

•

COMPUTER ASSISTED LEARNING OF BASIC WAVE CONCEPTS

BY

SALAMI ADEKUNLE LUQMAN

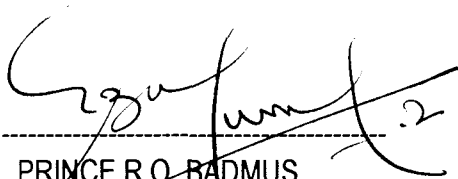
PGD / MCS / 244 / 96

**A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHS/COMPUTER ,
FEDERAL UNIVERSITY OF TECHNOLOGY , MINNA,
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF
POST-GRADUATE DIPLOMA IN COMPUTER SCIENCE.**

MARCH , 1998

APPROVAL PAGE / CERTIFICATION

This is to certify that having read through this research work carried out by SALAMI ADEKUNLE LUQMAN, it, in our opinion , conforms to the acceptable standard as a project for the award of post-graduate diploma in computer.



PRINCE R.O. BADMUS
PROJECT SUPERVISOR

Date

DR. K.R. ADEBOYE
Head of Department

Date

EXTERNAL EXAMINER

Date

ACKNOWLEDGEMENT

Glory be to God for giving me the idea of reading this course through a suggestion by a friend, Dr. Olowofela Niyi. I now fully understand what it is to "seek the kingdom of God first" so that other things can follow. The completion of this course is a miracle, considering the problems I encountered especially in completing the project. Thank you Lord.

Apart from supervising this project and numerous others, Alhaji Prince R.O. Badmus coordinates the P.G.D. Computer program. I am immensely grateful to him, for always sparing time, out his usually tight schedule, to patiently attend to me, any time called upon, and for making appropriate corrections to ensure the quality of this project. May Allah grant you more wisdom in the management of human affairs.

I wish to thank the Head of Mathematics / Computer science, Federal University of Technology, Minna, for providing the forum to acquire knowledge in computer at this level. I have learnt a lot from his acute sense of humour (while driving home some "bitter truths"), and his vision of a better nation.

My appreciation would not be complete without thanking the entire staff of Computer Science especially my lecturers. May God reward your efforts in building a literate nation. Also I am grateful to the entire staff of Computer centre, Federal College of Education, Kontagora, especially the Director of the centre, Mr John Haruna, for his kind support and understanding. Mr Bimpe Bamigbade has been a good friend offering professional advice from time to time from his former experiences. Not left out are the people with the magic fingers for their friendly "worries", and "troublesome" interludes which invariably removed part of the loneliness and, "wear and tear", of the "long" hours of writing and debugging the program.

The economic concept of choice was inevitable for me, in order to run the PGD program. It invariably demanded time and resources, which could have been given to my "fiancee", Funke Alaba Ajewole, who has been a pillar of support since I knew her. I don't know how to thank you nor repay you for all that you have done for me. Thank you so much for enduring and looking the other way while the computer became your "husband's" better half. I pray that God will adequately reward you as HE deems fit.

Daddy assisted morally and financially. Your prayer that your son will become "somebody" in life will surely be granted as God deems fit.

DEDICATION

To

Dr. Olowofela Niyi

Department of Physics,
University of Ibadan.

ABSTRACT

The versatility of a computer is exploited in what is generally referred to as Computer Assisted Learning (CAL). This work is one of such CAL programs intended to alleviate the problem of lack of adequate instructional materials to teach the concept of waves.

The computer is made to simulate the prevailing conditions in a medium, as a wave is propagated through it. It also serves as a wave-generator, and automatically displays the graph on its monitor.

The wave concepts are introduced with appropriate texts punctuated with interactive questions, and terminated by simple tests.

Provision is made for the more astute learner (or user) to investigate superposition of waves and Fourier series analysis of complex waveforms.

CONTENTS

Approval page	ii
Acknowledgement	iii
Dedication.....	v
Abstract	vi
Contents.....	vii

Chapter one

INTRODUCTION.....	1
1.1 Communication.....	1
1.2 Wave study.....	1
1.3 Instructional materials.....	1
1.3.1 Ripple tank.....	2
1.3.2 Analysis.....	2
1.3.3 Graph.....	2
1.3.4 Oscilloscope and Wavegenerator.....	2
1.3.5 Computer as an instructional material.....	3
1.3.6 Computer Aided Learning (CAL).....	3
1.4 Scope of Research.....	3
1.5 Methodology.....	4
1.5.1 Enhancements.....	4
1.5.2 Interpretation.....	4
1.5.3 Compilation.....	4
1.5.4 Editor and on-line help.....	4

1.5.5	Modular Programming.....	4
1.5.6	Line numbering.....	5
1.5.7	Graphics.....	5
1.5.8	Portability.....	5

Chapter Two

	WAVES.....	6
2.0.1	Transverse Waves.....	7
2.0.2	Longitudinal Waves.....	7
2.0.3	Progressive Waves.....	7
2.1	The Graph.....	7
2.2	Frequency.....	8
2.3	Wavelength.....	8
2.4	Amplitude.....	8
2.5	Period.....	8
2.6	Damped Oscillations.....	9
2.7	Clipping.....	9
2.8	Superposition.....	9
2.9	Modulation.....	10
2.10	Decomposition of Waves.....	11
2.11	Fourier Analysis.....	11

Chapter Three

	SYSTEM DESIGN AND STRUCTURE.....	13
3.1	Sections.....	13

3.1.1	Main module.....	13
3.1.2	Tutorial.....	13
3.1.3	Wave Draw.....	14
3.1.4	Fourier Wave Draw.....	16
3.2	Accessories.....	17

Chapter Four

	PROGRAM DEVELOPMENT AND SOFTWARE IMPLEMENTATION.....	18
4.0	General.....	18
4.0.1	Graphics mode.....	18
4.1	Program name.....	18
4.2	Control module.....	20
4.3	The Tutorial program.....	20
4.4	Wave draw.....	23
4.4.1	Menu display.....	23
4.4.2	Drawing of graphs.....	26
4.4.3	Speed and resolution.....	26
4.4.4	Memory management.....	28
4.4.5	Amplitude.....	28
4.4.6	Vertical offset.....	28
4.4.7	Horizontal offset.....	29
4.4.8	Drawing area.....	29
4.4.9	Frequency.....	30
4.4.10	Phase.....	30

4.4.11 Decay.....	30
4.4.12 Clipping.....	30
4.4.13 The "DRO" subprograms.....	31
4.4.14 Erasure.....	31
4.4.15 Loading.....	31
4.5 Fourier -Draw.....	32

Chapter Five

SUMMARY COMCLUSIONS AND RECOMMENDATIONS.....33

5.0.1 Executable file.....	33
5.0.2 User-friendliness.....	34
5.0.3 Speed.....	34
5.1 Evaluation of objectives.....	34
5.1.1 Tutorial.....	34
5.1.2 Wave-draw.....	34
5.1.3 Fourier-draw.....	35
5.2 Limitations.....	35
5.3 Recommendations.....	35

REFERENCES.....37

APPENDIX A

Samples from program while running.....	38
---	----

APPENDIX B

Control module program list.....	46
----------------------------------	----

APPENDIX C

Program List for Tutorial section.....50

APPENDIX D

Program List for Wave-draw section.....64

APPENDIX E

Ascii text file for Tutorial section.....79

APPENDIX F

Program List for Fourier-draw section.....85

APPENDIX G

Program List for generating the data tables.....99

CHAPTER ONE

INTRODUCTION

Examples of waves abound in nature. The Sun produces light which enables us to see during the day, warm the surface of the earth, as well as producing invisible ultraviolet radiation, which is essential for our body to manufacture vitamin D, by our skin. Plants utilize light to to manufacture food, through a process called photosynthesis. Light, ultraviolet and infrared rays are all radiations which travel at a speed of 3×10^8 metres per second to reach us here on earth. They are all propagated by wave motion.

1.1 COMMUNICATION

Sound and consequently speech is also transmitted from one place to another by wave motion. Wireless communication via radio, television, mobile telephones etc., are also propagated by wave motion.

1.2 WAVE STUDY

The study of waves is difficult to understand or comprehend by most students. At best a vague idea about the concept is impressed on their minds. Most students perceive a wave as a "snake-like" curve, whose mathematical representation and analysis they have to understand. The real nature of waves is lost to them.

One of the primary reasons for this lies in the nature of waves itself. It exists only when there is motion. Therefore any passive representation, cannot effectively convey the true nature of it.

Secondly wave motion involves microscopic particles. A wave is transmitted through a medium by transient vibrations of the molecules of the medium, as a result of impulse, usually received from an external source. Since these molecules are too small to be seen, an observer merely notices the effect of the wave and not the actual mechanism of propagation. Even if it were possible to see the molecules, usually the motion of the molecules take place at such high rates that it cannot be followed by the eye.

1.3 INSTRUCTIONAL MATERIALS

In order to facilitate better understanding of the wave concepts, several instructional materials are used. One of the most popular, is the slinky spring. It is a coil of wire which is about 1.5 metres long.

Usually, the spring constant is low so that it can be easily compressed (or rarefied). A tug at one end of the spring is transmitted to the other end by successive compression and rarefaction of the spring.

1.3.1 Ripple tank

Another popular material is the ripple tank. It consists of a metal bob attached to an electrical vibrator

touching the surface of water in a shallow tank. Ripples spread out from the point of contact each time the bob is incident on the surface. Another version of the ripple tank has two vibrators, so that two waves are produced at the same time. Mostly the vibrations are at the same frequency. Hence the effect of the two waves interacting with one another (superposition), can be observed.

1.3.2 Analysis

Despite these instructional materials, the fact remains that it is only the effect that is noticed and not the actual mechanism of propagation of the wave that is observed.

For more qualitative analysis, the nature of waves are mathematically resolved to equations from which the behaviour of a wave may be analysed, interpreted, (or predicted). By skillful manipulation of equations, more complex wave equations are derived.

1.3.3 Graph

A complement to this mathematical analysis is the a displacement-time graph (or displacement-distance) which can be obtained by plotting the displacement of a molecule in the medium against time.

(All molecules in the medium will experience the same force). These graphs readily appeal to visual appreciation.

1.3.4 Oscilloscope and wavegenerator

For active graph plot, the pair of an oscilloscope and wavegenerator are indispensable. A wavegenerator is an electronic appliance that produces alternating current (a.c.) internally, usually with variable frequency and voltage. When the a.c. from the generator is applied across the input terminals of the oscilloscope, a graph of voltage against time is displayed on the screen. If instead of a wave generator, a voice is recorded with a microphone, converted to electrical signals a.c. and then connected to the oscilloscope, the typical wave pattern of speech, which is usually complex, can be observed. This is one of the best merits of the oscilloscope.

Some basic wave concepts like amplitude, frequency, wavelength, period etc., can be demonstrated. Double beam oscilloscopes, which can display two graphs simultaneously, are also available for comparative analysis. Superposition of waves leading to different effects like modulation, beats, constructive and destructive interference, can be comprehended and understood.

1.3.5 Computer as an instructional material

One of the limitations of the wavegenerator-oscilloscope combination, is the cost of the devices. Moreover their uses are limited to the laboratory and again limited to few applications. They are only useful to

wave analysis, though, an oscilloscope can be used to analyse electronic circuits.

In addition to the foregoing, they still cannot demonstrate the actual mechanism of propagation of the waves. With the advent of computers, virtually every establishment has a computer or aspire to have one. Computers were invented to speed up computational processes. Little wonder then that today computers are indispensable to science and technology. They are now equally applicable to other facets of life. Historians find it invaluable in storing and retrieving information at fractions of time it would have taken some years back. Even churches are becoming computerised nowadays.

1.3.6 Computer Aided Learning (CAL)

One of the areas of application of computers is called Computer Assisted Learning (CAL). A set of instructions (program) is fed into a computer in such a way that while carrying out the instructions the computer becomes the teacher, and the user the student. One of the advantages of CAL is that, a user can learn at his or her own pace, as opposed to the classroom environment, whereby many students, of varying intelligence, are taught simultaneously. The student is able to digest an aspect of the lesson fully, before proceeding to the next one. Former lessons can even be reviewed.

As an instructional material the computer can be made to simulate prevalent conditions as a wave is propagated through a medium. Molecular vibrations can be observed at lower speed.

With appropriate graphic instructions, diagrams can be precisely drawn to illustrate corresponding text. Limited interaction is also possible. Just as a teacher poses questions to his students the program is written, so that the computer poses questions at appropriate intervals to test the knowledge of the student, as well as sustain his or her interest.

1.4 **SCOPE OF RESEARCH**

This research is aimed at writing a computer program to teach any layman, the basic wave concepts, as well as portraying the concepts in a new light to those who are already familiar with them. It is intended to include a WAVE DRAW section to serve as an environment in which the student or user can generate and display waves in order to excite their interest as well as consolidate knowledge gained from the tutorial section. The slightly advanced concept of Fourier analysis of waves will also be included to serve as a research instrument to mathematicians and physicists as well as serve as a demonstration of the abstract concept.

1.5 **METHODOLOGY**

QBASIC, Microsoft's (a United States software company) version of the BASIC programming lan-

guage will be used in writing the program for the following reasons:-

1.5.1 Enhancements

It has sufficient enhancements, making it comparable to 'higher' languages like PASCAL, C, FORTRAN, etc., in terms of structured programming. There are sufficient commands to achieve superb program iteration. These include FOR-TO-NEXT, DO-WHILE-LOOP, DO-LOOP, and WHILE-WEND constructs. Conditional branching is also possible using IF-THEN-ELSEIF-ELSE-END IF, and SELECT CASE-CASE-CASE ELSE-END SELECT, constructs.

1.5.2 Interpretation

It has an advantage of being easier to debug, since it interpretes programs during execution.

1.5.3 Compilation

If VISUAL BASIC, another Microsoft product can, be obtained, it can be compiled to a stand-alone executable program.

1.5.4 Editor and on-line help

It has a versatile editor and HELP section that makes it a very useful research instrument. Traditional commands like LOAD, LIST, SAVE, etc., have been replaced by user-friendly text editor, featuring pull-down menu, cutting, copying and pasting. In addition, it points out syntax errors once a line is written (before running the program). It also indicates correct syntax by changing all the commands on a line originally in lower case to upper case while leaving the variables in lower case. This promptly assures the programmer that the line is error free.

1.5.5 Modular programming

QBASIC supports the development of programs in modules, to aid the understanding of the entire program logic as a whole. Since each module is generally designed to perform a particular task, editing of a program is made easier, by going directly to that part of a program (module) that is responsible for an error or malfunction.

1.5.6 Line numbering

The traditional numbering of each line of the standard BASIC is not mandatory. In fact, an entire program may not contain a single line number. For referencing subroutines, a line can be assigned a label which is usually a meaningful phrase followed by a colon (e.g. merge_diagrams:).

1.5.7 Graphics

QBASIC offers a very versatile graphics capability. There are library functions for drawing circles,

lines, eclipses, arc of circles, rectangles, etc.

Various screen modes support different image resolution as desired. Each screen mode offers a range of colors that can be printed in foreground, background and border. At the peak of this are high resolution screen modes that offers up to 253 color combinations.

Simulation and animation are also possible. "Photographing" a graphics image and PUTting it at any other part of the screen, with the options of retaining the background, erasing the background, or printing in reverse video. Hence with appropriate logic, the image, can be made to move on the screen.

1.5.8 Portability

Most microcomputers (PCs) have Microsoft's DOS operating system, which provides QBASIC as a standard feature. Hence any program written in QBASIC will run on most PCs.

CHAPTER TWO

WAVES

A wave allows energy to be transferred from one point to another some distance away without the particles of the medium travelling between the two points. For example if a small weight is suspended by a string, energy to move the weight may be obtained by repeatedly shaking the other end of the string up and down, through a small distance. Waves which carry energy then travel along the string from the top to the bottom. Likewise, water waves may be spread along the surface from one point A to another point B, where an object floating on the water will be disturbed by the wave. No particles of water at A, actually travel to B in the process. The energy in the electromagnetic spectrum comprising X-rays, light waves and ultraviolet rays for example may be considered to be carried by electromagnetic waves from the radiating body to the absorber. Again sound waves carry energy from the source to the ear by the disturbance of the air.

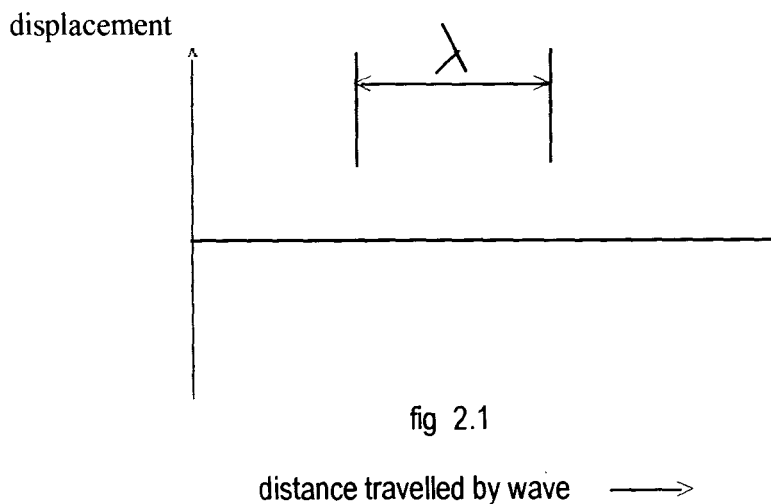


fig 2.1

If the source of origin of the wave oscillates with frequency f , then each point in the medium concerned oscillates with the same frequency. A snapshot of the wave profile or waveform may appear, at an instant, as in fig. 2.1 above. The source repeats its motion f times per second, so a repeating waveform is observed spreading from it. The distance between corresponding points in the successive waveforms, such as two successive crests and two successive troughs, is called the wavelength λ . Each time the source vibrates once, the wave moves forward a distance of λ . Hence the velocity c of the waves which is the distance the profile moves in one second is given by

$$c = f\lambda$$

The equation is true for all the wave motion, whatever its origin, that is it applies to sound waves,

electromagnetic waves and mechanical waves.

2.0 TYPES OF WAVES

2.0.1 Transverse

A wave which is propagated with the direction of vibrations of the molecules of the medium perpendicular to the direction of travel of the wave is called a transverse wave. Examples include waves on plucked strings, and on water. Electromagnetic waves which include light waves are also transverse in nature.

2.0.2 Longitudinal

In contrast to a transverse wave, a longitudinal wave is one in which the vibrations occur in the same direction of travel of the wave e.g. sound waves.

2.0.3 Progressive

Both transverse and longitudinal waves are progressive. This means that the wave profile moves along with the speed of the wave. If a snapshot is taken of a progressive wave, (fig 1 above), it repeats at equal distances. The repeat distance is the wavelength. If one point is taken and the profile is observed as it passes this point, then the profile is seen to repeat itself at equal intervals of time called the period T .

