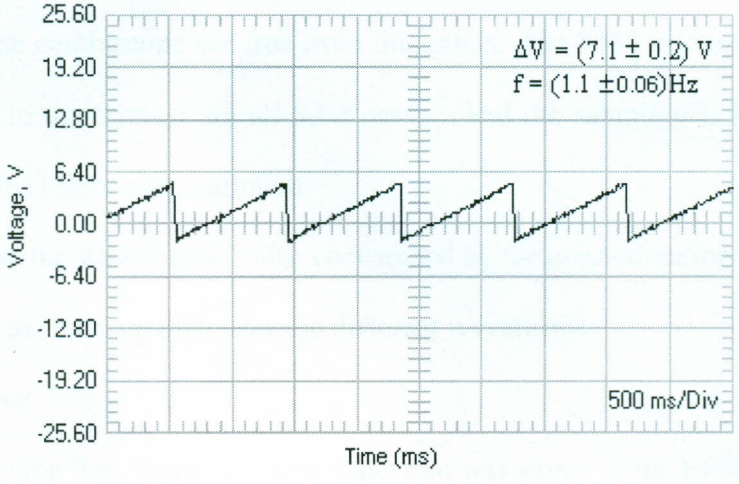


Fig. 5.4: Ramp Waveform Signal Generated by MCM Signal Generator

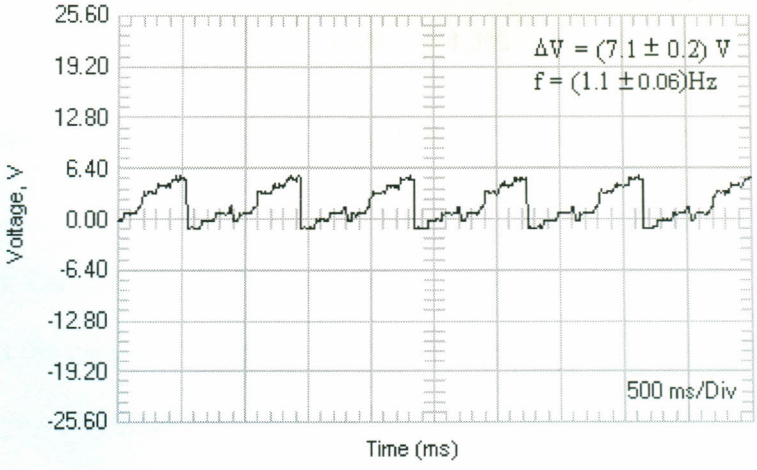


Fig. 5.5: Staircase Waveform Generated by MCM Signal Generator

The waveforms can only be displayed in the computer in digital mode and therefore the digitization effect of these waveforms by the PM 3384A autoranging combiscope that was set in digital mode contributed to the errors displayed by the above waveforms. Otherwise the waveforms from this MBM signal generator as displayed in analogue mode by the combiscope are free from this effect. The 8088 microprocessor and DAC 0800 used in this system are all 8-bit devices and the sampling is based on them and therefore this offers less resolution.

The noise in the waveforms is also contributed by the approximation error arising from generating the look-up tables for the different waveforms.

For example:

To generate the first byte of the sine wave that was stored in the EPROM, the following approximation was carried out:

Actual value = 3.136 After rounding the value becomes = 3.0

$$\%\text{Error} = \frac{(\sin\theta \times 128)\,\text{Round off}}{(\sin\theta \times 128)} \times 100 \quad = 4.3\%$$

Therefore the error in 80H the first byte in the look-up Table 4.4 is 4.3%.

5.2 Analysis Of The Generated Frequencies

Table 5.1 shows the expected frequencies, observed Frequencies obtained from PM 3384A autoranging combiscope (Technique No.1) and the dispersion (f_{Di}).

$$\frac{\sigma_f}{f} = \left[a^2 \left(\frac{\sigma_t}{t} \right)^2 \right]^{1/2} \tag{5.2}$$

The errors in the observed frequencies were calculated as follows [49,50]

$$F = T^{-1} \text{ (relationship between frequency and periodic time)} \tag{5.1}$$

Where f is the observed frequency, σ_f is the error in the observed frequency, σ_t is the error in the periodic time, t is the periodic time and a= -1 (power of periodic time)

Table 5.1: Frequencies generated by MBM Signal Generator

Expected Frequencies (Hz) (f_{ei})	Observed Frequencies (Hz) (f_{oi})	F_{DI} %
1	1.1 ± 0.06	10
10	9.6 ± 0.46	-4
30	32.5 ± 0.16	8.3
40	41.7 ±1.73	4.3
50	52.1 ± 2.71	4.2

The dispersion f_{Di} of the mean observed frequencies f_{oi} from the expected frequencies f_{ei} are evaluated as follows [51]

$$f_{Di} = \left(\frac{f_{oi} - f_{ei}}{f_{ei}} \right) x100\% \qquad (5.3$$

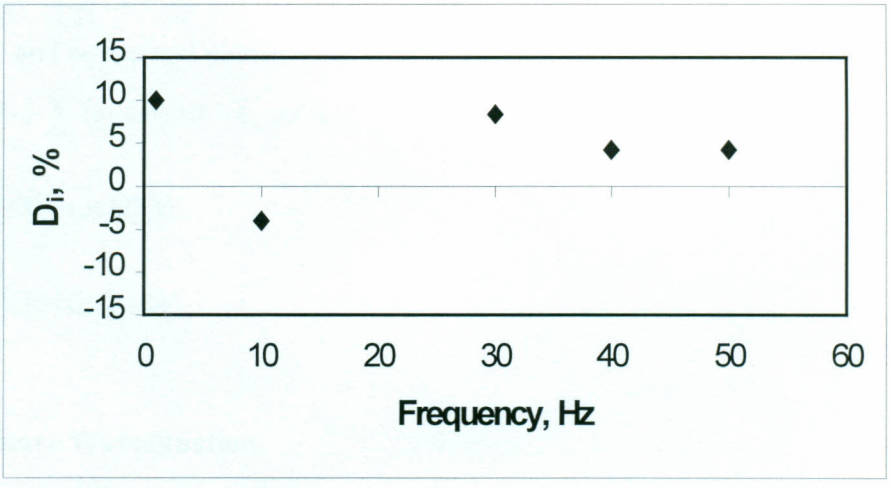


Fig. 5.6: Dispersion of the mean observed frequencies from the expected frequencies.

The deviation of the observed frequency (1.1Hz) from the expected frequency (1Hz) is contributed to the approximation error in evaluating the delay required for a particular frequency.

For example:

The delay required to achieve a frequency of 1 Hz is

Delay = 362.737 which can only be evaluated by the 8088 microprocessor after rounding to integral value, Delay = 363, introducing and error of 0.073%.

5.3 Spectral Analysis

Finally, all the waveforms generated are periodic and thus they were Fourier analysed. They can be represented in the interval *(–l, l)* by the infinite series of pure sine and cosine functions given as follows:

$$f(x) = \frac{a_o}{2} + \sum_{k=1}^{\infty}\left[a_k \cos\left(\frac{k\pi x}{l}\right) + b_k \sin\left(\frac{k\pi x}{l}\right)\right] \qquad (5.4)$$

where a_k and b_k are real coefficients independent of x which are obtained from

$$f(\theta) = \frac{a_o}{2} + \sum_{k=1}^{\infty}(a_k \cos k\theta + b_k \sin k\theta)$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\theta)\cos k\theta d\theta$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\theta)\sin k\theta d\theta$$

5.3.1 Square Wavefunction

$$f(\theta) = A \qquad 0 \le \theta \le \pi$$

Fourier series:

$$v = 7.05\left[\frac{1}{2} + \frac{2}{\pi}\sin\omega t + \frac{2}{3\pi}\sin 3\omega t + --------- + \frac{2}{n\pi}\sin(n\omega t)\right]$$

Where n = odd (1, 3, 5, 7, --------∞)

5.3.2 Ramp

$$f(\theta) = \frac{7.1}{2\pi}\theta \qquad 0 \le \theta \le 2\pi$$

Fourier series:

$$v = 7.1\left[\frac{1}{2} + \frac{1}{\pi}\sin\omega t - \frac{1}{2\pi}\sin 2\omega t + \frac{1}{3\omega}\sin 3\omega t - \frac{1}{4\pi}\sin^4 \omega t + ------ - \left(\frac{-1}{n\pi}\right)^n \sin(n\omega t)\right]$$

Where: n = 1, 2, 3, ----------, ∞

5.3.3 Triangular Wave

Fourier series:

$$v = 7.05\left[\frac{1}{2} + \frac{4\cos\omega t}{\pi^2} + \frac{4\cos 3\omega t}{9\pi^2} + - - - - - - - + \frac{4\cos(n\omega t)}{n^2\pi^2}\right]$$

where : n = odd integer (1, 3, 5, -------, ω)

CHAPTER 6

CONCLUSIONS

6.1 Summary and Recommendations

The research project aimed at designing, constructing and carrying out performance evaluations of a microprocessor controlled multifunction signal generator.Standard waveforms such as sine wave, square wave, triangular wave, ramp and other pre-stored waveforms can be generated. The MBM signal generator has also been observed to generate stable signals of high frequency stability and is flexible in waveform and frequency selections. The amplitudes of the signals range from ±4.8V to ±14.1V peak to peak while the programmed frequencies range from 1 Hz to 100Hz. Therefore, the objectives of this research work were achieved.

This high performance MBM signal generator can be used in universities and college laboratories for experimental work, in control systems and fault finding in industrial setting.

6.2 Suggestions for Further Work

Some of the further work that can be carried out include: generation of the waveforms at higher frequencies using a fast microprocessor ranging from 80486DX-Pentium III processors which have in-built math co-processors and cache memory. The math co-processors can be used to perform the necessary arithmetic involving the different waveforms whose solutions can be output to the DAC for conversion. Cache memory would speed up the memory read and write cycles. The resolution can be improved

further by using 16-bit DAC rather than an 8-bit DAC. Another improvement would be to include a keypad and a display system for entering and displaying the desired frequency. The MBM signal generator should perform best on a printed circuit board rather than on breadboards which are prone to stray capacitances and inductances which reduce the system performance. Finally, interfacing this system to a computer would provide better waveform, frequency and amplitude selections and printing of the waveforms generated for analysis.

REFERENCES

1. PC Instruments Datasheet: PC Instruments Ltd., USA, 1999.
2. Malcolm G.: "Analogue electronics analysis and design", pg. 265-283, Macmillan press Ltd., London, 1990.
3. Sodof, S.: "Applications of analogue integrated circuits" , Prentice-Hall, Englewood Cliffs, NJ, 1995.
4. Gudefin, E.J., and Louis, J.P.: "Non-linear Analysis of an analogue AM signal synthesizer", Intl. Conf. Ind. Electron. Contr., Vienna, Part II, no. 1-25, 1991.
5. Hodges, D.A., and H.G. Jackson: "Analysis and Design of digital Integrated circuits", McGraw-Hill Book Company, NewYork, 1993.
6. Taub, H., and D. Schilling: "Digital Integrated Electronics", McGraw-Hill Book Company, NewYork, 1997.
7. Mclaren, S.G. : "Direct Digital Control of signal sources", IFAC-symposium, Duesseldorf, 1992.
8. Subbarrao, W.V. : "The 8086/8088 family of microprocessors software and system applications", Delmar Publishers Inc, U.S.A, 1992.
9. Parzon., B. : "Design of crystal and other harmonic oscillators ", Wiley, NewYork, 1983.
10. Ahsan S.I. : "Microprocessors with applications in process control", McGraw-Hill publishing company , New Delhi , 1990.
11. Gopal M. : "Digital control engineering" , Wiley Eastern LTD, New Delhi, 1989.

12. Malvino, P.A. : "Digital computer electronics. An introduction to microcomputers", 2nd edition ; McGraw-Hill publishing company , New Delhi , 1992.

13. Rodnay Z. : "Programming the Z80", Sybex Inc., U.S.A, 1982.

14. Mathur, A.P. : "Introduction to Microprocessors 3rd edition", McGraw-Hill, U.S.A, 1994.

15. Kainakov, G.P. and Duchen L.S. : "A microcomputer system for lifetime measurement" , Journal of physics E, Scientific instruments, vol.17-No.2,98,1984.