2.1 THE GRAPH

Two kinds of graph may be drawn. A displacement-distance graph for a transverse mechanical wave. Figure 2 shows the displacements y of the vibrating particles of a medium, at different distances x from the source, at a certain instant.

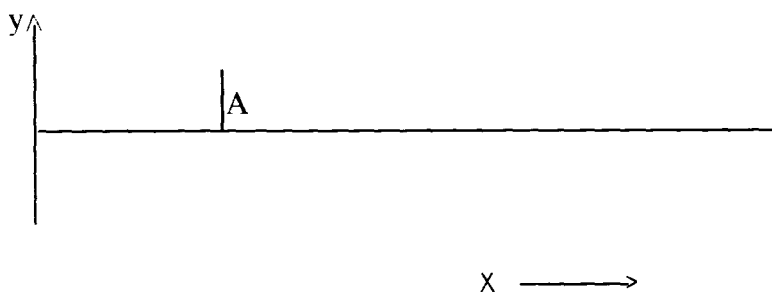


fig. 2.2

A longitudinal wave can also be represented by a transverse displacement-distance graph. The displacements in this case however being those of the vibrating particles in the line of travel of the wave. Thus y and x are in the same direction in the wave but at right angles to each other in the graph, i.e. the graph represents longitudinal displacements as if it were transverse.

2.2 FREQUENCY

If the source of a wave makes f vibrations per second, so too will the particles of the transmitting medium, in the case of a mechanical wave and as will the electric field E and magnetic field B , in an electromagnetic wave. That is, the frequency f of the wave equals the frequency of the source.

2.3 WAVELENGTH

When the source makes one complete vibration, one wave is generated and the disturbance spreads out a distance λ from the source. If the source continues to vibrate with constant frequency f , then f waves will be produced per second and the wave advances a distance $f\lambda$ in one second. If v is the speed, then $v = f\lambda$

This relationship holds for all wave motions.

2.4 AMPLITUDE

The maximum displacement of a wave from the equilibrium position (zero displacement position) is called the amplitude of a wave (A in fig. 2 above). Since the direction of motion of a wave changes every half of a period, the displacement take negative values. Hence the amplitude A is equal to $-A$ in this respect. In most cases, A is equal to absolute $-A$.

2.5 PERIOD

Since the motion of a wave is repeated periodically, the displacement-distance graph can also be plotted as a displacement-time graph. In this respect, the concepts of frequency, and period are easier to determine. The period is the time a wave takes to make one complete oscillation. If

$$v = f \lambda$$

then, distance X / time $t = f\lambda$

Thus, $X = f \lambda t$

The distance moved by a wave in one Period is $x = f \lambda T$

But, $f = 1 / T$

$$X = \lambda T / T = \lambda$$

Thus the distance travelled by a wave can be calculated after a specified length of time.

2.6 DAMPED OSCILLATIONS

A wave is sustained as a result of continuous supply of impulses from the source. However, with time, the energy of the source reduces. Consequently, the amplitude of vibration of the particles of the medium also reduce accordingly, until the finally it becomes zero. The oscillation stops. The oscillation is said to have

decayed to zero. If the decay process is imposed on the vibrating system, then it is called damping, as in motor vehicle shock absorbers. The displacement-time graphs of such system will be like figure 2.3 below.

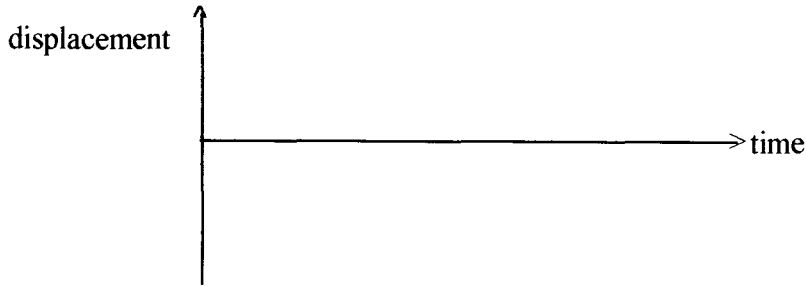


fig 2.3

2.7 CLIPPING

The molecules or atoms of a substance are naturally constrained. There is a limit to which an atom can be displaced from its equilibrium position. In the same vein, macroscopically, a vibrating object may be constrained as a result of its nature, external influences etc., so that there is a limit to the maximum amplitude it can assume, while vibrating.

In a situation where the amplitude of vibration of the source exceeds this limit, then the molecules or atoms stop at this limit, for the period of time that the displacement of the source exceeds this limit, and reduces to the limit. The graph of such a vibrating system is shown in figure 2.4 below. The vibration of the medium is shown in heavy lines superimposed on that of the source.



fig 2.4

2.8 SUPERPOSITION

When two waves travel through a medium, their combined effect at any point can be found by the principle of superposition. This states that the resultant displacement at any point is the sum of the separate displacements due to the two waves. This has the effect of reinforcing the waves at some points and cancellation at others. The resulting effect of this is called an interference pattern. Figure 2.5 below illustrates this phenomenon for two waves of slightly different frequencies.

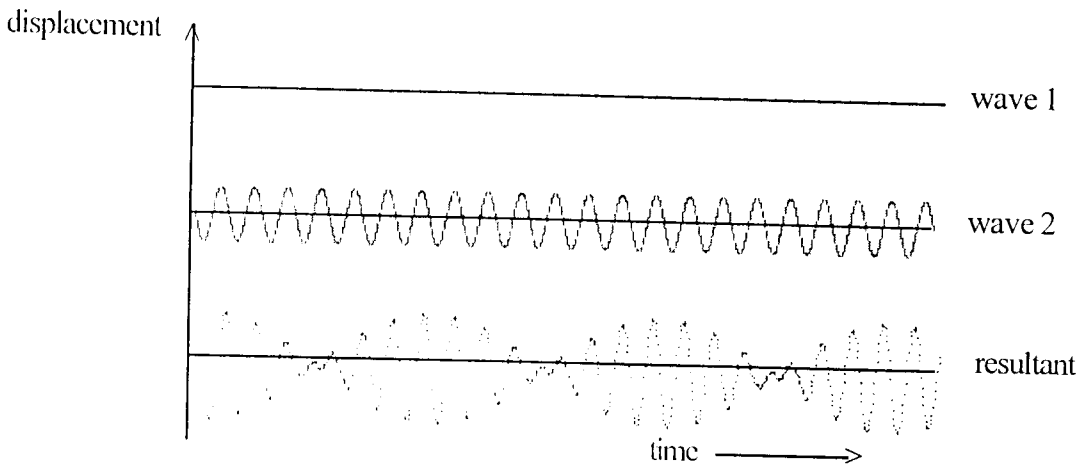


fig. 2.5

The resultant wave pattern is called BEATS, in this case.

2.9 MODULATION

Waves of low frequency are easily attenuated in air. The range of propagation of a wave increases as the frequency increases. In broadcasting, low frequency signals are superposed on high frequency waves (carrier) since the low frequency signals cannot travel by themselves. This process is known as modulation. The resulting wave pattern is shown in figure 2.6 below.

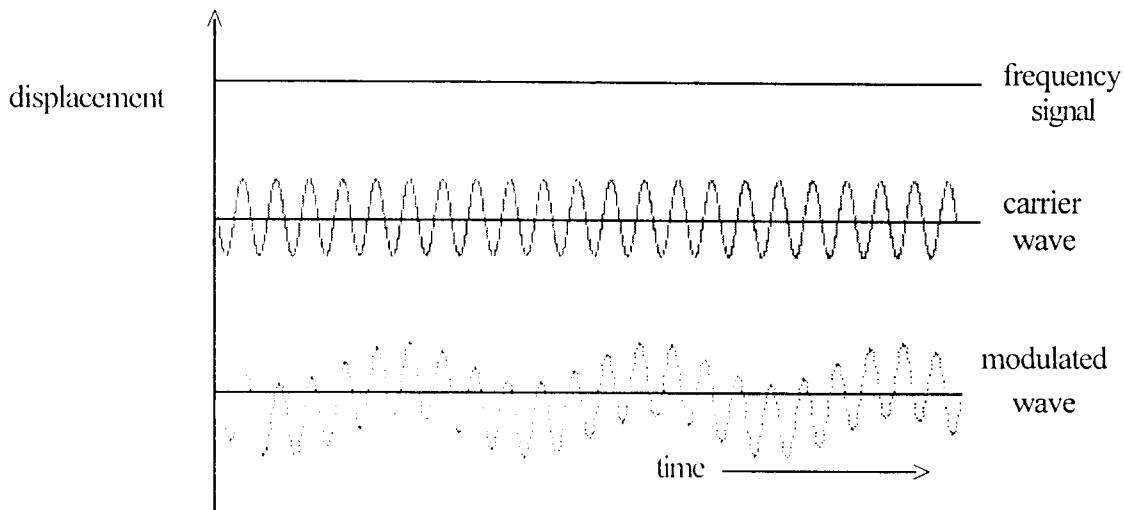


fig 2.6

2.10 DECOMPOSITION OF WAVES

If the same note sounded on the violin, is repeated on the piano, even an untrained listener will be able to tell which instrument is being used without seeing it. The quality (or timber) of the note is different in each case.

The explanation for this is that, no instrument produces a "pure" note, i.e. a note consisting of only a single frequency. In practice the nearest approach to a pure note is that obtained by sounding a tuning fork. When the same note is sounded on other instruments, they sound different because of the presence of other

The Fourier series for :

1. Square wave is

$$f(x) = 4 / \pi (\sin x + 1 / 3 \sin 3x + 1 / 5 \sin 5x + \dots)$$
$$= 4 / \pi \sum_{i=1}^{\infty} \sin (2i-1)x / 2i-1$$

2. Triangular wave is

$$f(x) = (\pi / 2 - 4 / \pi) \sum_{i=1}^{\infty} \cos (2i-1)x / 2i-1$$

3. Sawtooth wave is

$$f(x) = 1 / 2 - 1 / \pi \sum_{i=1}^{\infty} 1 / k \sin kx$$

Note :- The sawtooth wave function is

$$f(x) = x / 2\pi$$

CHAPTER THREE

SYSTEM DESIGN AND STRUCTURE

The program is designed to teach basic wave concepts, with illustrations, simulation, of the graphic representation of waves.

In line with structural programming technique the program is more of a package, divided into unit programs. Each program is designed to take care of a section of the overall program, and is further divided into subprograms, each of which is designed to accomplish a specific task. A subprogram may contain one or more subroutines to perform minor functions within it. Another unit contains the title given to the package as a whole as well as the MENU. Any of the three sections of the package can be invoked from this unit which in essence represent the control module. At the end of a session, within any of the sections, control is returned to this module from where the user may decide to proceed to another section of package or to quit to the operating system

3.1 SECTIONS

In all there are three basic unit, though other units are present that serve more or less as utilities, or containing data needed by one or more of the basic units.

3.1.1 Main Module

As customary of professional" software packages, the program is introduced by displaying "WELCOME TO THE WORLD OF WAVES"on the monitor.

The sequence of instructions displaying the title is terminated by a conditional branch construct, so as to enable the user to select a part of the program, to use.

The options are:-

1. TUTORIAL
2. WAVE DRAW
3. FOURIER DRAW
4. QUIT

3.1.2 Tutorial

This option is presented first since the program is basically meant to teach. It employs other classes of subprograms. Therefore, the user is given the freedom

to practice after going through the tutorial. Due to the technicalities involved, a lot of subprograms are found in this section. The hierarchy of operations is illustrated in figure 3.1 below.

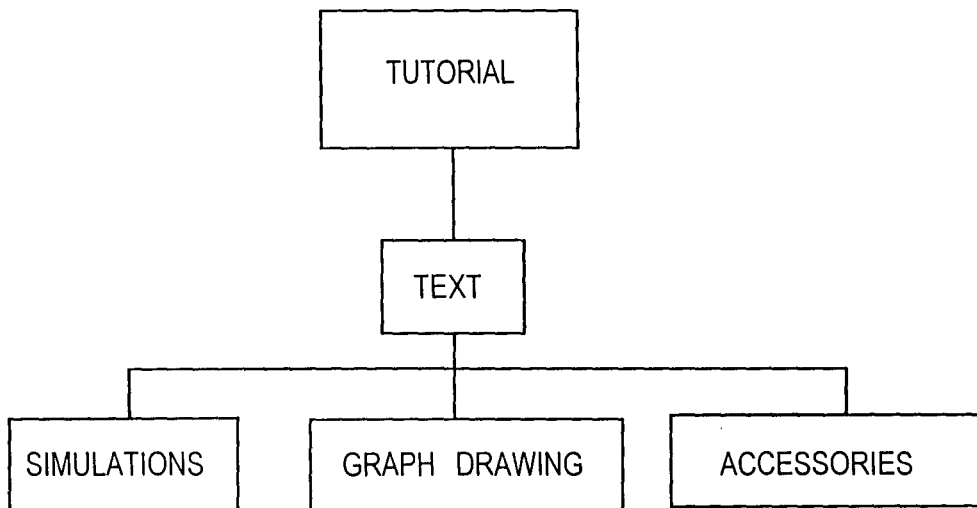


FIG. 3.1

The WAVE TUTOR program utilizes several subprograms and a text file. The text is retrieved bit by bit and mixed with simulations and diagrams as appropriate. There are four types of simulations namely:-

(a) Lmotion

This simulates a rectangular box that moves from one end of the screen to another.

(b) Cmotion

It simulates a satellite moving round a circular orbit for three revolutions.

(c) Pendulum

Generates a a pendulum that is at rest, displaced when a key is pressed, begins to vibrate when another key is pressed and finally comes to rest after ten oscillations (provided the user does not press another key before then).

It can also illustrate damped oscillations.

(d) Vmotion

Initially, this subprogram is made to generate a few layers of "atoms", before one of them is isplated and used to demonstrate vibrational motion.

3.1.3 WAVE DRAW

The drawing of graphs to represent waves is a convenient tool in analysing the nature of waves. Basically there are three graph generators. Two of them produce waves with individual differences in attributes, while the third, adds the displacements of the other two, and produces a resultant wave. This is essential to the investigation of the principle of superposition. The diagram shown below illustrate the relationship between the main program and the subprograms and subroutines.

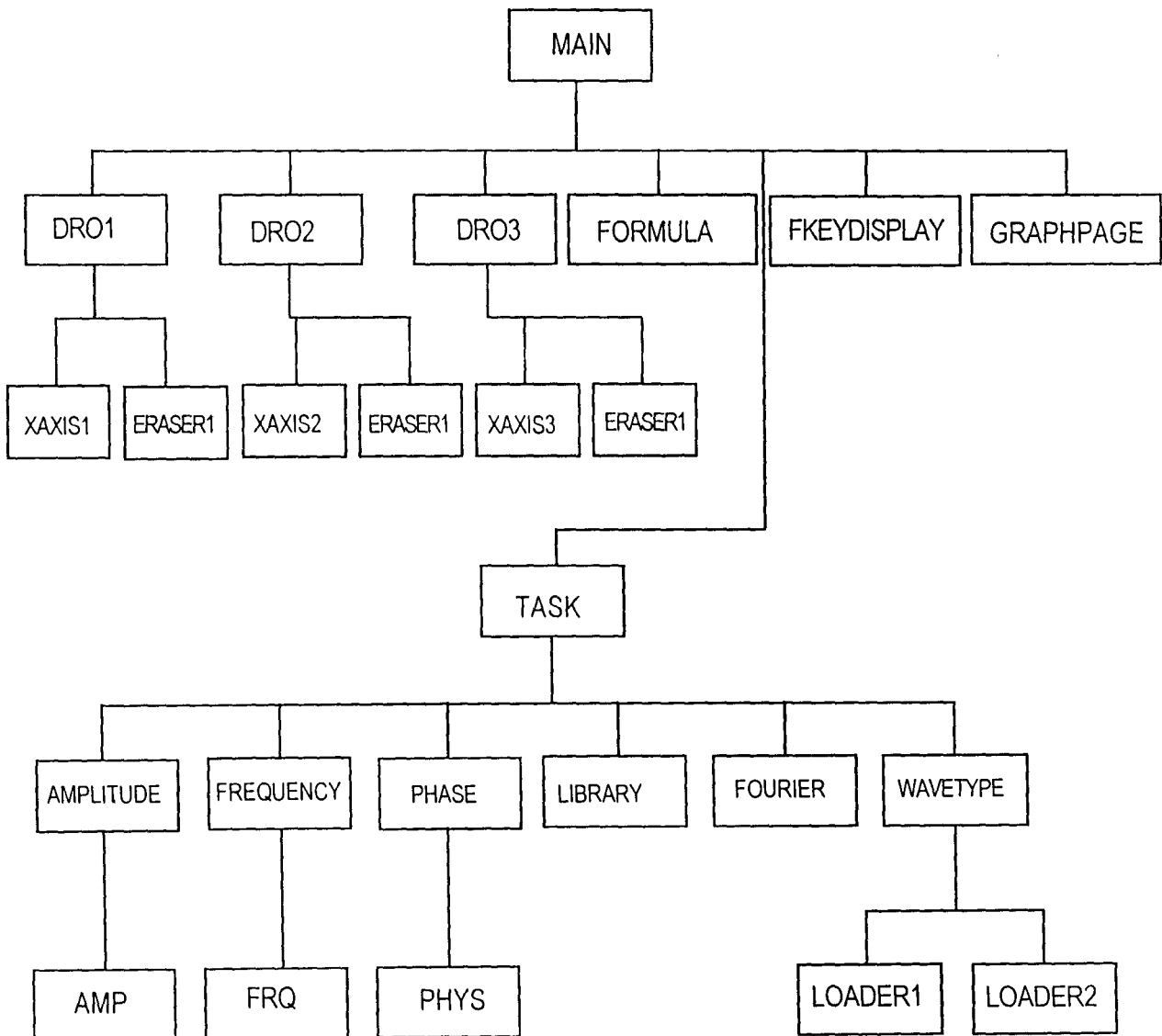


FIG. 3.2

The title given to each subprogram is suggestive of the function of the subprogram. At the beginning, declaration of all subprograms are made in line with syntax rules, followed by the declaration of the "common" global variable and dimensioning of arrays. Other variables are also initialized to set the initial conditions for the WAVE - DRAW environment. The "loader" subprograms are then CALLED to transfer data from wave-function files to array. Relevant names of keys that have been programmed to invoke subroutines are displayed.

Having set the preliminary conditions, the DRO subprograms are now called to draw graphs. Other utility subprograms print the formulae of the equations, phase, and other messages.