16. Garratt, J. : "Design and Technology", Cambridge University Press, U.S.A 1995.

17. Intel corporation: "iAPX 8088 manual", Santa clara 1981.

18. Osborne, A.: "An introduction to microcomputers", Vol. 1, McGraw-Hill, London, 1987.

19. Mano, M.: "Computer Systems Architecture", 2nd Edition", John Wiley and Sons, NewYork, 1992.

20. Gaonkar, R.S. : "Microprocessor Architecture, programming, and applications with the 8085 and 8080A", Wiley Eastern Limited, South East Asia, 1989.

21. Hall, D.V. : "Microprocessors and Interfacing, Programming and Hardware, 2nd Edition", Macmillan, U.S.A, 1992.

22. Milne J.S. and Fraser C.J. : "Microcomputer Applications in measurement systems",

Macmillan Education LTD, 1990.

23. Downtown A.C.: "Computers and Microprocessors 3rd edition", Chapman and Hall, 1992.

24. Yu-Cheng L. and Gibson A.G. : "Microcomputer Systems, the 8086/8088 family, Architecture, Programming and Design 2nd edition", Prentice-Hall of India, New Delhi, 1997.

25. Owitti A.O. : "Microprocessor based maximum power measurement for a photovoltaic source utilizing a resistor load", MSc. Thesis, Moi University, 1996.

26. Owade M. : "Design and development of a programmable laboratory interface system with an illustrative use in resistivity temperature experiment", MSc. Thesis, Kenyatta University, 1998.

27. Ahson S.I. : Microprocessor in process control", computer and electrical engineering, Vol. 7, 1980.

28. Hammond, P.H. and King P.J. : "A prospect for industrial control, Electronic and power, 1980.

29. Proceedings of IEEE-Special Issue on microprocessor applications. Vol. 66, 1980.

30. Stemmler, H.: "Digital signal processing and control methods in power and communication Electronics, Dusseldorf, Paper no. E-23, 1994.

31. Meade,M.I. and Dilton C.: " Signals and systems" , pg. 10-15, Van Nostrad, London, 1995.

32. Baher H.: "Analogue and digital signal processing " , John Wiley and sons, NewYork, 1986.

33. Lathi, B.P. : "Signals and Systems" , Berkeley Cambridge press, Carmichael, 1987.

34. Linsley,.T : "Electronics for electricians and service engineers", 2nd Ed., Bath Press, UK, 1993.

35. Schilling, D., and Belove C.: "Electronic Circuits : Discrete and Integrated", McGraw-Hill Book Company, New York, 1995.

36. Ghausi, M.S. : "Electronics Devices and Circuits: Discrete and Integrated", Holt, NewYork, 1996.

37. Hall, B.D. : "A general purpose interactive programmable laboratory interface system using the IEEE-488 Bus", computers in physics journal, vol.5-No.7,1991.

38. Taub, H. : "Digital circuits and Microprocessors", McGraw-Hill Book Company, New York, 1995.

39. Peatman, J.B.: "Design of Digital Systems", 2nd Edition, McGraw-Hill Book Company, NewYork, 1991.

40. Peatman, J.B.: "Microcomputer based design", McGraw-Hill, New York, 1977.

41. Savitch, W. : "Problem solving with C++ the object of programming 2nd edition", Addison-Wesley Longman, Inc, U.S.A, 1999.

42. Abel, P.: " IBM PC assembly language programming 2nd edition", Prentice Hall International, Inc, U.S.A, 1992.

43. Dataman company, Omnipro 2 Eprom programmer, USA, 1995.

44. Siebert, W.M. "Circuits, Signals, and Systems", MIT Press, Cambridge, 1986.

45. Texas Instruments: "The TTL Data book for Design Engineers", 2nd edition, U.S.A 1980.

46. RS Data Library. : RS components Ltd. Corby, 1999.

47. MCS 85 Users manual: Intel corporation, Santa Clara, 1978.

48. Green, D. C. : "Electronics II 4th edition", Longman Group, Hong Kong, 1989.

49. Taylor, J.R. : "An introduction to error analysis 2nd edition, University science books",1997.

50. Squires, G.L. : "Practical Physics", Mc Graw-Hill, U.S.A, 1968.

51. Crawshaw J. and Chambers J. : "A concise course in A-level statistics 3rd edition, Stanley Thornes Publishers, UK, 1994.

APPENDIX A

PROGRAM LISTING

The complete control software for the MBM signal generator is listed as follows:
Microsoft (R) Macro Assembler Version 5.10

```
;PROGRAMMER      : Karimi P. Mwangi
;PROGRAM TITLE   : Waveform generation Software (wavef.asm)  Date:28/04/2000
;ABSTRACT        : This program generates the standard waveforms and random
;                  waveforms. The user selects the desired waveform at port A and
;                   selects the desired frequency at port C. The signal bits are
;                   transmitted to port B of the 8255A PPI which are then
                   converted to analogue form by digital to analogue converter
                   (DAC) and then output to the PM 3384 autoranging
                   combiscope.
;REGISTERS USED : AL, CX, SP, CS, IP, SS, DS, ES
;PORTS           : Port A (00H) – input port, Port B (01H) – output port
;                : Port C (10H) – input port
;MEMORY          : 00000H~FE7FFH (Unused), FE800H~FEFFFH (RAM- SS)
;                : FF000H~FFFFFH (EPROM- DS, CS, and ES)

0000                    SSEG    SEGMENT
= 0100                          STACK EQU 100H
0000                    SSEG ENDS

0000                    CODE    SEGMENT
                              ASSUME  CS:CODE,DS:CODE,ES:CODE
                                ;
= 0000                      PPIA      EQU    00H   ;define port address
= 0001                      PPIB      EQU    01H
= 0002                      PPIC      EQU    02H
= 0003                      PPIC_C    EQU    03H
                              ;
0000  90            INIT:     NOP
0001  90                      NOP
0002  90                      NOP
0003  90                      NOP
0004  90                      NOP
0005  90                      NOP
0006  B8 FE80                MOV     AX,0FE80H
0009  8E D0                   MOV     SS,AX          ;initialise stack segment to
                                                     ;start at RAM's first
                                                     memory ;location
```

```
000B  B8 FF00                MOV    AX,0FF00H
000E  8E D8                  MOV    DS,AX          ;initialise DS to start at
                                                   ;ROM's first memory location
0010  BC 0100                MOV    SP, STACK      ;initialise SP to start at
                                                   offset  ;100H of the SS
          0013  B0 99                MOV    AL,10011001B
0015  E6 03                  OUT    PPIC_C,AL      ;configure port A as
                                                   ;input,;port B as output,
                                                   ;port C as input
                             ;
0017  E4 00            POLL: IN     AL,PPIA        ;select the desired waveform
0019  3C 01                  CMP    AL,01H
001B  74 16                  JE     SINE1
001D  3C 02                  CMP    AL,02
001F  74 2A                  JE     SQR1
0021  3C 04                  CMP    AL,04H
0023  74 3E                  JE     TRIA1
0025  3C 08                  CMP    AL,08H
0027  74 52                  JE     RAMP1
0029  3C 10                  CMP    AL,10H
002B  74 66                  JE     STAIR1
002D  3C 20                  CMP    AL,20H
002F  74 7B                  JE     RANDO1
0031  EB E4                  JMP    POLL
                             ;
0033  B9 0000                SINE1: MOV    CX,0
0036  B9 0100                       MOV    CX,256
0039  BE 012B R              MOV    SI,OFFSET SINE2
003C  2E: 8A 04        NEXT : MOV    AL,BYTE PTR CS:[SI]
003F  E6 01                  OUT    PPIB,AL        ; output the sine  wave
                                                   ; bits at port B
0041  51                     PUSH   CX
0042  E8 00C5 R              CALL   DELAY1
0045  59                     POP    CX
0046  46                     INC    SI
0047  E2 F3                  LOOP   NEXT
0049  EB CC                  JMP    POLL
004B  B9 0000                SQR1:  MOV    CX,0
004E  B9 0100                       MOV    CX,256
0051  BE 022B R              MOV    SI,OFFSET SQR2
0054  2E: 8A 04       NEXT1 : MOV    AL,BYTE PTR CS:[SI]
0057  E6 01                  OUT    PPIB,AL  ; output the square wave bits at
                                              ; port B
0059  51                     PUSH   CX
```

```
005A  E8 00C5 R                  CALL   DELAY1
005D  59                         POP    CX
005E  46                         INC    SI
005F  E2 F3                      LOOP   NEXT1
0061  EB B4                      JMP    POLL
                                 ;
0063  B9 0000                    TRIA1: MOV    CX,0
0066  B9 0080                           MOV    CX,256
0069  BE 032B R                  MOV    SI,OFFSET TRIA2
006C  2E: 8A 04          NEXT2 : MOV    AL,BYTE PTR CS:[SI]
006F  E6 01                      OUT    PPIB,AL  ; output the triangular wave
                                                  ; bits at port B
0071  51                         PUSH   CX
0072  E8 00C5 R                  CALL   DELAY1          ;
0075  59                         POP    CX
0076  46                         INC    SI
0077  E2 F3                      LOOP   NEXT2
0079  EB 9C                      JMP    POLL
                                 ;
007B  B9 0000            RAMP1:  MOV    CX,0
007E  B9 0100                    MOV    CX,256
0081  BE 03AB R                  MOV    SI,OFFSET RAMP2