The user now specifies changes to be made to one or the two basic waveforms. More than one change can be made at a time to save time before displaying the waves. The previous wave need to be erased before drawing another one. A subprogram **eraser** is called from within the corresponding DRO subprogram. The user may wish to clean the whole drawing area at once, to minimize the finite time required to erase each wave in turn. This can be done by pressing a function key (F12).

3.1.4 FOURIER WAVE DRAW

This facility is provided as tool for mathematicians and physicians who may wish to carry out experiments to observe how complex functions can be broken down (or built up) to simpler forms. It equally serves to verify the Fourier series analysis of waves.

Declaration of subprograms, arrays and initialization of variables are done at the of the beginning of the program. The user is then given the option of either manually (FOUMANUAL) specifying the parameters that govern the wave combinations such as amplitude and frequency, or allowing the program to do necessary calculations (FOUAUTO), before generating the waveforms. However the auto-generation facility can only cope with three types of waves namely Square, Triangular, and Sawtooth waveforms.

FUND subprogram generates and prints the fundamental frequency on the

screen. Once this is done, the harmonics are generated, added and printed on the screen by the DRO2 and FADD subprograms.

When the required waveform is obtained, (or otherwise), a model of the type required is printed for comparison. If the user wishes, the cumulative formulae of the equation of all the components can be printed for perusal.

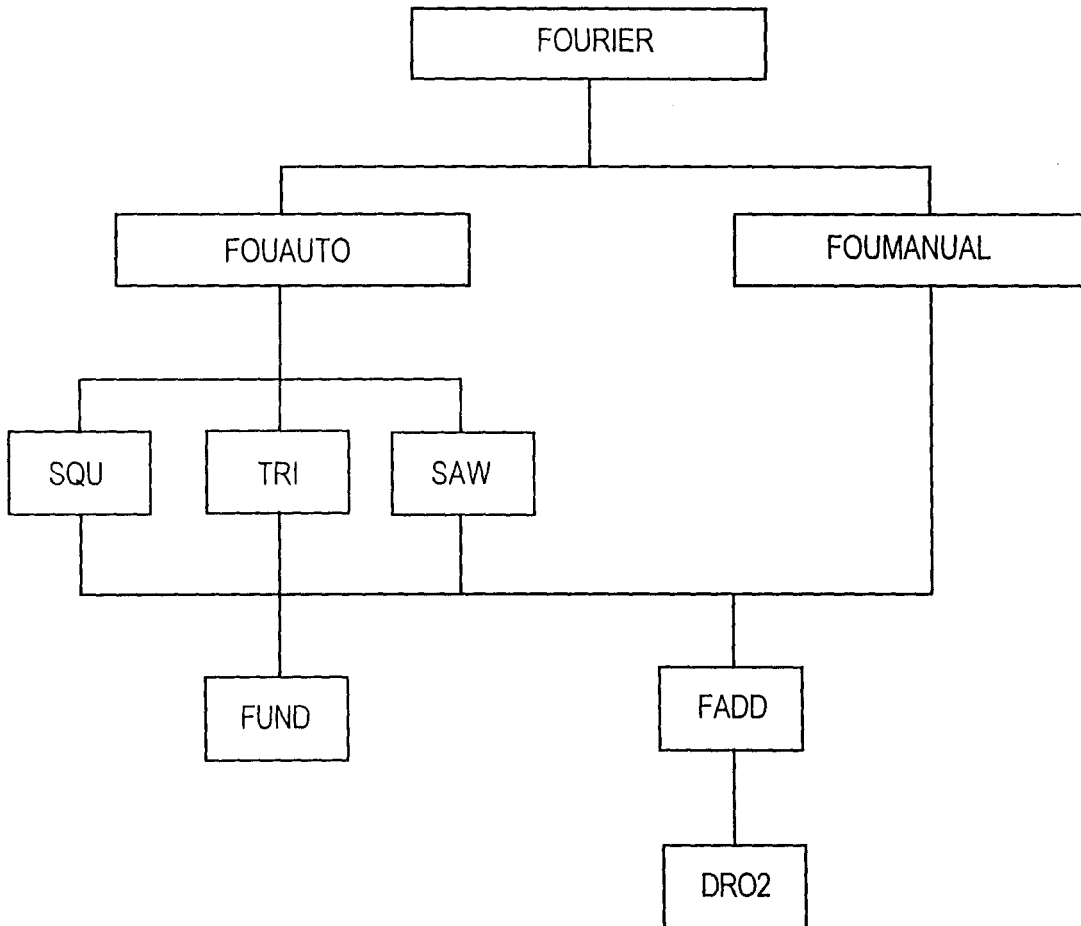


FIG. 3.3

3.2 ACCESSORIES

These are mainly subroutines and a few subprograms that serve as utilities for the entire program. They include :-

- (i) Prompt - to display the message, "Press any key to continue.."
- (ii) Klean - to clear a specified portion of the screen.

CHAPTER FOUR

PROGRAM DEVELOPMENT AND SOFTWARE IMPLEMENTATION

4.0 GENERAL

Modular programming syntax rule in QBASIC requires variables that will be shared between modules to be declared at the start of each subprogram. This distinguishes them from local variables and also gives each subprogram access to the values of these variables. A global declaration is also done within the main module for the passing of arguments (variables), to chained program(s). It will be assumed that shared variables have already been declared for any subprogram in the course of discussion of the subprograms in this chapter. This is to reduce repetition.

Static array declarations are done at the beginning of each program. For memory management other arrays are declared in subprograms that require them. The reserved space is re-allocated at the end of such subprogram(s).

Of higher importance to the factors mentioned above is the declaration of all subprograms. However, if this is omitted (or forgotten), QBASIC automatically declares such subprogram. The implication is that it is not as crucial to the programmer than the former two.

4.0.1 GRAPHICS MODE

Screen 12 of the QBASIC graphics mode is adopted throughout the program. In this mode, the screen is divided into 640 horizontal "x" points and 480 vertical "y" points. A location on the screen, called a picture element, PIXEL, can be specified by its (x,y) coordinates. There cannot be negative coordinates. Moreover, the vertical coordinates increase from zero at the top to 480 at the bottom. Similarly, the horizontal coordinates starts with zero at the extreme left to 640 at the extreme right.

Sixteen different colours are supported in the graphics mode. Each colour can be chosen (using appropriate syntax) by picking a number between 0 to 15. Zero corresponds to black while 15 corresponds to double intensity white. Only foreground color can be specified in this mode.

4.1 PROGRAM NAME

As characteristic of "professional" software packages, the program starts by displaying the name given to the program "WORLD OF WAVES", using letters generated with waves. This is achieved by dividing the screen into 5 horizontal and 4 vertical parts. Only the last three quarters of the vertical division are used in printing "WAVES". A letter occupies each "square". The program segment is listed in the WAVES.DRV program (appendix B).

4.2 CONTROL MODULE

Figure 4.1 below illustrates the logic of the control module stored in a file named WAVES.DRV .

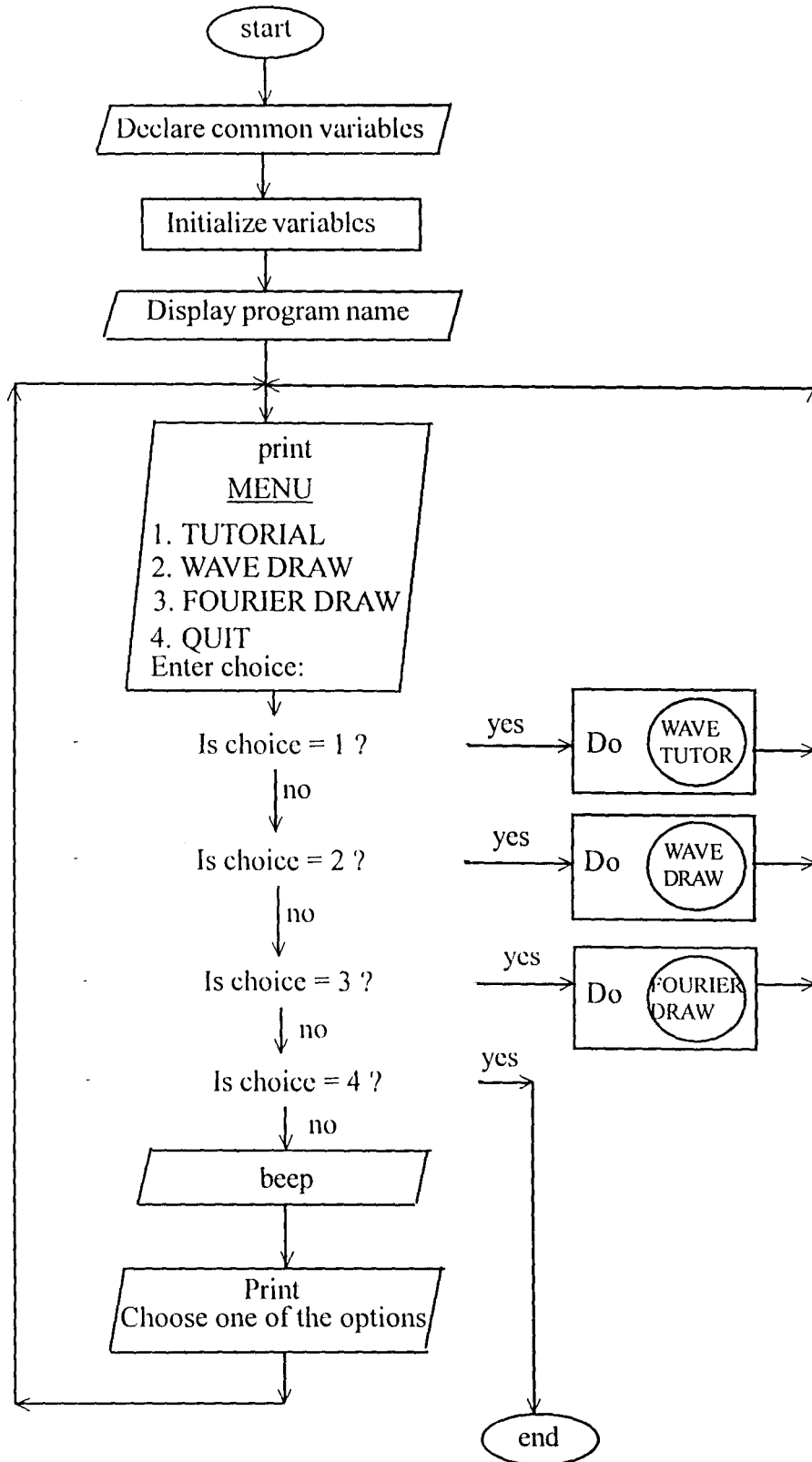


fig. 4.1

After declaring global variable(s) and initializing other variables, the program name is displayed for a brief period (about 2 seconds). The MENU follows, with the user being prompted to choose from the options. If a key is pressed that does not correspond to one of the options, a high frequency sound (beep) is emitted to indicate that a wrong key had been pressed. The user is given another chance to choose one of the options. This will continue until one of the options is selected.

Immediately one of the options is selected, the program corresponding to the option is invoked. Control is restored to this module at the end of the program (or before if so wished).

4.3 THE TUTORIAL PROGRAM

The section starts with a brief display of the title "WAVE TUTOR", the characters of which were generated using the ASCII (extended) code.

The tutorial runs as if the user is reading a book page by page. A page in this case corresponds to a screenful of information, which consists of text, diagrams and (or) simulations. Normally, the computer display is designed to scroll out text to provide room for others that will follow. This becomes a nuisance for this purpose as the user does not have the time to read anything before they are scrolled out. To stop scrolling, an input command characteristic is exploited. The computer is forced to wait for a key to be pressed before proceeding to the other instructions. Hence the user has sufficient time to read and digest what is displayed on the screen. An added advantage of this is the option of deciding whether to proceed to the next page, review the previous page or quit the tutorial. This choice will need to be made at the end of every page. To reduce repetition, a subroutine is created to cater for this. The subroutine is titled "continuity" (fig 4.2 below).

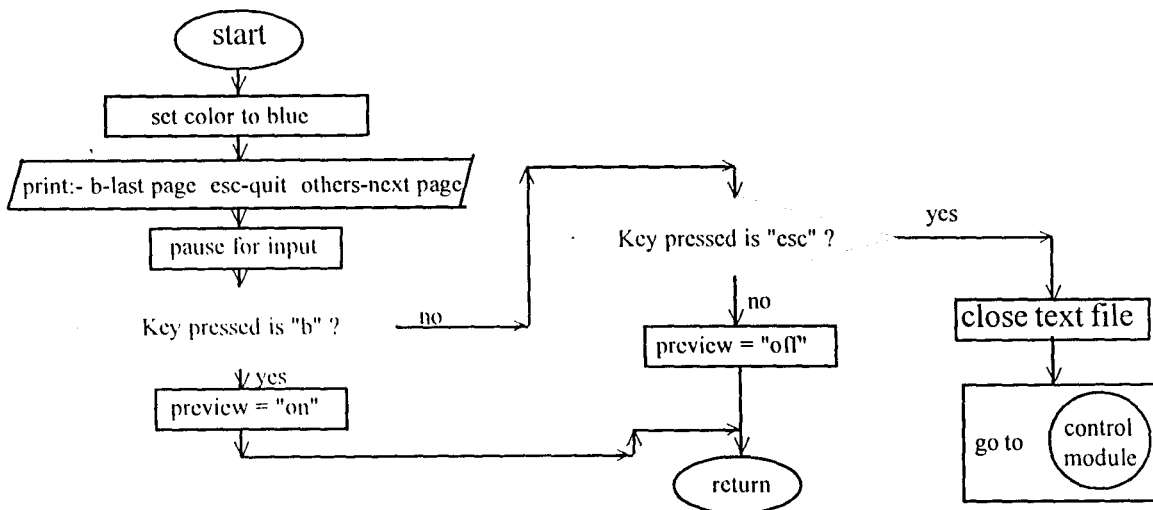


fig 4.2

The text, contained in a separate ASCII text file, need to be "loaded" bit by bit. The text is pre-written in the file using free lines (lines on which no text is printed) to separate paragraphs and other lines of text whose attribute is different from that of others or that need to be printed in a special way. A bit will then consist of lines of text not separated by a free line. Again since the "loading" will be repeated many times, another subroutine (Inputtext) takes care of this (fig 4.3).

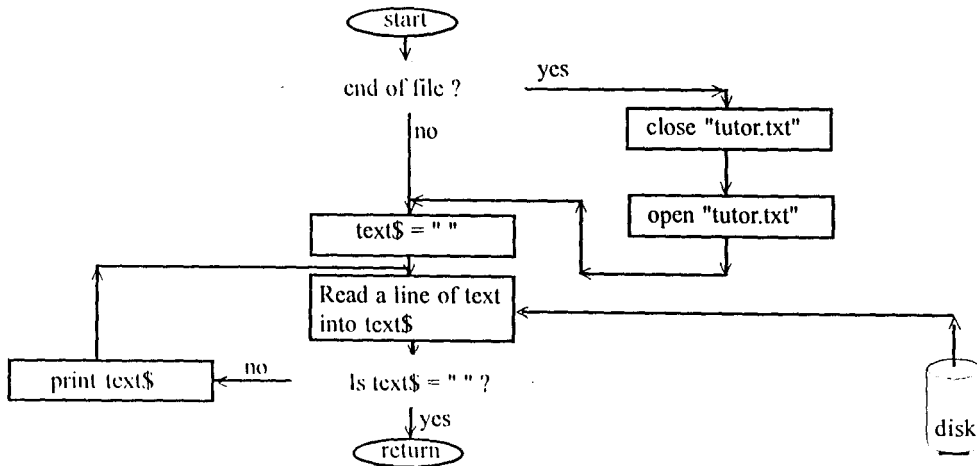
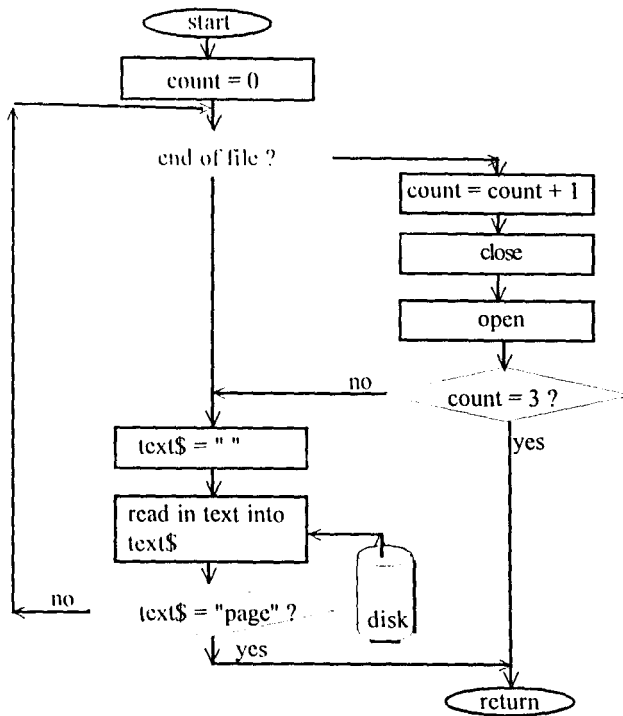


fig 4.3

The last subroutine in this program segment synchronizes the pages of the program and that of the



text file. It also serves the purpose of making reference to a page possible in the text file, at least for the purpose of review (fig 4.4). Simulations are generated by subprograms, pendulum, lmotion, cmotion and vmotion. The overall program flowchart is shown in figure 4.5 below.