0084  2E: 8A 04          NEXT3 : MOV    AL,BYTE PTR CS:[SI]
0087  E6 01                      OUT    PPIB,AL        ;output the ramp wave
                                                        ; bits at port B
0089  51                         PUSH   CX
008A  E8 00C5 R                  CALL   DELAY1
008D  59                         POP    CX
008E  46                         INC    SI
008F  E2 F3                      LOOP   NEXT3
0091  EB 84                      JMP    POLL
                                 ;
0093  B9 0000            STAIR1: MOV    CX,0
0096  B9 0100                    MOV    CX,256
0099  BE 04AB R                  MOV    SI,OFFSET STA2
009C  2E: 8A 04          NEXT4 : MOV    AL,BYTE PTR CS:[SI]
009F  E6 01                      OUT    PPIB,AL        ; output the square wave
                                                        ; bits at port B
00A1  51                         PUSH   CX
00A2  E8 00C5 R                  CALL   DELAY1
00A5  59                         POP    CX
00A6  46                         INC    SI
00A7  E2 F3                      LOOP   NEXT4
00A9  E9 0017 R                  JMP    POLL
```

```
                                    ;
00AC  B9 0000        RANDO1:        MOV    CX,0
00AF  B9 0100                       MOV    CX,256
00B2  BE 05AB R                     MOV    SI,OFFSET RAND2
00B5  2E: 8A 04      NEXT5 :        MOV    AL,BYTE PTR CS:[SI]
00B8  E6 01                         OUT    PPIB,AL        ; output the random wave
                                                           ;bits at port B
00BA  51                            PUSH   CX
00BB  E8 00C5 R                     CALL   DELAY1
00BE  59                            POP    CX
00BF  46                            INC    SI
00C0  E2 F3                         LOOP   NEXT5
00C2  E9 0017 R                     JMP    POLL
                                    ;
00C5  E4 02          DELAY1:        IN     AL,PPIC    ;select the desired frequency
00C7  3C 01                         CMP    AL,01H
00C9  74 1E                         JE     FREQ1
00CB  3C 02                         CMP    AL,02H
00CD  74 22                         JE     FREQ2
00CF  3C 04                         CMP    AL,04H
00D1  74 26                         JE     FREQ3
00D3  3C 08                         CMP    AL,08H
00D5  74 2A                         JE     FREQ4
00D7  3C 10                         CMP    AL,10H
00D9  74 2E                         JE     FREQ5
00DB  3C 20                         CMP    AL,20H
00DD  74 32                         JE     FREQ6
00DF  3C 40                         CMP    AL,40H
00E1  74 36                         JE     FREQ7
00E3  3C 80                         CMP    AL,80H
00E5  74 3A                         JE     FREQ8
00E7  EB DC                         JMP    DELAY1
                                    ;
00E9  B9 016B          FREQ1 :      MOV    CX,363
00EC  51               DELAY2 :     PUSH   CX
00ED  59                            POP    CX
00EE  E2 FC                         LOOP   DELAY2
00F0  C3                            RET
                                    ;
00F1  B9 002A          FREQ2 :      MOV    CX,42
00F4  51               DELAY4 :     PUSH   CX
00F5  59                            POP    CX
00F6  E2 FC                         LOOP   DELAY4
00F8  C3                            RET
```

```
                                ;
00F9  B9 0006       FREQ3 :     MOV    CX,6
00FC  51            DELAY6 :    PUSH   CX
00FD  59                        POP    CX
00FE  E2 FC                     LOOP   DELAY6
0100  C3                        RET
                                ;
0101  B9 0005       FREQ4 :     MOV    CX,5
0104  51            DELAY8 :    PUSH   CX
0105  59                        POP    CX
0106  E2 FC                     LOOP   DELAY8
0108  C3                        RET
                                ;
0109  B9 0004       FREQ5 :     MOV    CX,4
010C  51            DELAY10 :   PUSH   CX
010D  59                        POP    CX
010E  E2 FC                     LOOP   DELAY10
0110  C3                        RET
                                ;
0111  B9 0003       FREQ6 :     MOV    CX,3
0114  51            DELAY13 :   PUSH   CX
0115  59                        POP    CX
0116  E2 FC                     LOOP   DELAY13
0118  C3                        RET
                                ;
0119  B9 0002       FREQ7 :     MOV    CX,2
011C  51            DELAY15:    PUSH   CX
011D  59                        POP    CX
011E  E2 FC                     LOOP   DELAY15
0120  C3                        RET
                                ;
0121  B9 0001       FREQ8 :     MOV    CX,1
0124  51            DELAY17:    PUSH   CX
0125  90                        NOP
0126  90                        NOP
0127  59                        POP    CX
0128  E2 FA                     LOOP   DELAY17
012A  C3                        RET
                                ;
```

```
;Sine waveform bits

SINE2:DB 080H,083H,086H,089H,08DH,090H,093H,096H,099H,09CH,09FH,0A2H
      DB  A5H,0A8H,0ABH,0AEH,0B1H,0B4H,0B7H,0BAH,0BDH,0BFH,0C2H,0C5H
      DB C7H,0CAH,0CDH,0CFH,0D1H,0D4H,0D6H,0D9H,0DBH,0DDH,0DFH,0E1H
      DB 0E3H,0E5H,0E7H,0E9H,0EBH,0ECH,0EEH,0F0H,0F1H,0F3H,0F4H,0F5H
      DB 0F6H,0F8H,0F9H,0FAH,0FBH,0FCH,0FCH,0FDH,0FEH,0FEH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FEH,0FEH
      DB 0FDH,0FDH,0FCH,0FBH,0FAH,0F9H,0F8H,0F7H,0F6H,0F5H,0F3H,0F2H
      DB 0F0H,0EEH,0EDH,0ECH,0EAH,0E8H,0E6H,0E4H,0E2H,0E0H,0DEH,0DCH