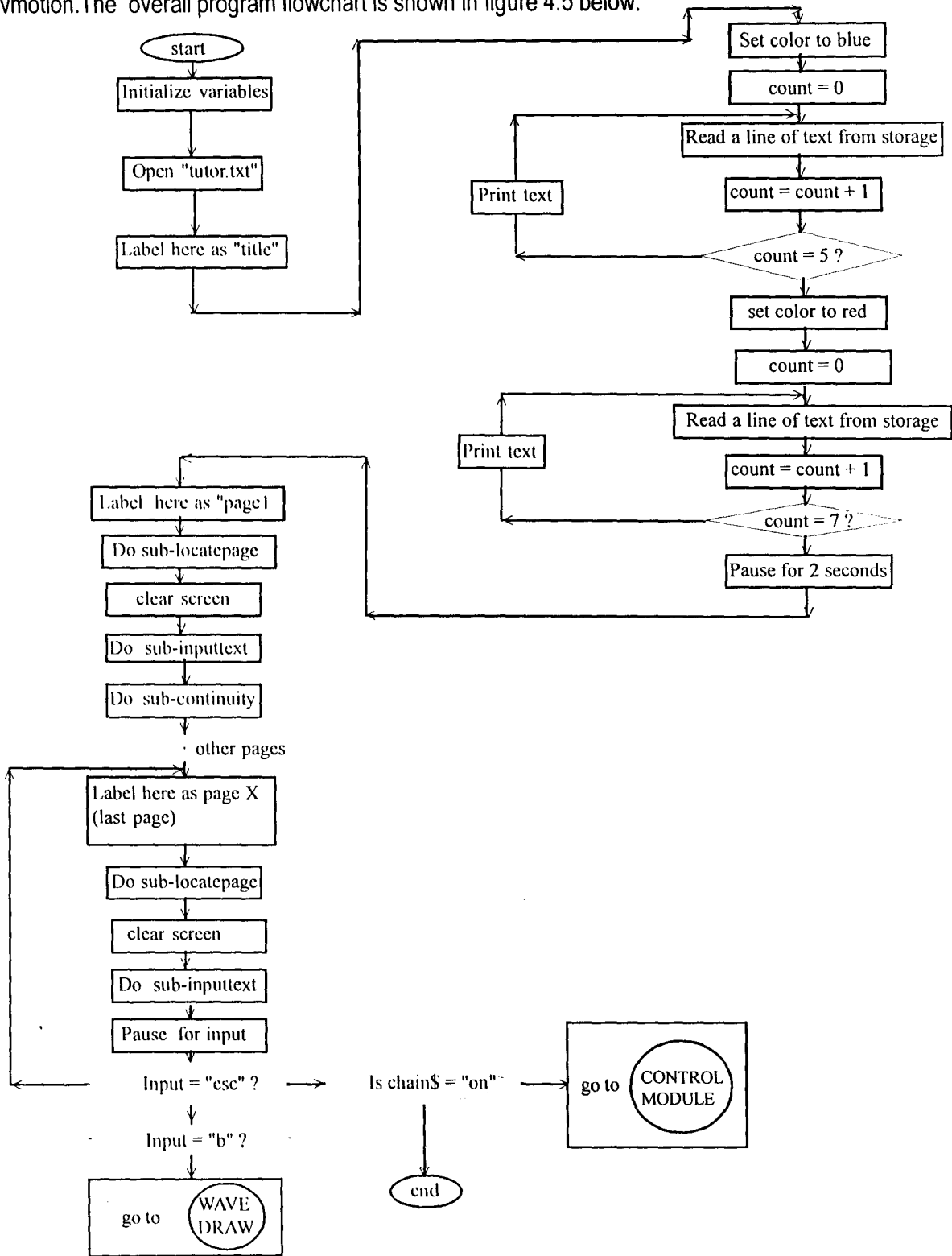


fig 4.5

4.4 WAVE DRAW

The program starts with declaration of all subprograms within the module followed by the declaration of the global variable chain\$. A few variables are initialized to set the environment.

4.4.1 Menu display

Subprogram fkeydisplay (listed in the appendix) is responsible for displaying the MENU i.e. words that point to functions that can be carried out while the program is on. A letter in each word is highlighted by printing it in red. This suggests to the user the key to press to accomplish a particular function. Each function is usually carried out by one or more other subprograms. They include :-

- (a) Amplitude - to alter the amplitude of waves.
- (b) Frequency - to vary the frequency
- (c) Phase - to vary the phase
- (d) Wavetype - to select another type of waveform
- (e) Library - to view a chosen wave effect e.g. modulation.
- (f) Fourier - to invoke the program for Fourier series analysis.
- (g) Clipping - to make a selected wave clipped
- (h) Decay - to make a selected wave damped.
- (i) Rub_off - to clean the graphics area of the screen
- (j)

For mathematical analysis, a graph is drawn on the screen before the waves are drawn on top. This is done by the subprogram graphpage.

The waves are drawn by three subprograms, DRO1, DRO2, and DRO3. Initially all three waves are drawn to aid the user. A new wave will need to be drawn if the user changes the attribute of one or both of the basic waves. In order not to congest the drawing area, which can make the waves incomprehensible, the previously drawn waves will need to be erased before another one can be drawn. Each drawing subprogram therefore has a corresponding erasing subprogram (ERASER 1, 2 and 3). A logic is incorporated such that if any of wave1 or wave2 is modified without the other, only that wave and the third wave will be erased and redrawn. This saves a third of the drawing time.

At the end of each drawing, the program pauses to allow the user to make changes. A subroutine TASK determines the subprogram to call depending on the key pressed which in turn should depend on the need of the user. Figure 4.6 is the flowchart for the main part of the program. Subroutines TASK and DROR

are expressed in pseudocode below.

begin
ch1 = 0 , ch2 = 0

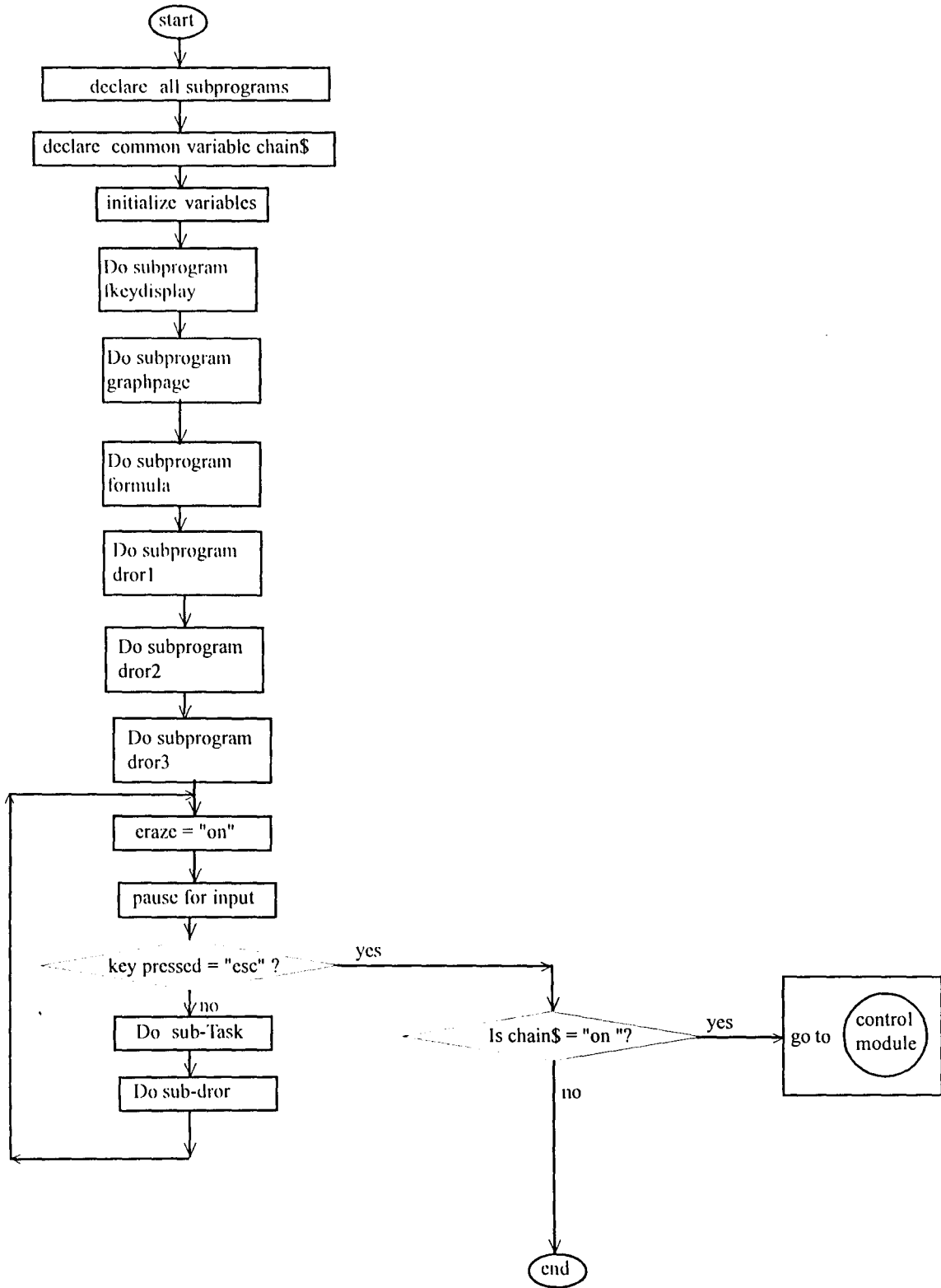


fig 4.6

```
Repeat
  task = key pressed
  select case task
    case "w"
      Do subprogram to change type of wave
    case "a"
      Do subprogram to change amplitude
    case "f"
      Do subprogram to change the frequency
    case "p"
      Do subprogram to change the phase
    case "l"
      Do subprogram to display a selected wave effect as example
    case "m"
      Do subprogram that will draw all waves at the same position
    case "o"
      Do program FOURIER DRAW
    case else
      beep
  end select
end
  print "press 'enter' if no more changes"
  Pause for input
  task = key pressed

loop until task = 'enter'

end
```

Pseudocode for subroutine DROR

```
begin
  If ch1 = 0 AND ch2 = 0 then
    dror = 0      rem indicator to draw nothing
    print "No changes made "
    Pause for 1 second
  else if ch1 + ch2 = 2 then
    dror = 3      rem indicator to draw the three waves
  else if ch1 = 1 AND ch2 = 0 AND dror is NOT equal to 1 then
    dror = 4      rem indicator to draw waves 1 and 3 only
  else if ch1 = 0 AND ch2 = 1 AND dror is NOT equal to 2 then
    dror = 5      rem indicator to draw waves 2 and 3
  end if
  select case dror

  case 1
    Do subprogram dro1 rem draw wave 1 only
  case 2
    Do subprogram dro2 rem draw wave 2 only
```

```
case 3
  Do subprogram dro1
  Do subprogram dro2
  Do subprogram dro3
case 4
  Do subprogram dro1
  Do subprogram dro2
case 5
  Do subprogram dro2
  Do subprogram dro3
end select

end
```

4.4.2 Drawing of graphs

The heart of the subprograms that draw is the command PSET that instructs a computer to draw a point at a specified location on the screen. However a lot of factors need to be taken into consideration, in order to determine the actual values of the coordinates of the point.

4.4.3 Speed and Resolution

For a coordinate (x,y), y is usually dependent on x. Its value can be calculated from a given equation relating the two variables. Periodic functions like sine, cosine, triangular, square, and sawtooth, are expected to be generated in this program. A typical equation is $y = A \sin f(x)$. For a cycle, x varies from zero degrees to 360. But there are 640 horizontal points and 480 vertical points (screen 12 mode of QBASIC graphics). For maximum optimal resolution, a one to one degree to pixel is desirable. If x is varied in steps of 0.5, this gives rise to 720 values, which is more than required.

Apart from drawing it is necessary to reserve some parts of the screen for other displays of information like formulae, interactive messages etc. Therefore, not all of the 640 by 480 pixels are used for drawing. Consequently, if all the 720 points are drawn for every cycle, a lot of points will be repeated for the same pixel, i.e. the points will just fall on one another. As the frequency increases, the situation becomes worse, in terms of the valuable time, wasted in printing points that will not make any difference to the resolution of the image. Moreover considering that there are usually three waves to print at a time, it becomes a real waste of computer time.

A solution to the problem is to incorporate a way by which some points are skipped while drawing. For example, if every other point is skipped, that is a step of 2, then 360 points are printed in a cycle. The drawing time is reduced by half. For higher frequencies, it may be necessary to skip by 6 or even 10 to optimize speed.

However, gain in speed can sometimes be at the expense of the resolution. For a frequency f of low amplitude, the resolution may be fine. If the amplitude is increased, gaps appear along the curve, indicating that all the pixels have not been used up. Therefore, for optimization of resolution, each pixel along the horizontal and vertical directions must be utilized. For speed, no two points must be printed on the same pixel.

The greatest changes in y values occur between $x = 0$ and $x = 70$ (for sine and cosine functions). By multiplying each value by the amplitude and taking their average, an incremental value q is obtained.

$$q = (\sin 70 - \sin 0) / (70 \times \text{frequency}) \text{ ————— eqn 4.0}$$

But q for each pixel should be 1. Therefore the step (called RESO, to avoid using a QBASIC library name) is given by

$$\text{RESO} \times q = 1 \text{ ————— eqn 4.1}$$

$$\text{RESO} = 1 / q \text{ ————— eqn 4.2}$$

For some waves, e.g. square waves, $q = 0$, which will give rise to an infinitely large value of RESO (a computation error which will stop the program from running). In this kind of situation, q is programmed to take a minimum value of 0.2.

Another factor that also contribute to speed is the calculation of values of y . To draw a curve, the equation

$$y = A \sin f(X \cdot \pi / 180)$$

will need to be evaluated for each of the 721 points (zero included). This takes a little "computer time" especially if the frequency f is large in which case the calculation will be done " $f \times 721$ " times. This problem can be overcome if the values are pre-calculated and stored in an array (since they are repetitive for a particular function). For the other functions, four other arrays are necessary.

4.4.4 Memory management

Each wavetype considered in the program requires its own set of y values. Hence five arrays each consisting of 721 values need to be set up. Moreover, due to the design of the drawing subprograms, another array need to be set up, for data transfer from any one of the chosen wavetypes. Since there are three drawing subprograms, another three arrays are needed to do this in addition to other parts of the program requiring memory space, the need to find other types of storage becomes expedient.

The solution is to transfer data that is not used frequently to permanent storage (hard disk) for retrieval when needed. Though the time for "loading" of data is longer than if it were in an array, the gain in

memory space management (and avoidance of critical errors) far outweighs the loss in speed.

4.4.5 Amplitude

The maximum value of y as obtained from a sine or cosine table is 1 while the minimum is -1. If these values of y are directly used in specifying the pixel coordinates, an approximate straight line is obtained, because the values are so low that there is no significant difference in the printing of y on the screen. Therefore, there is a need to multiply these values by a constant. Depending on the chosen value of the constant, y values will become significant enough for a curve to be visually comprehensible. The constant is the amplitude of the wave.

The amplitude must not be so large that the value of y will eventually be greater than 480, which is the last y point that can be printed on the screen. Any such error occurring while the program is running will lead to a critical error sufficient to terminate the program. To protect against any such occurrence and in addition to the fact that part of the lower part of the screen will be used for some text displays, a condition is imposed that any value of y greater than 450 (and less than 50) should not be printed.

Subprogram AMPLITUDE is dedicated to varying the value of amplitude as desired. It is listed in appendix C.

4.4.6 Vertical offset

The topmost pixel on the screen correspond to a y position of zero, which is the minimum the computer recognizes. (Any negative value of y will generate an error). This implies that the negative values of a function, must be processed before they can be printed.

If a certain value, $yoff$, greater than the absolute of the maximum negative value is added to y , then all the points become "positive" as far as the computer is concerned. Since it is an addition, the values of all positive values are also increased by the same amount. The relationship between the points is preserved. Carrying this a little further, the wave can be "pushed" to an appropriate position on the screen by adding the necessary offset value.

The three waves are given offset values

$$yoff1\% = 150, \quad yoff2\% = 250, \quad \text{and} \quad yoff3\% = 350.$$

For comparative analysis it is sometimes desirable to put the waves on top of one another. This can be achieved by resetting the offset values of two of them to that of the third central one so that all of them will be at the centre of the screen.

4.4.7 Horizontal offset

Better appreciation of the waves can be achieved if equations, and or text are further used to describe them. Space is therefore reserved to the left and right of the screen. The first few horizontal points $x = 0$ to $x = \text{xoff}\% = 140$, and the last few points $x = 500$ to $x = 640$, are reserved. No drawing take place on these places. However if necessary (as in the WAVE TUTOR program), the values may be adjusted. This leaves a drawing area $\text{xlen}\%$ consisting of 360 horizontal points.

4.4.8 Drawing area

The $\text{yoff}\%$ and $\text{xoff}\%$ values do not affect the original relationship between x and y . Recall that the values of y for x varying between zero and 360 degrees taken in steps of 0.5 is precalculated.

The program segment, in pseudocode, is:-

```
For i% = 0 to 720 step reso
    x = xdummy x i% + xoff%
    y = ampl x ydummy(i%) + yoff%
    If y > 50 AND y < 450 then
        plot (x, y)
    end if
Next i%
```

$\text{ydummy}(i\%)$ represents the precalculated value of y stored in the i th position of the array ydummy . Xdummy is obtained by dividing the horizontal drawing area into 720 places.

In general, $\text{xdummy} = \text{xlen}\% / (720 \times \text{frequency})$

where

$\text{xlen}\% =$ horizontal length of the drawing area,

$\text{ampl} =$ amplitude

4.4.9 Frequency

The frequency of a wave is the number of cycles per second. If it is assumed that the length of the drawing area corresponds to 1 second, then for f waves to be drawn, the drawing area is divided into $\text{xlen}\% / f$ places. A wave can consequently be drawn in each region simply by subdividing each region into 720 places. This gives a unit of x to be

$$\begin{aligned}\text{xdummy} &= (\text{xlen}\% / f) / 720 \\ &= \text{xlen}\% / (720 \times f)\end{aligned}$$

The FREQUENCY subprogram is responsible for varying f . It's logic is similar to that of subprogram amplitude.

4.4.10 Phase

Phase difference is accomplished by altering the starting position in the y table. A positive phase change corresponds to starting to print at $2 \times x$ position in the y table. (Recall that x is stored in steps of 0.5). However, after the 720th position in the table, a means must be found of making the printing to start again at the zero position in the y table. The counter is reset to zero, since the retrieval of the y values depends on the counter number. This program segment falls within the DRO1, and DRO2 subprograms, which will be discussed shortly.

Subprogram PHASE varies the phase as required. Its structure is similar to that of subprogram AMPLITUDE.

4.4.11 Decay

In practice the amplitude of oscillation of a vibrating object decreases with time if the source of energy is removed. To implement this, the amplitude is gradually reduced at a specified rate as x varies from zero to 360 degrees, for one cycle. It becomes more complicated for larger frequencies. The subprogram DECAY handles the necessary initialization of variables, for the DRO1 or DRO2 subprogram which implements it.

4.4.12 Clipping

A clipped waveform is obtained by approximating all values of y greater than a certain value called limit%, to that value.

$$y = \text{ampl} \times y_{\text{dummy}}(i\%)$$

If $y > \text{limit}\%$ then $y = \text{limit}\%$

The limiting value limit% is specified through the subprogram CLIP. It is listed in appendix A.

4.4.13 The "DRO" Subprograms

Two different types of waves may be drawn simultaneously via subprograms DRO1 and DRO2, which are structurally similar. A third subprogram, DRO3, adds (the displacements y of) the other two. Most of the logic behind subprograms have been discussed above.

For DRO3, most of the conditions that determine the nature of the waves in 1 and 2 e.g. clipping, damping, amplitude etc., are absent. DRO3 simply takes the final values of y1 and y2 that are used for printing and adds them together to obtain its own value of y. The only basic difference is its rigid structure in terms of the number of

To erase a printed wave, the eraser subprograms must follow the same path as that of the ones that printed the wave. Therefore, all the attributes of the wave e.g. frequency, amplitude, etc., must be known. Since

the values of these variables can be accessed, by the eraser subprogram, this should not be a problem. But it is quite possible that those attributes have been changed, in preparation, for the drawing of a new wave. Then it will almost be impossible to erase the previous wave.

The solution is to store the former values of y with which the values were printed in an array. Knowledge about the period and resolution must also be stored. Luckily, the latter two require only single element arrays.

The ERASER subprograms' structure, is similar to that of the corresponding DRO subprograms, without conditions and tests. It is listed in appendix C.

4.4.15 Loading

Whenever a different type of wave is desired, it is necessary to change the initial y values stored in the y dummy array. Each function has its set of values stored in a file on the magnetic disk (permanent memory). The retrieval of these values is done by subprograms LOADER1 and LOADER2 for wave1 and wave2 respectively.

The type of wave needed can be chosen, by pressing the letter "W", which makes the program branches to a subprogram WAVETYPE which presents a MENU of the types of waves, and calls the appropriate LOADER after a choice is made. (See appendix C).

4.5 **FOURIER DRAW**

Most of the program segments in the FOURIER DRAW are similar in logic to those of WAVE DRAW described above. It is listed in appendix D.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.0 SYSTEM EVALUATION

The program was originally written as a single program consisting of several subprograms and subroutines. During test-running on a 80286 computer with 640 kb base memory and 1024 kb extended memory, memory exhaustion error messages often popped up. There were lots of arrays, both fixed and dynamic. As first corrective measure, data contained in arrays were transferred to files (the .tbl files). Subsequently, the error messages decreased in frequency. However, as the program grew in size, the memory exhaustion messages grew with it. Evaluation of the system again showed that while a certain section of the program is running, other sections are more or less redundant, thereby tying up computer memory. This prompted the idea of breaking up the program into modules that are linked by the CHAIN command. The sections are separated by single stand-alone units, though sharing some resources (tables, global subroutines, etc.) of the entire package.

An additional creation of a module was necessitated by the passion for a trademark for the package. Part of the "tradename" is written with "waves" while the rest are just normal characters. Since the name is written once at the start of the program, there is no need to tie up the computer memory with the rather lengthy program segment. This, and the MENU, make up a module which remains in memory only long enough for the user to make a choice. Once a choice is made, the program invokes (using the CHAIN command) another program, which now replaces the former in memory. Since the MENU program will invariably be needed again (it is more or less the control program), each of the other programs contain commands that returns control to it when the user wishes to quit or use other parts of the package.

5.0.1 Executable file

It was impossible to obtain VISUAL BASIC, which is the compiler for QBASIC. Normally to run the "World of WAVES" QBASIC will first be loaded. The program file "Waves.drv" is the control program as discussed above. Once it is loaded, within QBASIC, the "World of WAVES" begins to run. This method has the disadvantage that the user will see the program listings, and may be tempted to alter command lines. Other laymen to programming may become totally confused. This is dangerous to the integrity of the package.

To circumvent this problem, a DOS batch file named "wv.bat" was created. Once "wv" is typed at the prompt, "World of WAVES" package starts. Quitting automatically returns the user to the c prompt

without seeing the listings. The batch file contain the following DOS command lines:

```
echo off
```

```
Qbasic /run waves.drv
```

The package appears to be a compiled stand-alone program.

5.0.2 User-friendliness

By repeated testing and debugging, likely “user-invoked errors” are avoided. They include:-

- (a) Wrong answer to input prompts
- (b) Pressing more than a designated number of keys during input routines
- (c) Unexpected division by zero, resulting from pressing “enter” instead of a number
- (d) Entering a number where a character is expected or vice-versa.

5.0.3 Speed

An interpreted program runs more slowly than compiled ones. During program design, evaluations that tend to take time and need to be repeated frequently were calculated at the beginning of the program and stored in arrays. Some were calculated during program development and stored in files. The data is retrieved when needed and stored in arrays. This tremendously enhanced the speed during run.

5.1 EVALUATION OF OBJECTIVES

Printed samples of parts of the program while running are shown in the appendix.

5.1.1 Tutorial

Though a comprehensive study on waves is an enormous task, for which this researcher neither has the time nor the resources to accomplish, the tutorial is able to introduce the basic concepts of waves to laymen and students of Physics and Mathematics while portraying the concept in a new light to other “learned” people. It will serve as a solid foundation on which further knowledge may be built.

5.0.2 WAVE DRAW

This section of the adds to the knowledge gained after going through the tutorial. It also serves as an instructional material for teaching a large class. The teacher, being familiar with its use, generates and displays the waves as appropriate while instructing the students on the basic concepts of waves.

It also serves as a good environment for a do-it-yourself practice session.

1.3 FOURIER DRAW

This section is able to show that complex wave functions can be built up (or broken down) from simple sine and / or cosine waves added at the right proportions of amplitude and frequency. The demon-

stration can show Fourier wave-drawing analysis of Square, Sawtooth, and Triangular waves. However, if the user can work out the amplitude, frequency and type of wave (sine or cosine) of each harmonic component, the computer will use these input parameters to build up the wave.

5.2 LIMITATIONS

1. All waves generated are assumed to have a period of one second. This is not always the case in practice. However, it makes the concept of frequency easier to understand by laymen.
2. The waves are simulations only. They do not represent real-time waves obtainable from electronic wave generators (and displayed by oscilloscopes). The periods and frequencies are symbolic.
3. The FOURIER- DRAW section is not versatile enough to take care of all types of complex waves possible. Only three types, namely, Square, Sawtooth, and Triangular can be automatically generated. Others require little effort by the user in pre-calculating the appropriate amplitudes, frequency ,(etc.) of each harmonic before the program will generate the wave.

RECOMMENDATIONS

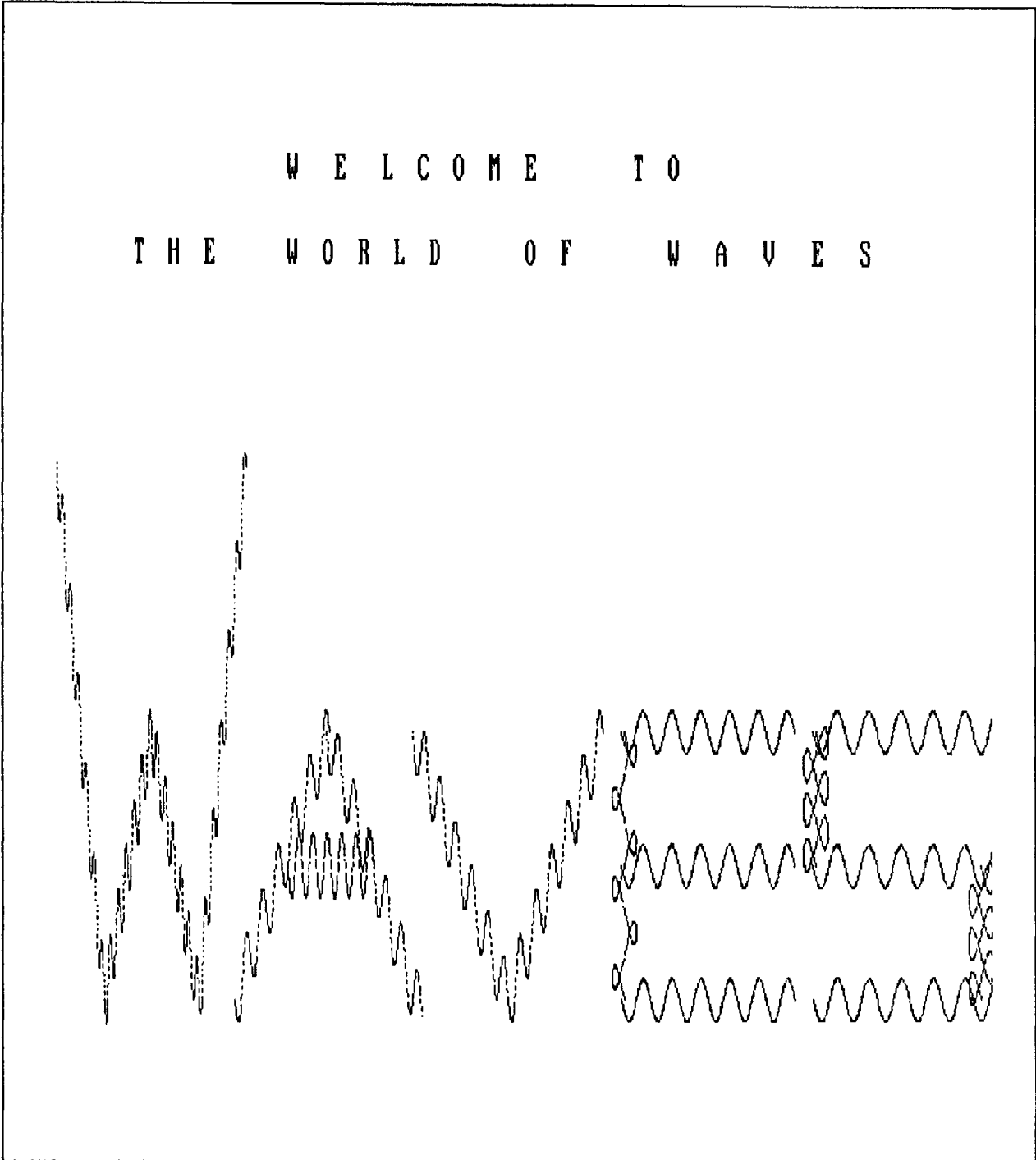
1. Efforts should be made to acquire the QBASIC compiler VISUAL BASIC. This will make the package become a stand-alone (i.e. not requiring the QBASIC environment).
2. Further development of the package, should be undertaken in future. The FOURIER-DRAW section should be made more user-friendly and capable of handling any desired complex wave function.
3. Waves drawn cannot be printed directly from the package. In order to obtain the graphics diagrams shown in this write-up, the package had to be run under Windows 95 environment, by exploiting the cut-and-paste, bitmap graphics capability of the Windows environment. A means should be found saving, loading and printing of the graphics image in future. (Note:- Though the print-screen facility of DOS can be used for printing , it is not possible to print selected areas of the screen. It prints the whole screen all the time which is sometimes undesirable.

REFERENCES

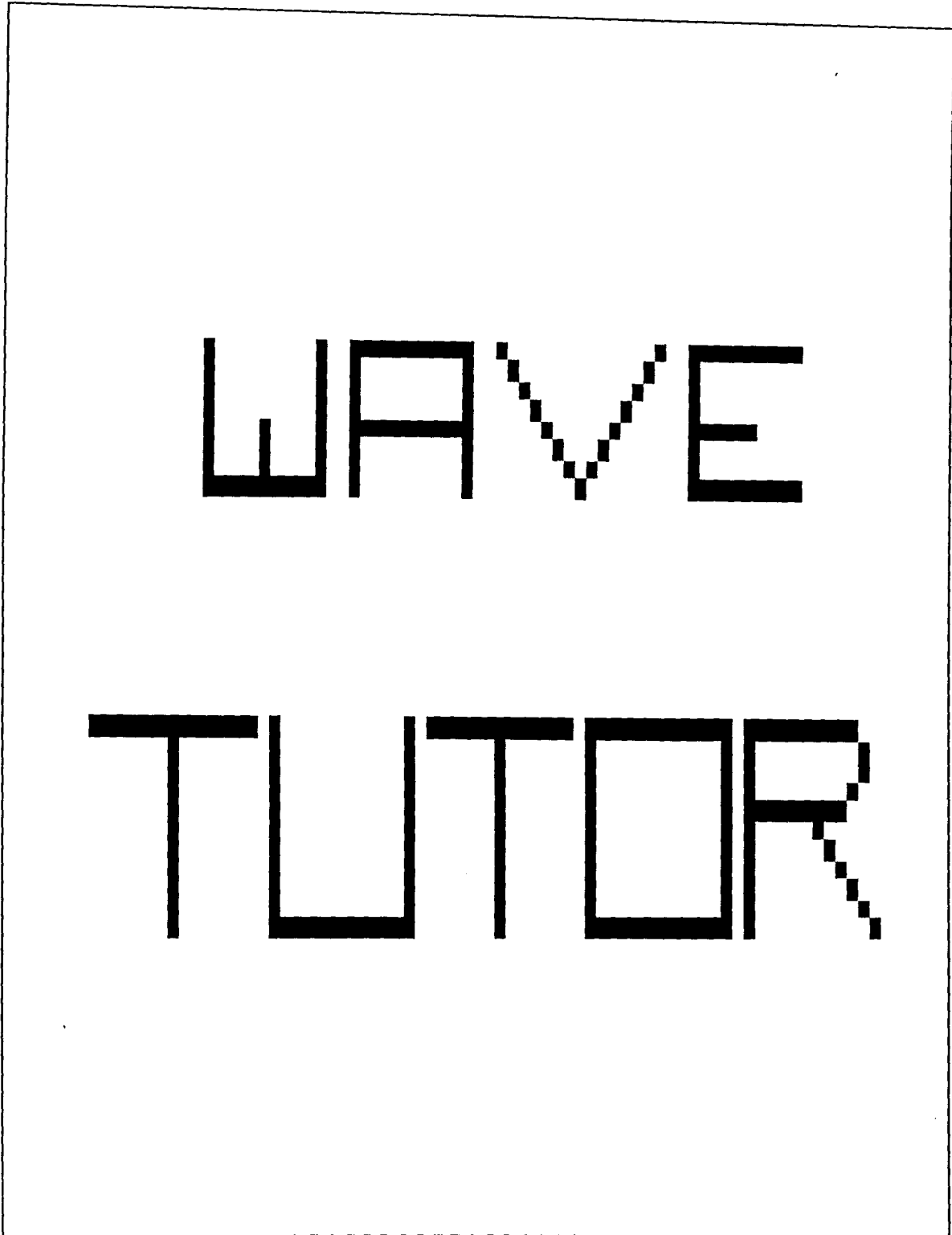
- ARYA, A.P. (1979) Introductory College Physics. New York. Macmillan Publishing Co., Inc.
- GOTTFRIED, B.S. (1986) Schaum Outline Series: Theory and Problems of Programming with Basic. Singapore. Mcgraw-Hill Book Co.
- KING, R.A. (1986) The MS-DOS Handbook. Berkeley. Sybex Inc.
- NELKON AND PARKER (1987) Advanced level Physics, Sixth edition. Heinemann Educational Books (Nigeria) Ltd.
- OCHIE-OKORIE (1993) Computer Fundamentals - Introduction and Utilization. Solid Rock Computers Press.
- SOLYMAR, L. (1988) Lectures on Fourier Series, Oxford. Oxford University Press.

APPENDIX A

PRINTED SAMPLES FROM PROGRAM WHILE RUNNING



SAMPLE 1 - General opening screen



SAMPLE 2 - Opening screen for the tutorial section.

CHAPTER 1

=====

INTRODUCTION

Probably the most essential object to life as we know it is the SUN. It enables us to see during the day as well as providing invisible ultraviolet light, which is essential for our skin to manufacture vitamin D. The sun produces light (and other radiations) travelling at a speed of three hundred thousand kilometres per hour to get to us on earth. This is achieved by wave motion.

Sound and consequently speech is also transferred from one place to another by wave motion. Wireless communication via radio, television, and mobile telephones, is achieved by wave propagation.

However, despite this preponderance of waves around us, many are still intrigued by the nature of waves. One of the reasons for this, is that wave motion involves atoms or molecules which are microscopic (too small to be seen).

Another reason lies in the nature of the wave itself. A wave exist only when there is motion. Any passive method of learning it cannot suffice.

PRESS:- b -(previous page) esc -(quit) any other key -(next page)

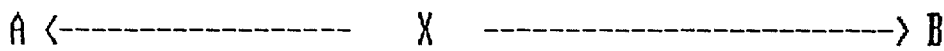
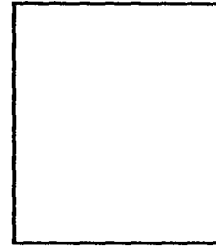
SAMPLE 3 - Page 1 of the Tutorial section

(The colors shown here are the complements of the actual actual colors that appear on the computer monitor.

The main text is actually in white over black background. The same applies to other samples shown.)

Displacement

The most basic type of motion, is movement from one place to another in a particular direction. Press any key.



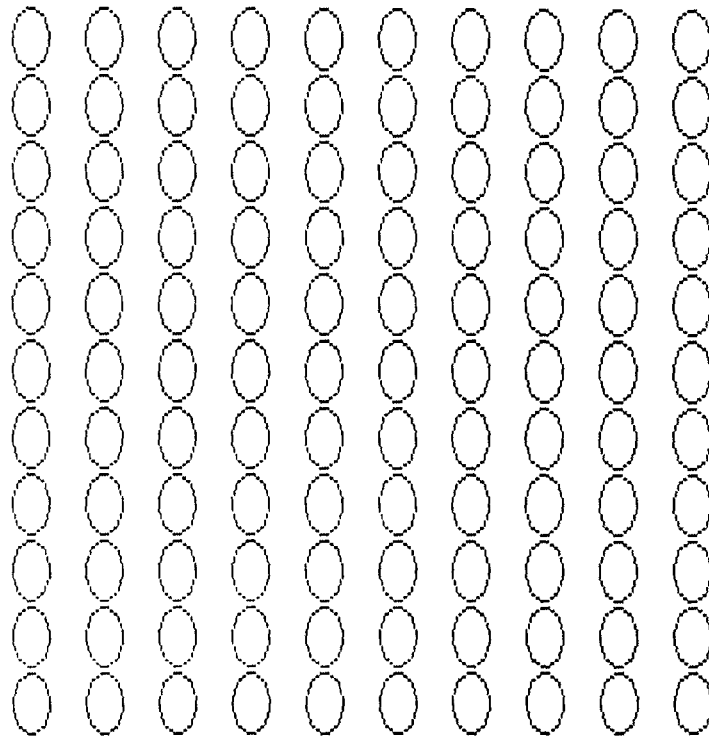
The block has moved from A to B in a straight line. It has been displaced from A to B by an amount X metres.

If the body had moved from A to B following an irregular path, (curved path), then it would have moved by a distance Y metres instead. The basic difference between displacement and distance is the direction. Displacement is a vector quantity (it has magnitude and direction), while distance is a scalar quantity (it has magnitude only).

PRESS:- b -(previous page) esc -(quit) any other key -(next page)

atoms

Natural constraint to free motion or oscillation is exemplified in the arrangement of the atoms. Atoms are the 'building blocks' of matter. They are so small that they cannot be seen. The specific position of an atom in relation to the others is maintained because of interatomic forces depending on the natural arrangement of atoms for a particular substance.