      DB DAH,0D7H,0D5H,0D3H,0D0H,0CEH,0CBH,0C9H,0C6H,0C3H,0C1H,0BEH
      DB 0BBH,0B8H,0B5H,0B3H,0B0H,0ADH,0AAH,0A7H,0A4H,0A1H,09EH,09BH
      DB 098H,094H,091H,08DH,08BH,088H,085H,082H,07EH,07BH,078H,075H
      DB 072H,06FH,06CH,069H,066H,063H,05FH,05CH,059H,056H,053H,050H
      DB 04DH,04BH,048H,045H,042H,03FH,03DH,03AH,037H,035H,032H,030H
      DB 02DH,02BH,029H,026H,024H,022H,020H,01EH,01CH,01AH,018H,016H
      DB 014H,013H,011H,010H,00EH,00DH,00BH,00AH,009H,008H,007H,006H
      DB 005H,004H,003H,003H,002H,002H,001H,001H,000H,000H,000H,000H
      DB 000H,000H,000H,001H,001H,001H,002H,002H,003H,004H,004H,005H
      DB 006H,007H,008H,009H,00BH,00CH,00DH,00FH,010H,011H,014H,015H
      DB 017H,019H,01BH,01DH,01EH,021H,023H,025H,028H,02AH,02CH,02FH
      DB 031H,033H,036H,039H,03BH,03EH,041H,043H,046H,049H,04CH,04FH
      DB 051H,055H,058H,05BH,05EH,061H,064H,067H,06AH,06EH,070H,073H
      DB 077H,07AH,07DH,080H
    ;
;square waveform bits

SQR2 :DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
      DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
```

```
    DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H
    ;
    ;Triangular waveform bits

TRIA2:DB 07FH,081H,083H,085H,087H,089H,08BH,08DH,08FH,091H,093H,095H
    DB 097H,099H,09BH,09DH,09FH,0A1H,0A3H,0A5H,0A7H,0A9H,0ABH,0ADH
    DB 0AFH,0B1H,0B3H,0B5H,0B7H,0B9H,0BBH,0BDH,0BFH,0C1H,0C3H,0C5H
    DB C7H,0C9H,0CBH,0CDH,0CFH,0D1H,0D3H,0D5H,0D7H,0D9H,0DBH,0DDH
    DB 0DFH,0E1H,0E3H,0E5H,0E7H,0E9H,0EBH,0EDH,0EFH,0F1H,0F3H,0F5H
    DB 0F7H,0F9H,0FBH,0FDH,0FFH,0FDH,0FBH,0F9H,0F7H,0F5H,0F3H,0F1H
    DB 0EFH,0EDH,0EBH,0E9H,0E7H,0E5H,0E3H,0E1H,0DFH,0DDH,0DBH,0D9H
    DB 0D7H,0D5H,0D3H,0D1H,0CFH,0CDH,0CBH,0C9H,0C7H,0C5H,0C3H,0C1H
    DB BFH,0BDH,0BBH,0B9H,0B7H,0B5H,0B3H,0B1H,0AFH,0ADH,0ABH,0A9H
    DB 0A7H,0A5H,0A3H,0A1H,09FH,09DH,09BH,099H,097H,095H,093H,091H
    DB 08FH,08DH,08BH,089H,087H,085H,083H,081H,07FH,07DH,07BH,079H
    DB 077H,075H,073H,071H,06FH,06DH,06BH,069H,067H,065H,063H,061H
    DB 05FH,05DH,05BH,059H,057H,055H,053H,051H,04FH,04DH,04BH,049H
    DB 047H,045H,043H,041H,03FH,03DH,03BH,039H,037H,035H,033H,031H
    DB 02FH,02DH,02BH,029H,027H,025H,023H,021H,01FH,01DH,01BH,019H
    DB 017H,015H,013H,011H,00FH,00DH,00BH,009H,007H,005H,003H,001H
    DB 001H,003H,005H,007H,009H,00BH,00DH,00FH,011H,013H,015H,017H
    DB 019H,01BH,01DH,01FH,021H,023H,025H,027H,029H,02BH,02DH,02FH
    DB 031H,033H,035H,037H,039H,03BH,03DH,03FH,041H,043H,045H,047H
    DB 049H,04BH,04DH,04FH,051H,053H,055H,057H,059H,05BH,05DH,05FH
    DB 061H,063H,065H,067H,069H,06BH,06DH,06FH,071H,073H,075H,077H
    DB 079H,079H,07BH,07DH,07FH
    ;
    ;Ramp waveform bits

RAMP2:DB 07FH,080H,080H,081H,081H,082H,082H,083H,083H,084H,084H,085H
    DB 085H,086H,086H,087H,087H,088H,088H,089H,089H,08AH,08AH,08BH
    DB 08BH,08CH,08CH,08DH,08DH,08EH,08EH,08FH,08FH,090H,090H,091H
    DB 091H,092H,092H,093H,093H,094H,094H,095H,095H,096H,096H,097H
    DB 097H,098H,098H,099H,099H,09AH,09AH,09BH,09BH,09CH,09CH,09DH
    DB 09DH,09EH,09EH,09FH,09FH,0A0H,0A0H,0A1H,0A1H,0A2H,0A2H,0A3H
    DB 0A3H,0A4H,0A4H,0A5H,0A5H,0A6H,0A6H,0A7H,0A7H,0A8H,0A8H,0A9H
DB A9H,0AAH,0AAH,0ABH,0ABH,0ACH,0ACH,0ADH,0ADH,0AEH,0AEH,0AFH
    DB 0AFH,0B0H,0B0H,0B1H,0B1H,0B2H,0B2H,0B3H,0B3H,0B4H,0B4H,0B5H
    DB 0B5H,0B6H,0B6H,0B7H,0B7H,0B8H,0B8H,0B9H,0B9H,0BAH,0BAH,0BBH
    DB BBH,0BCH,0BCH,0BDH,0BDH,0BEH,0BEH,0BFH,0BFH,0C0H,0C0H,0C1H
```

```
    DB 0C1H,0C2H,0C2H,0C3H,0C3H,0C4H,0C4H,0C5H,0C5H,0C6H,0C6H,0C7H
    DB C7H,0C8H,0C8H,0C9H,0C9H,0CAH,0CAH,0CBH,0CBH,0CCH,0CCH,0CDH
    DB0CDH,0CEH,0CEH,0CFH,0CFH,0D0H,0D0H,0D1H,0D1H,0D2H,0D2H,0D3H
    DB 0D3H,0D4H,0D4H,0D5H,0D5H,0D6H,0D6H,0D7H,0D7H,0D8H,0D8H,0D9H
DB D9H,0DAH,0DAH,0DBH,0DBH,0DCH,0DCH,0DDH,0DDH,0DEH,0DEH,0DFH
    DB 0DFH,0E0H,0E0H,0E1H,0E1H,0E2H,0E2H,0E3H,0E3H,0E4H,0E4H,0E5H
    DB 0E5H,0E6H,0E6H,0E7H,0E7H,0E8H,0E8H,0E9H,0E9H,0EAH,0EAH,0EBH
    DB 0EBH,0ECH,0ECH,0EDH,0EDH,0EEH,0EEH,0EFH,0EFH,0F0H,0F0H,0F1H
    DB 0F1H,0F2H,0F2H,0F3H,0F3H,0F4H,0F4H,0F5H,0F5H,0F6H,0F6H,0F7H
    DB 0F7H,0F8H,0F8H,0F9H,0F9H,0FAH,0FAH,0FBH,0FBH,0FCH,0FCH,0FDH
    DB 0FDH,0FEH,0FEH,07FH
    ;
;Staircase waveform bits

STA2: DB 08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH
    DB 08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH
    DB 08FH,08FH,08FH,08FH,08FH,08FH,08FH,08FH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
    DB CFH,0CFH,0CFH,0CFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
    DB DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
    DB DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
    DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
    DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
    DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0FFH,0FFH,0FFH,0FFH
    DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
    DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
    DB 0FFH,0FFH,0FFH,0FFH
    ;
    ;Random waveform bits
RAND2:DB 07FH,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H,000H
    DB 000H,000H,000H,000H,000H,000H,000H,000H,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH,09FH
    DB 09FH,09FH,09FH,09FH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
```

```
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH
    DB AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0AFH,0BFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
    DB 0CFH,0CFH,0FFH,0FFH
06AB
CODE    ENDS

END
```

symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
CODE	06AB	PARA	NONE
SSEG	0000	PARA	NONE

Symbols:

N a m e	Type	Value	Attr
DELAY1	L NEAR	00C5	CODE
DELAY10	L NEAR	010C	CODE
DELAY13	L NEAR	0114	CODE
DELAY15	L NEAR	011C	CODE
DELAY17	L NEAR	0124	CODE
DELAY2	L NEAR	00EC	CODE
DELAY4	L NEAR	00F4	CODE
DELAY6	L NEAR	00FC	CODE
DELAY8	L NEAR	0104	CODE

```
FREQ1 . . . . . .  L NEAR    00E9  CODE
FREQ2 . . . . . .  L NEAR    00F1  CODE
FREQ3 . . . . . .  L NEAR    00F9  CODE
FREQ4 . . . . . .  L NEAR    0101  CODE
FREQ5 . . . . . .  L NEAR    0109  CODE
FREQ6 . . . . . .  L NEAR    0111  CODE
FREQ7 . . . . . .  L NEAR    0119  CODE
FREQ8 . . . . . .  L NEAR    0121  CODE

NEXT . . . . . .  L NEAR    003C  CODE
NEXT1 . . . . . .  L NEAR    0054  CODE
NEXT2 . . . . . .  L NEAR    006C  CODE
NEXT3 . . . . . .  L NEAR    0084  CODE
NEXT4 . . . . . .  L NEAR    009C  CODE
NEXT5 . . . . . .  L NEAR    00B5  CODE

POLL . . . . . .  L NEAR    0017  CODE
PPIA . . . . . .  NUMBER    0000
PPIB . . . . . .  NUMBER    0001
PPIC . . . . . .  NUMBER    0002
PPIC_C . . . . . .  NUMBER    0003

RAMP1 . . . . . .  L NEAR    007B  CODE
RAMP2 . . . . . .  L NEAR    03AB  CODE
RAND2 . . . . . .  L NEAR    05AB  CODE
RANDO1 . . . . . .  L NEAR    00AC  CODE

SINE1 . . . . . .  L NEAR    0033  CODE
SINE2 . . . . . .  L NEAR    012B  CODE
SQR1 . . . . . .  L NEAR    004B  CODE
SQR2 . . . . . .  L NEAR    022B  CODE
STA2 . . . . . .  L NEAR    04AB  CODE
STACK . . . . . .  NUMBER    0100

STAIR1 . . . . . .  L NEAR    0093  CODE

TRIA1 . . . . . .  L NEAR    0063  CODE
TRIA2 . . . . . .  L NEAR    032B  CODE

@CPU . . . . . .  TEXT  0101h
@FILENAME . . . . . .  TEXT  wavef3
@VERSION . . . . . .  TEXT  510
```

319 Source Lines
319 Total Lines
48 Symbols

47696 + 364363 Bytes symbol space free

0 Warning Errors
0 Severe Errors

APPENDIX B

8086/8088 INSTRUCTION SET

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT / OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
SHIFTS	
SHL /SAL	Shift logical / arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word

ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert word to double word

STRING OPERATIONS	
MOVS	Move byte or word string
MOVSB/MOVSW	Move byte or word string
CMPS	Compare byte or word string
SCAS	Scan byte word string
LODS	Load byte or word string

STOS	Store byte or word string
REPE/REPZ	Repeat while equal/zero
FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag

EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for TEST pin active
ESC	Escape to external processor
LOCK	Lock bus during next instruction
NO OPERATION	
NOP	No operation

ITERATION CONTROLS	
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/ zero
LOOPNE/ LOOPNZ	Loop if not equal / not zero
JCXZ	Jump if register CX=0
INTERRUPT	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Interrupt return

UNCONDITIONAL TRANSFERS	
CALL	Call procedure
RET	Return from procedure
JMP	Jump

CONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal /zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity /parity even
JS	Jump if sign

APPENDIX C

Carrying Out Measurement from MBM signal generator With The PM 3384 Autoranging Combiscope

1. Make sure that you are in digital mode by pressing Analogue Key (Yellow Key).
2. Press the **AC DC/GND** key once for DC input coupling; the bottom text line now displays "=".
3. Use **TIME/DIV** keys <> to change the display until you have the wanted number of periods of oscillations on the screen.
4. Use the **POS** control of the channels to center the waveform.
5. Press **AMPL** up or down to have a good resolution for the signal waveform.

Reading of Periods or Frequencies:

1. Press the **CURSOR** key to activate the cursor menu. The menu will appear on the right side of the display. The blue soft-keys right to it controls it. There are two types of cursors. The symbol|| stands for the vertical cursors to measure time differences (frequency), whereas the symbol = is for the horizontal cursors and is used to measure amplitudes.
2. To read periods, make sure that the vertical cursor is on (selected). Turn the **TRACK** control to move the left cursor to one extreme of the waveform.
4. Turn the Δ knob to bring the right cursor to the next extreme of the same waveform.
5. To read frequencies press the softkey 6 next to **READOUT** and the softkey 1 until 1/ΔT is highlighted, then press softkey 6 (**RETURN**). The difference between the cursors will now be displayed in Hz.

Reading Amplitudes:

1. To read amplitudes, use softkey 2 to highlight " ", the horizontal cursor is now on.

2. Turn the Δ key until both lines of the cursor coincides (ΔV =0 is shown) and then use **TRACK** to bring the cursor line to the baseline of the displayed waveform.

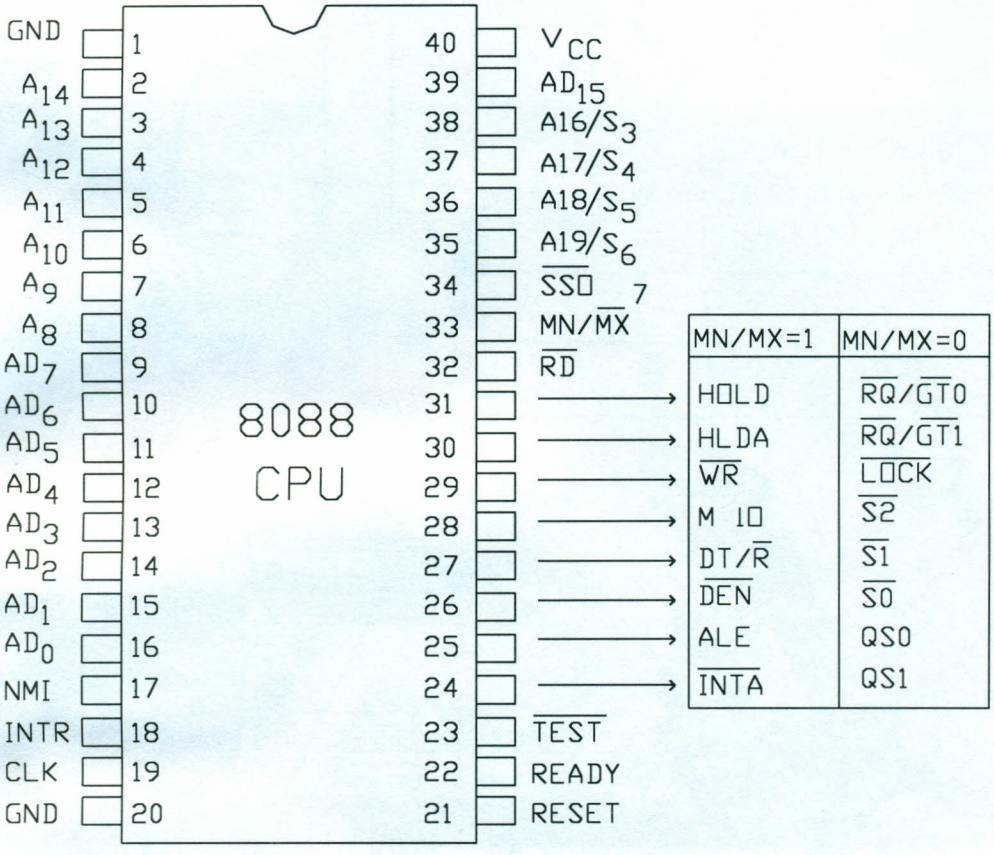
3. Turn now the Δ key until your cursor line is at an extreme. The voltage difference between the cursors is the amplitude (A) can then be read.