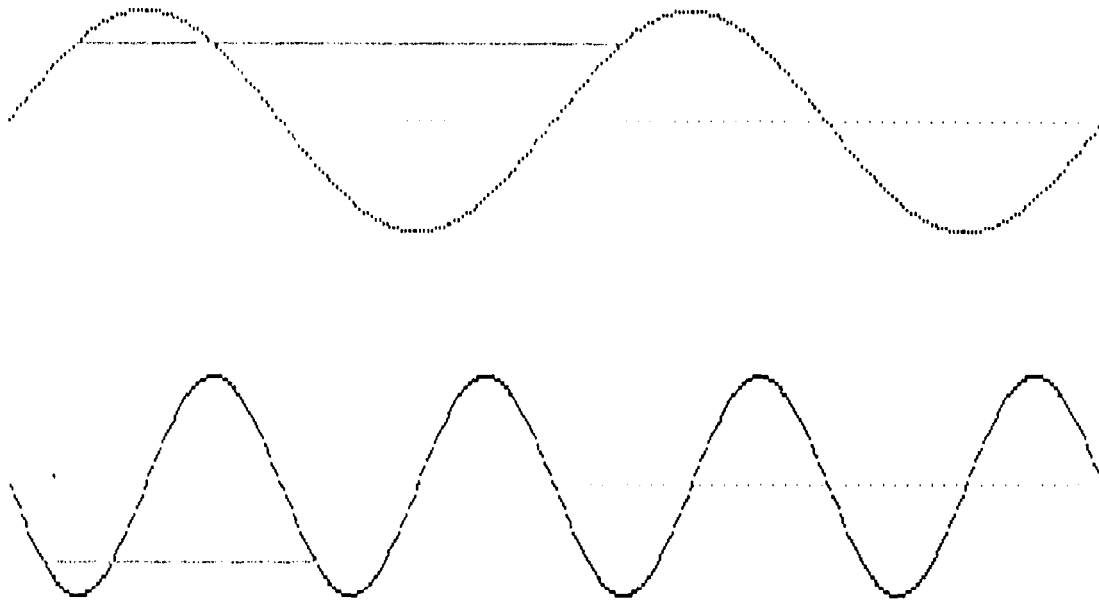
PRESS:- b -(previous page) esc -(quit) any other key -(next page)

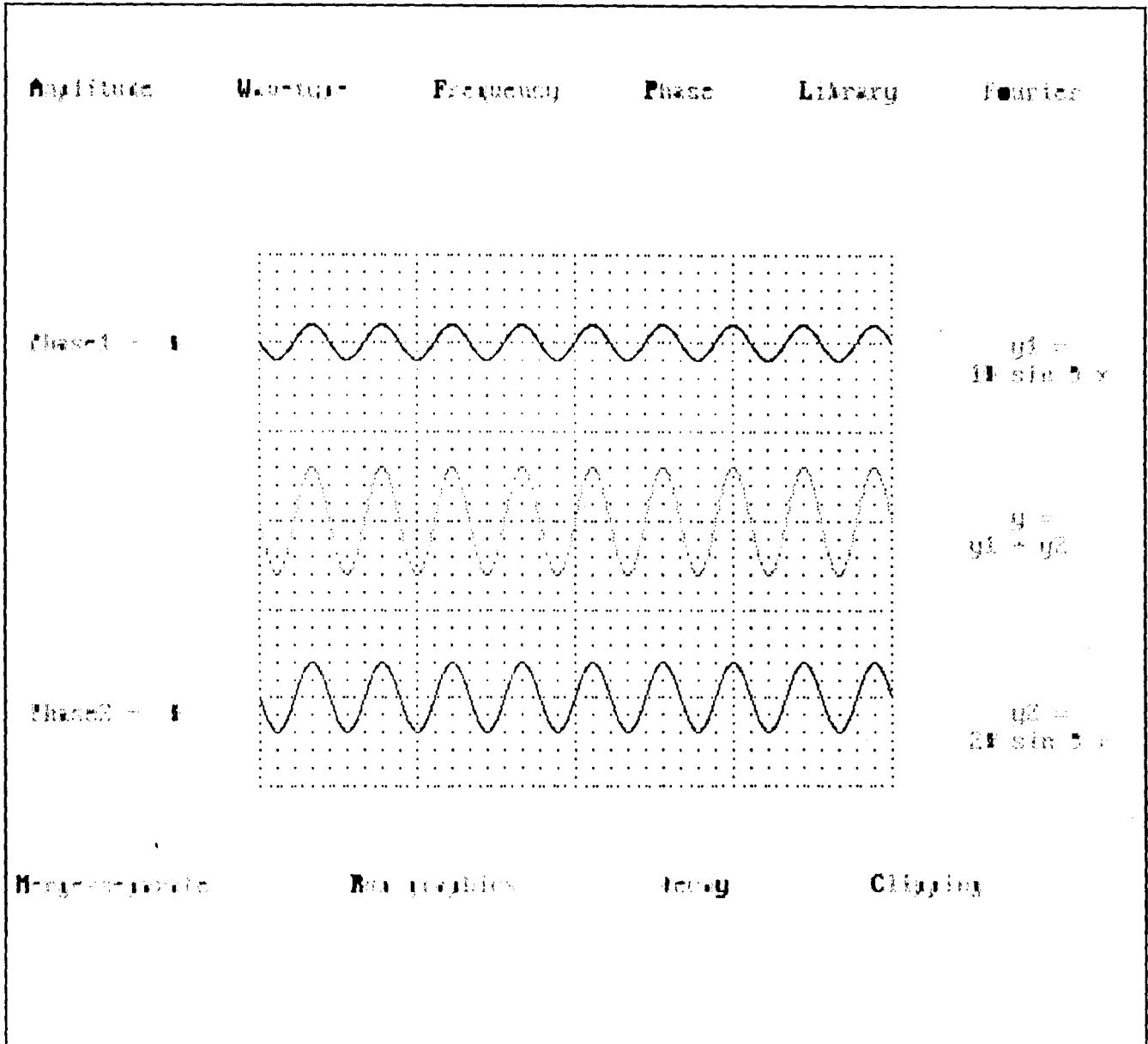
WAVELENGTH

of a wave is defined as the distance between two successive points in phase.

Consider the diagram shown below. The white line is the wavelength, for each wave. It has been drawn from several points in phase with one another. Note that the line is shorter for the red wave. Which of the two waves has the greater frequency, red or green ?

Answer = ? ■



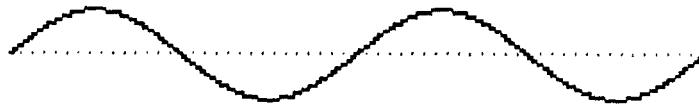


SAMPLE7 - MAIN OPERATING ENVIRONMENT OF WAVE-DRAW SECTION

FOURIER ANALYSIS

Press any key to continue, 'esc' to quit.

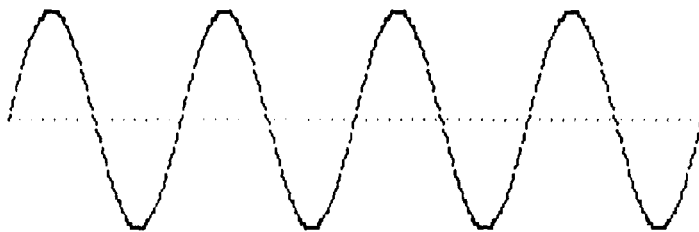
Last addition



New addition



Harmonic 2



$$-5.851236 \sin 4 x$$

SAMPLE 8 - Second stage of building up a sawtooth waveform from simple sine waves, (Fourier-draw section).

APPENDIX B

CONTROL MODULE PROGRAM LIST

```
REM /* CONTROL MODULE FOR PROGRAMS PACKAGE "WORLD OF WAVES" */
REM /* FILENAME IS "WAVES.DRV" */
COMMON SHARED chain$
SCREEN 12 ***** configure for graphics image *****
IF chain$ = "on" THEN GOTO menu
yoff1 = 150: ampl1 = 12: col1% = 4: col2% = 2: col3% = 3
pi = 22 / 7
'write W *****
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (32 / 360) * i% + ampl1
PSET (k, y), col1%
yoff1 = yoff1 + 200 / 360
NEXT i%

FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (32 / 360) * i% + ampl1 + 32
PSET (k, y), col1%
yoff1 = yoff1 - 100 / 360
NEXT i%

FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (32 / 360) * i% + ampl1 + 64
PSET (k, y), col1%
yoff1 = yoff1 + 100 / 360
NEXT i%

FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (32 / 360) * i% + ampl1 + 96
PSET (k, y), col1%
yoff1 = yoff1 - 200 / 360
NEXT i%
'write A*****
yoff1 = 350
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (64 / 360) * i% + ampl1 + 120
PSET (k, y), col2%
yoff1 = yoff1 - 100 / 360
NEXT i%

FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (64 / 360) * i% + ampl1 + 182
PSET (k, y), col2%
yoff1 = yoff1 + 106 / 360
NEXT i%

yoff1 = 300
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (58 / 360) * i% + ampl1 + 156
PSET (k, y), col2%
```

NEXT i%

'write V*****

```
yoff1 = 250
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (64 / 360) * i% + ampl1 + 240
PSET (k, y), col3%
yoff1 = yoff1 + 100 / 360
NEXT i%
```

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (64 / 360) * i% + ampl1 + 304
PSET (k, y), col3%
yoff1 = yoff1 - 100 / 360
NEXT i%
```

'write E*****

```
ampl1 = 8
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = ampl1 * SIN(3 * (i% * (pi / 180))) + 394
PSET (k, y), 5
yoff1 = yoff1 + 100 / 360
```

NEXT i%

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = ((120 - ampl1 / 2) / 360) * i% + ampl1 + 384
PSET (k, y), 5
```

NEXT i%

yoff1 = 300

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = ((120 - ampl1 / 2) / 360) * i% + ampl1 + 384
PSET (k, y), 5
```

NEXT i%

```
yoff1 = 250
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = ((120 - ampl1 / 2) / 360) * i% + ampl1 + 384
PSET (k, y), 5
NEXT i%
```

'write S*****

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (128 / 360) * i% + ampl1 + 512
PSET (k, y), 6
NEXT i%
```

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
```

```
y = u + yoff1
k = ampl1 * SIN(3 * (i% * (pi / 180))) + 522
PSET (k, y), 6
yoff1 = yoff1 + 50 / 360
NEXT i%
```

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (128 / 360) * i% + ampl1 + 512
PSET (k, y), 6
NEXT i%
```

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = ampl1 * SIN(3 * (i% * (pi / 180))) + 640 - ampl1
PSET (k, y), 6
yoff1 = yoff1 + 50 / 360
NEXT i%
```

```
FOR i% = 0 TO 360
u = ampl1 * SIN(6 * (i% * (pi / 180)))
y = u + yoff1
k = (128 / 360) * i% + ampl1 + 512
PSET (k, y), 6
NEXT i%
```

```
COLOR 15
LOCATE 3, 22: PRINT "W E L C O M E   T O"
LOCATE 5, 8: PRINT "T H E   W O R L D   O F   W ";
PRINT "A V E S   "
SLEEP 3
menu:
DO
CLS
COLOR 14:
LOCATE 4, 36: PRINT "M E N U"
LOCATE 5, 36: PRINT "======"
LOCATE 8, 4:
PRINT "Select a number corresponding to the part of this program you want to use."
LOCATE 10, 10: COLOR 12: PRINT "1. ";: COLOR 14: PRINT "TUTORIAL"
LOCATE 12, 10: COLOR 12: PRINT "2. ";: COLOR 14: PRINT "WAVE DRAW"
LOCATE 14, 10: COLOR 12: PRINT "3. ";: COLOR 14: PRINT "FOURIER DRAW"
LOCATE 16, 10: COLOR 12: PRINT "4. ";: COLOR 14: PRINT "QUIT THIS PROGRAM"
LOCATE 20, 10: PRINT "Enter choice (number) = ";
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
choice$ = INPUT$(1)
c% = VAL(choice$)
ch% = ASC(LCASE$(choice$))
PRINT choice$
IF choice$ = CHR$(27) OR choice$ = "4" OR LCASE$(choice$) = "q" THEN
SYSTEM
ELSEIF c% = 1 OR ch% = ASC("t") THEN
chain$ = "on"
VIEW PRINT 27 TO 28
LOCATE 27: COLOR 7: PRINT "Please wait..."
CHAIN "wvtutor.drv"
VIEW PRINT 3 TO 28
ELSEIF c% = 2 OR ch% = ASC("w") THEN
dror% = 3: xoff% = 140: xlen% = 360: yoff1% = 150: yoff2% = 350
```

```
chain$ = "on"
VIEW PRINT 27 TO 28
LOCATE 27: COLOR 7: PRINT "Please wait..."
CHAIN "wavedro.bas"
VIEW PRINT 3 TO 28
ELSEIF c% = 3 OR ch% = ASC("f") THEN
chain$ = "on"
VIEW PRINT 27 TO 28
LOCATE 27: COLOR 7: PRINT "Please wait..."
CHAIN "wvfuria.drv"
VIEW PRINT 3 TO 28
ELSE
CLS : BEEP
COLOR 15
LOCATE 12, 20: PRINT "Please , choose one of the options."
FOR i = 1 TO 8000: NEXT i
END IF
LOOP
SYSTEM
```


APPENDIX C

PROGRAM LIST FOR TUTORIAL SECTION

```
REM /* PROGRAM FOR TUTORIAL SECTION OF WORLD OF WAVES */
REM /* FILENAME IS "WVTUTOR.DRV" */
DECLARE SUB loader1 ()
DECLARE SUB lmotion ()
DECLARE SUB cmotion ()
DECLARE SUB pendulum ()
DECLARE SUB vmotion ()
DECLARE SUB dro1 ()
DECLARE SUB dro2 ()
DECLARE SUB dro3 ()
DECLARE SUB loader2 ()
DECLARE SUB Xaxis2 ()
DECLARE SUB Xaxis3 ()
DECLARE SUB Xaxis1 ()

COMMON SHARED chain$
COMMON SHARED ampl1, ampl2, col1%, freq1%, xlen%, phase1%, erase%, freq2%
COMMON SHARED wtype1%, yoff1%, xoff%, amp%, yoff2%
COMMON SHARED prct3$, yoff3%, damp%, condition$, vtype%

DIM pix1%(25000)

DIM ydummy1(721), ydummy2(721), y1(721), y2(721), y(721)
DIM reso1(1), reso2(1), reso3(1), period1(1), period2(1)
SCREEN 12
wtype1% = 1: wtype2% = 1: CALL loader1: CALL loader2
ampl1% = 30: ampl2% = 30: xoff% = 0: xlen% = 360: col1% = 4: col2% = 2
col3% = 3

VIEW PRINT 3 TO 28
title:
OPEN "wvtutor.txt" FOR INPUT AS #1
CLS
COLOR 4: LOCATE 6
FOR i% = 1 TO 5
    LINE INPUT #1, text$
    PRINT text$
NEXT i%

COLOR 3: LOCATE 15:
FOR i% = 1 TO 7
    LINE INPUT #1, text$
    PRINT text$
NEXT i%
SLEEP 3
page1:
page$ = "page1"
GOSUB locatepage
CLS
COLOR 5: LOCATE 3, 33: GOSUB inputtext
LOCATE 4, 33: GOSUB inputtext
COLOR 3: LOCATE 6: GOSUB inputtext
COLOR 7
LOCATE 8: GOSUB inputtext
    GOSUB continuity
    IF preview% THEN
        CLOSE #1: GOTO title
    END IF

page2:
page$ = "page2"
GOSUB locatepage
```

```
CLS
COLOR 3: GOSUB inputtext
COLOR 7
GOSUB inputtext
LOCATE 14: GOSUB inputtext
CALL lmotion
VIEW PRINT 19 TO 28
LOCATE 19: GOSUB inputtext
VIEW PRINT 3 TO 28
GOSUB continuity
IF preview% THEN GOTO page1
page3:
page$ = "page3"
GOSUB locatepage
```

```
CLS
COLOR 3
VIEW PRINT
GOSUB inputtext
COLOR 7: GOSUB inputtext
CALL cmotion
LOCATE 28: GOSUB inputtext
PRINT
GOSUB continuity
IF preview% THEN GOTO page2
page4:
page$ = "page4"
GOSUB locatepage
```

```
CLS
COLOR 3: GOSUB inputtext
COLOR 7: GOSUB inputtext
damp% = 0: CALL pendulum
VIEW PRINT 3 TO 13
SLEEP 2
GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page3
page5:
page$ = "page5"
GOSUB locatepage
```

```
CLS : VIEW PRINT 3 TO 28
GOSUB inputtext
damp% = 1: CALL pendulum
GOSUB continuity
IF preview% THEN GOTO page4
page6:
page$ = "page6"
GOSUB locatepage
```

```
CLS : LOCATE 12: GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page5
page7:
page$ = "page7"
GOSUB locatepage
```

```
CLS
COLOR 3: GOSUB inputtext
COLOR 7: GOSUB inputtext
nmol% = 10: vib% = 0
FOR yoff3% = 200 TO 400 STEP 20
CALL vmotion
NEXT yoff3%
GOSUB continuity
```

```
IF preview% THEN GOTO page6
page8:
page$ = "page8"
GOSUB locatepage
  CLS : GOSUB inputtext
  vib% = 1: nmol% = 1: yoff3% = 300
  CALL vmotion
  GOSUB continuity
  IF preview% THEN GOTO page7
page9:
page$ = "page9"
GOSUB locatepage

  CLS : LOCATE 3: GOSUB inputtext
  GOSUB continuity
  IF preview% THEN GOTO page8
page10:
page$ = "page10"
GOSUB locatepage

  CLS : LOCATE 3: GOSUB inputtext
  nmol% = 10: vib% = 1
  yoff3% = 350
  CALL vmotion
  GOSUB continuity
  IF preview% THEN GOTO page9
page11:
page$ = "page11"
GOSUB locatepage
  CLS : LOCATE 3: GOSUB inputtext
  GOSUB continuity
  IF preview% THEN GOTO page10
page12:
page$ = "page12"
GOSUB locatepage
  CLS : LOCATE 12: GOSUB inputtext
  GOSUB continuity
  IF preview% THEN GOTO page11
page13:
page$ = "page13"
GOSUB locatepage