4. After completing the measurement use soft-key 1 to turn off the cursor menu.

The PM 3384 Autoranging combiscope is then interfaced with computer via RS-232 serial port so that desired waveforms can be magnified for proper Viewing and printing.

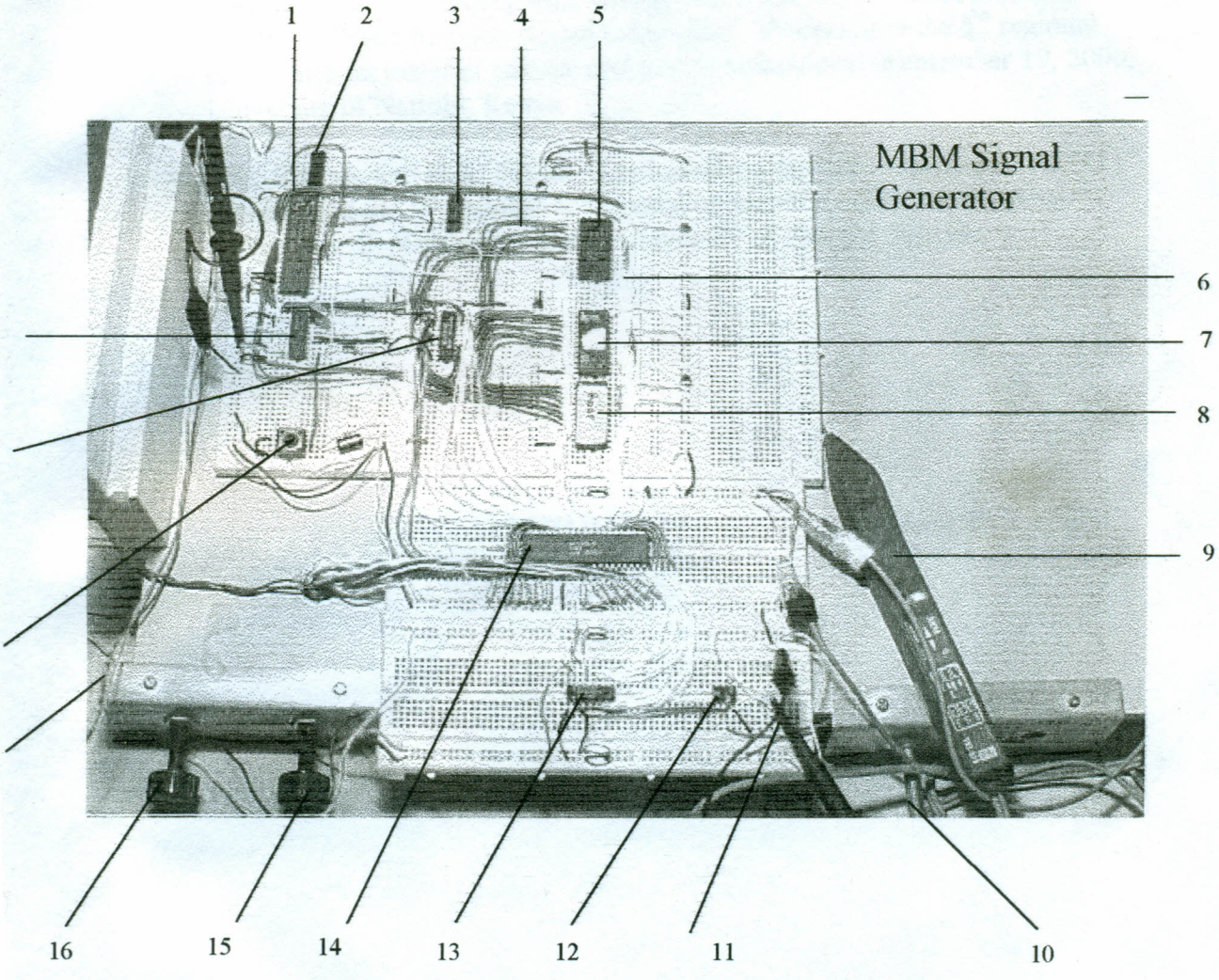
APPENDIX D

8088 MICROPROCESSOR PIN CONFIGURATION



APPENDIX E

PHOTOGRAPH OF THE MBM SIGNAL GENERATOR



1. 8088 microprocessor
2. NOT gate
3. 74LS138 decoder
4. Address bus
5. 6116 RAM
6. Data bus
7. EPROM 1
8. EPROM 2
9. Logic probe
10. Amplitude selector
11. Output to CRO
12. Current to voltage converter (741 OPAMP)
13. DAC 0800
14. 8255A PPI
15. Frequency selector
16. Waveform selector
17. Power supply
18. Reset switch
19. 74LS373 latch
20. 8284A clock generator

APPENDIX F

LIST OF PUBLICATIONS

1. P.M. Karimi, G.A. Ibitola, R.L. Stangl and S. Namuye: "Microprocessor Controlled Multifunction Signal Generator". Presented at the 5th regional conference on material science and device technology. September 19, 2000, University of Nairobi, Kenya.

2. P.M. Karimi, and G.A. Ibitola: "High electron mobility transistors (HMTS)" Presented at the 5th regional conference on material science and device technology. September 19, 2000, University of Nairobi, Kenya.