  CLS : LOCATE : GOSUB inputtext
  PRINT : GOSUB inputtext
  vib% = 1: nmol% = 1: yoff3% = 300
  CALL vmotion
  GOSUB continuity
  IF preview% THEN GOTO page12
page14:
page$ = "page14"          ***** CHAPTER 2 *****
GOSUB locatepage
CLS
COLOR 5: LOCATE 3, 33: GOSUB inputtext
LOCATE 4, 33: GOSUB inputtext
COLOR 7: LOCATE 4, 33: GOSUB inputtext
GOSUB inputtext
  GOSUB inputtext
xoff% = 20: yoff1% = 350: freq1% = 1: erase% = 0:
phase1% = 0: ampl1 = 40: xlen% = 360
  CALL dro1
  GOSUB continuity
  IF preview% THEN GOTO page13
page15:
page$ = "page15"
GOSUB locatepage
CLS
```

```
GOSUB inputtext
xoff% = 20: yoff1% = 350: freq1% = 1: eraze% = 0:
phase1% = 0: ampl1 = 10: xlen% = 360
CALL dro1
  LOCATE 8: GOSUB inputtext
  WHILE ampl1 < 40
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  IF LCASE$(a$) = "a" THEN
  ampl1 = ampl1 + 10: CALL dro1
  ELSE
  BEEP
  END IF
WEND
GET (20, 310)-(380, 390), pix1%
SLEEP 2
GOSUB continuity
IF preview% THEN GOTO page14
page16:
page$ = "page16"
GOSUB locatepage
CLS
PUT (150, 310), pix1%, OR
LOCATE 3: GOSUB inputtext
LOCATE 10: COLOR col1%: GOSUB inputtext
COLOR 7: GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page15
page17:
page$ = "page17"
GOSUB locatepage
CLS
pi = 22 / 7
xlen% = 270: xoff% = 0: freq1% = 1: yoff1% = 250: ampl1 = 30
CALL dro1
GOSUB inputtext
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1)
xoff% = 270
CALL dro1
LOCATE 20: GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page15
page18:
page$ = "page18"
GOSUB locatepage
CLS : GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page17
page19:
page$ = "page19"
GOSUB locatepage
CLS
GOSUB inputtext
xoff% = 0: yoff1% = 150: freq1% = 1: eraze% = 0:
phase1% = 0: ampl1 = 30: xlen% = 540
CALL dro1
GET (0, 120)-(540, 180), pix1%
LOCATE 16: GOSUB inputtext
COLOR col1%: GOSUB inputtext
COLOR 7: GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page18
page20:
page$ = "page20"
GOSUB locatepage
```

```
CLS
PUT (0, 120), pix1%, XOR
xoff% = 0: yoff1% = 150: freq1% = 1: eraze% = 0:
phase1% = 0: ampl1 = 30: xlen% = 540
yoff2% = 150: ampl2 = 20: freq2% = 2: phase2% = 0
CALL dro2
GET (0, 120)-(540, 180), pix1%
SLEEP 2
GOSUB inputtext
DO
i$ = INPUT$(1)
LOCATE 5, 54: PRINT i$: SLEEP 2
IF i$ = "2" THEN
EXIT DO
ELSEIF i$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
LOCATE 6: PRINT "Wrong. Try again."
END IF
LOOP

LOCATE 16: GOSUB inputtext
PRINT
COLOR 10: GOSUB inputtext
COLOR 7: GOSUB inputtext
GOSUB continuity
IF preview% THEN GOTO page19
page21:
page$ = "page21"
GOSUB locatepage
CLS
COLOR 10: GOSUB inputtext
yoff1% = 250: freq1% = 2: ampl1 = -30: xlen% = 540: xoff% = 0
CALL dro1
yoff2% = 350: freq2% = 4: ampl2 = 30
CALL dro2
GET (0, 220)-(540, 380), pix1%
LINE (34, 229)-(304, 229), 7
LINE (17, 371)-(152, 371), 7
COLOR 7
PRINT : GOSUB inputtext
DO
LOCATE 13, 54:
INPUT "Answer = "; n$
SLEEP 2
IF UCASE$(n$) = "GREEN" THEN
EXIT DO
ELSEIF a$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
LOCATE 10: PRINT "Wrong, try again."
END IF
LOOP

SLEEP 2
CLS
GOSUB inputtext
PUT (0, 200), pix1%, XOR
LINE (67, 200)-(337, 200), 7
LINE (34, 360)-(165, 360), 7
GOSUB continuity
IF preview% THEN GOTO page19
page22:
page$ = "page22"
```

```
GOSUB locatepage
  CLS
  yoff2% = 350: freq1% = 2: yoff1% = 150: xoff% = 0: xlen% = 540: ampl1 = -30
  CALL dro1
  freq2% = 4: CALL dro2
  GOSUB inputtext
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  yoff3% = 250:
  CALL dro3
  GOSUB continuity
  IF preview% THEN GOTO page21
page23:
page$ = "page23"
CLS
LOCATE 4: GOSUB inputtext
  GOSUB continuity
  IF preview% THEN GOTO page22
page24:
page$ = "page24"
GOSUB locatepage
  CLS
  GOSUB inputtext
  ampl1 = 30: yoff1% = 150: yoff2% = 350: freq1% = 1: freq2% = 1:
  ampl2 = -30: yoff3% = 250
  CALL dro1
  CALL dro2
  GOSUB inputtext
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  CALL dro3
  GOSUB continuity
  IF preview% THEN GOTO page23
page25:
page$ = "page25"
GOSUB locatepage
  CLS
  GOSUB inputtext
  LOCATE 11: COLOR col1%: GOSUB inputtext
  COLOR 7
  GOSUB continuity
  IF preview% THEN GOTO page24
page26:
page$ = "page26"
GOSUB locatepage
  CLS
  GOSUB inputtext
  ampl2 = -30: yoff1% = 150: yoff2% = 350: yoff3% = 250: ampl1 = -30
  freq1% = 1: freq2% = 1
  IF wtype2% = 2 THEN wtype2% = 1: CALL loader2
  CALL dro1: CALL dro2: CALL dro3
  GOSUB continuity
  IF preview% THEN GOTO page25
page27:
page$ = "page27"
GOSUB locatepage
  CLS
  LOCATE 3: : GOSUB inputtext
  COLOR 9: LOCATE 11: GOSUB inputtext
  GOSUB continuity
  IF preview% THEN GOTO page26
page28:
page$ = "page28"
GOSUB locatepage
  CLS
  LOCATE 3: GOSUB inputtext
```

```
wtype2% = 2: yoff2% = 250: yoff1% = 250: xoff% = 0:
ampl1 = -30: ampl2 = -30
CALL loader2
CALL dro1
CALL dro2
GOSUB continuity
IF preview% THEN GOTO page27
page29:
page$ = "page29"
GOSUB locatepage
CLS
GOSUB inputtext
LOCATE 10: GOSUB inputtext
LOCATE 18, 24: COLOR 10: GOSUB inputtext
COLOR 7
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1)
IF LCASE$(a$) = "b" THEN
    GOTO page28
ELSEIF a$ = CHR$(27) THEN
    CLOSE #1:
IF chain$ <> "on" THEN
    SYSTEM
ELSE
    VIEW PRINT 27 TO 28
    PRINT "Returning to main module..."
    VIEW PRINT 3 TO 28
    CHAIN "waves.drv"
END IF
ELSE
    prct$ = "yes"
    chain$ = "on"
    VIEW PRINT 27 TO 28
    PRINT " Please wait..."
    VIEW PRINT 3 TO 28
    CHAIN "wavedro.bas"
END IF
IF chain$ <> "on" THEN SYSTEM
***** sub routines *****
inputtext:
DO
    IF EOF(1) THEN
        CLOSE #1
        OPEN "tutor.txt" FOR INPUT AS #1
    END IF
    text$ = ""
    LINE INPUT #1, text$
    IF LEFT$(text$, 4) <> "page" THEN
        PRINT text$
    END IF
LOOP UNTIL text$ = ""
RETURN

locatepage:
kount% = 0
DO
    IF EOF(1) THEN
        kount% = kount% + 1
        CLOSE #1
        OPEN "wvtutor.txt" FOR INPUT AS #1
        IF kount% = 3 THEN RETURN
    END IF
    text$ = ""
    LINE INPUT #1, text$
LOOP UNTIL text$ = page$
RETURN
```

```
continuity:
REM ***** next screen display prompt *****
COLOR 1
VIEW PRINT 29 TO 30
PRINT "PRESS:- b -(previous page)  esc -(quit)  any other key -(next page)"
VIEW PRINT 3 TO 28
COLOR 7
DO WHILE NOT INKEY$ = "": LOOP 'EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1)
preview% = 0
IF a$ = CHR$(27) THEN
  CLOSE #1
  IF chain$ <> "on" THEN
    SYSTEM
  ELSE
    VIEW PRINT 29 TO 30
    PRINT : PRINT
    PRINT "Returning to main module..."
    VIEW PRINT 3 TO 28
    CHAIN "waves.drv"
  END IF
ELSEIF a$ = "b" OR a$ = "B" THEN
  preview% = 1
END IF
RETURN

SUB cmotion
'CIRCULAR MOTION
DIM x(361), y(361)
DIM pix1%(2000)

CIRCLE (420, 250), 5, 7
GET (415, 245)-(425, 255), pix1%
r = 140 'r = radius of circle (path)

CIRCLE (280, 250), 140, 4
pi = 22 / 7
FOR i% = 0 TO 360
  x(i%) = 275 + (r * COS(i% * (pi / 180)))
  y(i%) = 245 + (r * SIN(i% * (pi / 180)))
NEXT i%
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1)
kount% = 0
DO
  FOR i% = 1 TO 361
    PUT (x(i%), y(i%)), pix1%, XOR
    IF i% <> 361 THEN
      PUT (x(i% - 1), y(i% - 1)), pix1%, XOR
    END IF
  NEXT i%
  kount% = kount% + 1
LOOP UNTIL kount% = 3
END SUB

SUB dro1 'For drawing wave 1
SHARED ampl1, col1%, freq1%, xlen%, phase1%, eraze%
SHARED wtype1%, xoff%, damp1%, yoff1%, ydummy1(), y1()
SHARED stepp1, limit1%, dror%, yoff3%, reso1(), period1()

IF phase1% < 0 THEN
  faze% = 720 - ABS(2 * phase1%)
ELSE
  faze% = 2 * phase1%
END IF
```



```
k% = 0
i% = faze%
***** store values to memory .Necessary for eraser to work.*****
DO
    z = ydummy1(i%) * ampl1
    y1(k%) = z + yoff1%
    k% = k% + 1
    i% = i% + 1
    IF i% > 720 THEN i% = 1 'Because of phase considerations
LOOP UNTIL k% = 721
***** To set resolution of screen. Necessary for optimum speed.***
q = (ampl1 * SIN(70)) / (70 * freq1%)
IF q < .2 THEN q = .2
reso1(1) = 1 / q
*****

nwaves% = freq1%
period1(1) = xlen% / freq1%
xdummy = period1(1) / 720

CALL Xaxis1    *** Draw equilibrium line
FOR p% = 1 TO nwaves%
    nextcycle = (p% - 1) * period1(1)
    FOR i% = 0 TO 720 STEP reso1(1)
        x1 = i% * xdummy + nextcycle + xoff%
        IF y1(i%) > 80 AND y1(i%) < 400 THEN 'set graph print-area of screen]
            PSET (x1, y1(i%)), col1%
        END IF
    NEXT i%
NEXT p%
END SUB

SUB dro2
SHARED ampl2, offset2%, col2%, freq2%, xlen%, phase2%, erase%
SHARED reso2, wtype2%, ydummy2(), y2(), yoff2%, xoff%, yoff3%
SHARED stepp2, limit2%, dror%, reso2(), period2()

IF phase2% < 0 THEN
    faze% = 720 - ABS(2 * phase2%)
ELSE
    faze% = 2 * phase2%
END IF

q = (ampl2 * SIN(70)) / (70 * freq2%)
IF q < .2 THEN q = .2
reso2(1) = 1 / q

k% = 0
i% = faze%
DO
    z = ydummy2(i%) * ampl2
    y2(k%) = z + yoff2%
    k% = k% + 1
    i% = i% + 1
    IF i% > 720 THEN i% = 1
LOOP UNTIL k% = 721
rate = 0
nwaves% = freq2%
period2(1) = xlen% / freq2%
xdummy = period2(1) / 720

CALL Xaxis2
FOR p% = 1 TO nwaves%
    nextcycle = (p% - 1) * period2(1)

    FOR i% = 0 TO 720 STEP reso2(1)
        x2 = i% * xdummy + nextcycle + xoff%
```

```
IF y2(i%) > 80 AND y2(i%) < 400 THEN
    PSET (x2, y2(i%)), col2%
END IF
NEXT i%
NEXT p%
END SUB

SUB dro3 'To add wave 1 and wave 2
    SHARED ydummy1(), ydummy2(), y(), yoff3%, xoff%, erase%, reso3()
    SHARED ampl1, ampl2, col3%, xlen%, reso%, reso2, freq1%, freq2%, y1(), y2()
```

```
m = 0: n = 0
xdummy = xlen% / 720
CALL Xaxis3
reso3(1) = 1
FOR i = 0 TO 720 STEP reso3(1)
    x = xdummy * i + xoff%
    j = m * freq1%
    IF j > 720 THEN m = 1: j = 0
        h = n * freq2%
        IF h > 720 THEN n = 1: h = 0
            y(i) = y1(j) + y2(h)
            y = y(i) - yoff3%
            IF y > 80 AND y < 400 THEN
                PSET (x, y), col3%
            END IF
        m = m + reso3(1): n = n + reso3(1)
    NEXT i
END SUB
```

```
SUB lmotion
DIM pix1%(20000)
'linear motion
LINE (30, 120)-(130, 192), 4, B
GET (30, 120)-(130, 192), pix1%
LINE (20, 194)-(600, 194), 5
DO WHILE INKEY$ = "": LOOP
    stepp% = 2
    FOR i% = 1 TO 420 'STEP stepp%
        LINE (28 + i%, 119)-(131 + i%, 193), 0, B
        PUT (30 + i%, 120), pix1%, PSET
        'IF i% <> 0 THEN
        'PUT (30 + i% - stepp%, 120), pix1%, XOR
        'END IF
    NEXT i%
END SUB
```

```
SUB loader1 'Retrieve values from relevant
SHARED wtype1%, ydummy1() 'tables (on file ) and store in memory
```

```
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7
SELECT CASE wtype1%
CASE 1
    OPEN "sine.tbl" FOR INPUT AS #2
CASE 2
    OPEN "cosine.tbl" FOR INPUT AS #2
CASE 3
    OPEN "sawtooth.tbl" FOR INPUT AS #2
CASE 4
    OPEN "square.tbl" FOR INPUT AS #2
CASE 5
    OPEN "triangle.tbl" FOR INPUT AS #2
END SELECT
```

```
FOR i% = 0 TO 720
  INPUT #2, ydummy1(i%)
NEXT i%
CLOSE #2
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
SUB loader2 'see loader one for relevant comments
SHARED wtype1%, ydummy1(), dror%, ydummy2(), wtype2%
```

```
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7
```

```
SELECT CASE wtype2%
CASE 1
  OPEN "sine.tbl" FOR INPUT AS #3
CASE 2
  OPEN "cosine.tbl" FOR INPUT AS #3
CASE 3
  OPEN "sawtooth.tbl" FOR INPUT AS #3
CASE 4
  OPEN "square.tbl" FOR INPUT AS #3
CASE 5
  OPEN "triangle.tbl" FOR INPUT AS #3
END SELECT
FOR i% = 0 TO 720
  INPUT #3, ydummy2(i%)
NEXT i%
CLOSE #3
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
SUB pendulum
DIM pix1%(20000)
DIM x(120), x1(120), q1(120), y(120)
DIM e(121), e1(121), e2(121), e3(121)
SCREEN 12
r = 160: pi = 22 / 7: r1 = r
i% = 120
FOR i% = 120 TO 60 STEP -1
  x(i%) = 273 + (r1 * COS(i% * (pi / 180)))
  y(i%) = 250 + r * SIN(i% * (pi / 180))
  x1(i%) = (r * COS(i% * (pi / 180)))
  q1(i%) = r * SIN(i% * (pi / 180))

  e(i%) = (r * COS((i% + 1) * (pi / 180)))
  e1(i%) = r * SIN((i% + 1) * (pi / 180))
  e2(i%) = (r * COS((i% - 1) * (pi / 180)))
  e3(i%) = r * SIN((i% - 1) * (pi / 180))
NEXT i%
```

```
CIRCLE (280, 421), 10, 4 'DRAW PENDULUM BOB
PAINT (280, 421), 4
GET (270, 411)-(290, 431), pix1% 'STORE IMAGE OF BOB
```

```
LINE (260, 250)-(300, 250), 1 'horizontal support
LINE (280, 251)-(280, 411), 7
LINE (281, 251)-(281, 411), 7 ' DRAW PENDULUM
LINE (282, 251)-(282, 411), 7
```

```
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1)
```

```
LINE (280, 251)-(283 + x1(120), 251 + q1(120)), 7
LINE (281, 251)-(284 + x1(120), 251 + q1(120)), 7 'DRAW PENDULUM
LINE (282, 251)-(285 + x1(120), 251 + q1(120)), 7 'AT NEW LOCATION
PUT (270 + x1(120), 251 + q1(120)), pix1%, PSET
```

```
LINE (280, 251)-(280, 411), 0 '
LINE (281, 251)-(281, 411), 0 ' ERASE AT FORMER LOCATION
LINE (282, 251)-(282, 411), 0 '
PAINT (280, 421), 0 '
```

```
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
```

```
a$ = INPUT$(1)
```

```
j% = 120: k% = 60: kount% = 0: n% = -1: m% = 1: d% = 0: prerub% = 0
```

```
DO
```

```
FOR i% = j% TO k% STEP n%
```

```
IF prerub% THEN
```

```
LINE (280, 251)-(280 + e(i% + 3), 251 + e1(i% + 3)), 0
```

```
LINE (281, 251)-(281 + e(i% + 3), 251 + e1(i% + 3)), 0 'ERASE FORMER
```

```
LINE (282, 251)-(282 + e(i% + 3), 251 + e1(i% + 3)), 0 ' POSITION
```

```
PUT (x(i% + 4), y(i% + 4)), pix1%, PSET
```

```
prerub% = 0
```

```
END IF
```

```
LINE (280, 251)-(280 + x1(i%), 251 + q1(i%)), 7
```

```
LINE (281, 251)-(281 + x1(i%), 251 + q1(i%)), 7 'MOVE PENDULUM
```

```
LINE (282, 251)-(282 + x1(i%), 251 + q1(i%)), 7 'TO THE RIGHT
```

```
PUT (x(i%), y(i%)), pix1%, PSET
```

```
LINE (280, 251)-(280 + e(i%), 251 + e1(i%)), 0
```

```
LINE (281, 251)-(281 + e(i%), 251 + e1(i%)), 0 'ERASE FORMER POSITION
```

```
LINE (282, 251)-(282 + e(i%), 251 + e1(i%)), 0
```

```
NEXT i%
```

```
FOR i% = k% TO j% STEP m%
```

```
LINE (280, 251)-(280 + x1(i%), 251 + q1(i%)), 7
```

```
LINE (281, 251)-(281 + x1(i%), 251 + q1(i%)), 7 'MOVE PENDULUM TO THE LEFT
```

```
LINE (282, 251)-(282 + x1(i%), 251 + q1(i%)), 7
```

```
PUT (x(i%), y(i%)), pix1%, PSET
```

```
LINE (280, 251)-(280 + e2(i%), 251 + e3(i%)), 0
```

```
LINE (281, 251)-(281 + e2(i%), 251 + e3(i%)), 0 'ERASE
```

```
LINE (282, 251)-(282 + e2(i%), 251 + e3(i%)), 0
```

```
NEXT i%
```

```
kount% = kount% + 1
```

```
LOCATE 24, 65: PRINT "count =": kount%
```

```
IF damp% THEN
```

```
j% = j% - 4: k% = k% + 4: prerub% = 1
```

```
END IF
```

```
LOOP UNTIL INKEY$ = CHR$(27) OR kount% = 10
```

```
END SUB
```

```
SUB vmotion
```

```
SHARED xlen%, nmol%, xoff%, yoff3%, col3%, vib%
```

```
col3% = 3: xlen% = 360: xoff% = 140
```

```
DIM pix2%(25000)
```

```
DIM x(nmol% + 1), ch%(nmol% + 1), vstop%(nmol%), Xnot(nmol%)
```

```
SCREEN 12
```

```
rad = xlen% / (4 * nmol%)
```

```
IF rad > 25 THEN rad = 25
```

```
CIRCLE (xoff%, yoff3%), rad, col3%
```

```
xup = xoff% - rad: yup = yoff3% - rad
```

```
xdan = xoff% + rad: ydan = yoff3% + rad
```

```
GET (xup, yup)-(xdan, ydan), pix2%
```

```
Xnot(1) = xoff% - rad
```

```
y = yup
```

```
FOR i% = 2 TO nmol%
```

```
  Xnot(i%) = (xlen% / nmol%) * (i% - 1) + xoff% - rad
```

```
  PUT (Xnot(i%), y), pix2%
```

```
NEXT i%
```

```
IF vib% THEN
```

```
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
```

```
a$ = INPUT$(1)
```

```
apltd = 3 * rad: rate% = 2
```

```
ch%(1) = rate%:
```

```
x(1) = Xnot(1)
```

```
FOR i% = 2 TO nmol%
```

```
  ch%(i%) = 0: x(i%) = Xnot(i%)
```

```
  vstop%(i%) = 0
```

```
NEXT i%
```

```
kount% = 0
```

```
DO
```

```
  ch%(1) = rate%: x(1) = Xnot(1): x(nmol%) = Xnot(nmol%)
```

```
DO WHILE NOT x(nmol%) < Xnot(nmol%)
```

```
  FOR i% = 1 TO nmol%
```

```
    x(i%) = ch%(i%) + x(i%)
```

```
    diff% = x(i%) - Xnot(i%)
```

```
IF ch%(i%) <> 0 THEN
```

```
  IF x(i%) = x(i% + 1) - 2 * rad THEN
```

```
    ch%(i%) = -rate%
```

```
    ch%(i% + 1) = rate%
```

```
    PUT (x(i%), y), pix2%, PSET
```

```
  ELSEIF x(i%) < Xnot(i%) THEN ch%(i%) = 0
```

```
  ELSEIF diff% > apltd THEN
```

```
    ch%(i%) = -rate%
```

```
    PUT (x(i%), y), pix2%, PSET
```

```
  ELSE
```

```
    PUT (x(i%), y), pix2%
```

```
    IF ch%(i%) = rate% THEN
```

```
      PUT (x(i%) - rate%, y), pix2%
```

```
    ELSE
```

```
      PUT (x(i%) + rate%, y), pix2%
```

```
    END IF
```

```
  END IF
```

```
END IF
```

```
NEXT i%
```

```
LOOP
```

```
kount% = kount% + 1
```

```
LOOP UNTIL INKEY$ <> "" OR kount% = 6
```

```
END IF
```

```
END SUB
```

```
SUB Xaxis1 'To draw equilibrium line for wave 1
```

```
SHARED yoff1%, xlen%, xoff%
```

```
FOR i% = 0 TO xlen% STEP 6
```

```
  PSET (i% + xoff%, yoff1%), 1
```

```
NEXT i%
```

```
END SUB
```

```
SUB Xaxis2 'To draw equilibrium line for wave 2
```

```
SHARED yoff2%, xlen%, xoff%
```

```
FOR i% = 0 TO xlen% STEP 6
```

```
  PSET (i% + xoff%, yoff2%), 1
```

```
NEXT i%
```

```
END SUB
```

```
SUB Xaxis3 'To draw equilibrium line for wave 3
```

```
SHARED xlen%, yoff3%, xoff%  
FOR i% = 0 TO xlen% STEP 6  
PSET (i% + xoff%, yoff3%), 1  
NEXT i%  
END SUB
```

APPENDIX D

PROGRAM LIST FOR WAVE-DRAW SECTION

```
REM /* PROGRAM FOR WAVE-DRAW SECTION OF THE WORLD OF WAVES */
REM /* PROGRAM FILE NAME IS "WAVEDRO.BAS" */
DECLARE SUB ruboff ()
DECLARE SUB merge ()
DECLARE SUB amplitude ()
DECLARE SUB frequency ()
DECLARE SUB phase ()
DECLARE SUB loader1 ()
DECLARE SUB loader2 ()
DECLARE SUB wavetype ()
DECLARE SUB graphpage ()
DECLARE SUB library ()
DECLARE SUB clip ()
DECLARE SUB decay ()
DECLARE SUB fkeydisplay ()
DECLARE SUB eraser1 ()
DECLARE SUB eraser2 ()
DECLARE SUB eraser3 ()
DECLARE SUB klean ()
DECLARE SUB formula ()
DECLARE SUB Xaxis1 ()
DECLARE SUB Xaxis2 ()
DECLARE SUB Xaxis3 ()
DECLARE SUB dro1 ()
DECLARE SUB dro2 ()
DECLARE SUB dro3 ()

COMMON SHARED chain$

DIM SHARED period1(1), period2(1)
DIM SHARED ydummy1(721), ydummy2(721), reso1(1), reso2(1), reso3(1)
DIM SHARED y1(721), y2(721), y(721)

SCREEN 12 ***** configure for graphics image *****
***** Initialize variables *****
freq1% = 9: freq2% = 9: yoff1% = 150: yoff2% = 350: wtype1% = 1: ampl1 = 10
ampl2 = 20: dror% = 3: col1% = 4: col2% = 2: wtype2% = 1
col3% = 3: a = 1.5: yoff3% = 250: eraze% = 0: nmol% = 3
xlen% = 360: phase1% = 0: phase2% = 0: xoff% = 140
stat% = xoff% - 5: stp% = xoff% + xlen% + 5
dror% = 3: xoff% = 140: xlen% = 360: yoff1% = 150: yoff2% = 350

CALL loader1: CALL loader2

CLS
CALL fkeydisplay
CALL graphpage
eraze% = 0: 'to set erasers on , after first draw.
ch1% = 1: ch2% = 1
DO
CALL formula
***** 'To specify the no. of waves to draw
IF ch1% = 0 AND ch2% = 0 THEN
dror% = 0
VIEW PRINT 27 TO 28
COLOR 14
LOCATE 27, 30: PRINT "No changes made."
COLOR 7
SLEEP 1
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
ELSEIF ch1% + ch2% = 2 THEN
```

```
        dror% = 3
ELSEIF ch1% = 1 AND ch2% = 0 AND dror% <> 1 THEN
    dror% = 4
ELSEIF ch2% = 1 AND ch1% = 0 AND dror% <> 2 THEN
    dror% = 5
END IF

SELECT CASE dror%
CASE 1
    CALL dro1
CASE 2
    CALL dro2
CASE 3
    CALL dro1
    CALL dro2
    CALL dro3
CASE 4
    CALL dro1
    CALL dro3
CASE 5
    CALL dro2
    CALL dro3
CASE ELSE
    BEEP
END SELECT
    eraze% = 1
    DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
    a$ = INPUT$(1)
    IF a$ = CHR$(27) THEN EXIT DO
    GOSUB task
LOOP
VIEW PRINT 3 TO 28
IF chain$ <> "on" THEN
    SYSTEM
ELSE
    VIEW PRINT 27 TO 28
    COLOR 15: PRINT "Returning to main module..."
    VIEW PRINT 3 TO 28: COLOR 7
    CHAIN "waves.drv"
END IF

***** sub-routines *****
.
.
.
task:
ch1% = 0: ch2% = 0
DO
    task% = ASC(LCASE$(a$))
SELECT CASE task%
CASE ASC("w")
    CALL wavetype      'Select a type of wave
CASE ASC("a")
    CALL amplitude     'Vary amplitude
CASE ASC("f")
    CALL frequency     'Vary frequency
CASE ASC("p")
    CALL phase         'Vary phase
CASE ASC("l")
    CALL library       'Select wave effect
CASE ASC("o")
    chain$ = "on"
    VIEW PRINT 27 TO 28
    LOCATE 27: COLOR 7: PRINT "Please wait..."
    CHAIN "wvfuria.drv" 'go to Fourier wave analysis and drawing
    VIEW PRINT 3 TO 28
CASE ASC("y")
    CALL decay         'Produce damped oscillation
```



```
CASE ASC("c")
  CALL clip      'Produce clipped oscillation
CASE ASC("r")
  CALL ruboff
CASE ASC("m")
  CALL merge
CASE ELSE
  RETURN
END SELECT
VIEW PRINT 3 TO 6
  PRINT :PRINT :PRINT :PRINT
  COLOR 14
  LOCATE 3, 25
  PRINT SPC(5); "press 'enter' if no more changes."
  COLOR 7
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  PRINT :PRINT
  PRINT :PRINT
  VIEW PRINT 3 TO 28
  IF a$ = CHR$(13) THEN EXIT DO
LOOP
RETURN

SUB amplitude    'To change the amplitude of the waves
SHARED col1%, col2%, amp12, ch1%, ch2%, amp1

VIEW PRINT 3 TO 6
LOCATE 3, 10: COLOR col1%: PRINT "Amplitude of 1 = "; amp1
LOCATE 3, 40: COLOR col2%: PRINT "Amplitude of 2 = "; amp2
COLOR 7
LOCATE 4: PRINT "Change 1 or 2 ? (select)";
DO
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1): LOCATE 4, 25: PRINT a$; SPC(5);
a% = VAL(a$)
IF a% = 1 THEN
  PRINT "press d-decrease i-increase esc-quit"
  amp% = amp1    'Initialize variable for subroutine
  DO
    DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
    B$ = INPUT$(1)
    GOSUB amp: amp1 = amp%
    LOCATE 3, 27: PRINT amp1
    LOOP UNTIL B$ = CHR$(27)
    PRINT :PRINT :PRINT
    ch1% = 1 'flag to show that a change has taken place
  EXIT SUB
ELSEIF a% = 2 THEN
  PRINT "press d-decrease i-increase esc-quit"
  amp% = amp2
  DO
    DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
    B$ = INPUT$(1)
    GOSUB amp: amp2 = amp% 'return new value of amplitude
    LOCATE 3, 56: PRINT amp2
    LOOP UNTIL B$ = CHR$(27)
    PRINT :PRINT :PRINT
    ch2% = 1
  EXIT SUB
ELSEIF a$ = CHR$(13) THEN
  EXIT SUB
ELSE
  BEEP
END IF
LOOP
```

```
amp:
IF LCASE$(B$) = CHR$(ASC("i")) THEN
  amp% = amp% + 1
  IF amp% > 160 THEN BEEP: amp% = 160
ELSEIF LCASE$(B$) = CHR$(ASC("d")) THEN
  amp% = amp% - 1
  IF amp% < -160 THEN BEEP: amp% = -160
END IF
RETURN
END SUB
```

```
      SUB clip  'To set maximum limit of amplitude
SHARED limit1%, limit2%, ch1%, ch2%
```

```
VIEW PRINT 3 TO 6
LOCATE 3, 5:
PRINT "Clip wave 1 or wave 2 ? (select number)"
DO
  DO WHILE NOT INKEY$ = "": LOOP  'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  LOCATE 3, 50: PRINT a$
  a% = VAL(a$)
  IF a% = 1 THEN
    ch1% = 1
    LOCATE 4, 5: INPUT "Enter maximum amplitude desired.": limit1%
    EXIT DO
  ELSEIF a% = 2 THEN
    ch2% = 1
    LOCATE 4, 5: INPUT "Enter maximum amplitude desired.": limit2%
    EXIT DO
  ELSEIF a$ = CHR$(27) THEN
    EXIT DO
  ELSE
    BEEP
  END IF
LOOP
PRINT : PRINT : PRINT
END SUB
```

```
      SUB decay  'To produce damped oscillation
SHARED ampl1, ampl2, stepp1, stepp2, freq1%, freq2%, yoff1%, ch1%, ch2%
SHARED yoff2%
CALL klean
VIEW PRINT 3 TO 6
LOCATE 3, 5:
PRINT "Dampen wave 1 or wave 2 ? (select number)"
DO
  DO WHILE NOT INKEY$ = "": LOOP  'TO EMPTY KEYBOARD INPUT BUFFER
  a$ = INPUT$(1)
  LOCATE 3, 50: PRINT a$
  a% = VAL(a$)
  IF a% = 1 THEN
    ch1% = 1
    ampl = ampl1
    frq% = freq1%  'Initialize subroutine variable
    GOSUB dec
    stepp1 = stePP
    EXIT SUB
  ELSEIF a% = 2 THEN
    ch2% = 1
    ampl = ampl2
    frq% = freq2%
    GOSUB dec
    stepp2 = stePP
    EXIT SUB
```

```
ELSEIF a$ = CHR$(27) THEN
  EXIT SUB
ELSE
  BEEP
END IF
LOOP
dec:
LOCATE 3
PRINT "Dampen :1.slowly 2.heavilly 3.critically 4.naturally 5.Reset(select no.)"
DO
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  B$ = INPUT$(1): LOCATE 3, 78: PRINT a$
  B% = VAL(B$)
  IF a$ = CHR$(27) THEN
    EXIT DO
  ELSEIF B% > 0 AND B% < 6 THEN
    SELECT CASE B%
      CASE 1
        stePP = ampl / (4 * frq% * 720)
      CASE 2
        stePP = ampl / (720)
      CASE 3
        stePP = ampl / (2 * frq% * 720)
      CASE 4
        rate = rate
      CASE 5
        stePP = 0
    END SELECT
  END IF
ELSE
  BEEP
END IF
LOOP
RETURN
END SUB
```

```
SUB dro1 'For drawing wave 1
SHARED ampl1, offset1%, col1%, frq1%, xlen%, phase1%, erase%
SHARED wtype1%, ydummy1(), y(), xoff%, damp1%, yoff1%
SHARED stepp1, limit1%, dror%, yoff3%
```

```
IF erase% THEN 'if condition to erase is yes
  CALL eraser1
END IF
```

```
IF phase1% < 0 THEN
  faze% = 720 - ABS(2 * phase1%)
ELSE
  faze% = 2 * phase1%
END IF
```

```
k% = 0
i% = faze%
***** store values to memory .Necessary for eraser to work.*****
DO
  z = ydummy1(i%) * ampl1
  ***** if clipping is desired *****
  IF limit1% > 0 AND limit1% < 71 THEN
    IF z > limit1% THEN
      z = limit1%
    ELSEIF z < -limit1% THEN
      z = -limit1%
    END IF
  END IF
END IF
*****
```

```
y1(k%) = z + yoff1%
k% = k% + 1
i% = i% + 1
IF i% > 720 THEN i% = 1 'Because of phase considerations
LOOP UNTIL k% = 721
***** To set resolution of screen. Necessary for optimum speed.***
q = (ampl1 * SIN(70)) / (70 * freq1%)
IF q < .2 THEN q = .2
reso1(1) = 1 / q
*****
nwaves% = freq1%
period1(1) = xlen% / freq1%
xdummy = period1(1) / 720

CALL Xaxis1 '** Draw equilibrium line
FOR p% = 1 TO nwaves%
nextcycle = (p% - 1) * period1(1)
FOR i% = 0 TO 720 STEP reso1(1)
x1 = i% * xdummy + nextcycle + xoff%
***** If damping is required *****
IF stepp1 > 0 THEN
IF dror% = 1 THEN yoff1% = yoff3%
IF y1(i%) <= yoff1% THEN
y1(i%) = y1(i%) + rate
IF y1(i%) >= yoff1% THEN y1(i%) = yoff1%
ELSEIF y1(i%) > yoff1% THEN
y1(i%) = y1(i%) - rate
IF y1(i%) < yoff1% THEN y1(i%) = yoff1%
END IF
END IF
*****
IF y1(i%) > 80 AND y1(i%) < 400 THEN '[set graph print-area of screen]
PSET (x1, y1(i%)), col1%
END IF
rate = rate + stepp1 * reso1(1)
NEXT i%
NEXT p%
END SUB

SUB dro2
SHARED ampl2, offset2%, col2%, freq2%, xlen%, phase2%, erase%
SHARED reso2, wtype2%, ydummy2(), y(), yoff2%, xoff%, yoff3%
SHARED stepp2, limit2%, dror%

IF erase% THEN
CALL eraser2
END IF

IF phase2% < 0 THEN
faze% = 720 - ABS(2 * phase2%)
ELSE
faze% = 2 * phase2%
END IF

q = (ampl2 * SIN(70)) / (70 * freq2%)
IF q < .2 THEN q = .2
reso2(1) = 1 / q

k% = 0
i% = faze%
DO
z = ydummy2(i%) * ampl2
IF limit2% > 0 AND limit2% < 20 THEN
IF z > limit2% THEN
z = limit2%
ELSEIF z < -limit2% THEN
```

```
        z = -limit2%
    END IF
END IF
y2(k%) = z + yoff2%
k% = k% + 1
i% = i% + 1
IF i% > 720 THEN i% = 1
LOOP UNTIL k% = 721
rate = 0
nwaves% = freq2%
period2(1) = xlen% / freq2%
xdummy = period2(1) / 720

CALL Xaxis2
FOR p% = 1 TO nwaves%
    nextcycle = (p% - 1) * period2(1)

    FOR i% = 0 TO 720 STEP reso2(1)
        x2 = i% * xdummy + nextcycle + xoff%

        IF stepp2 > 0 THEN
            IF dror% = 1 THEN yoff2% = yoff3%
            IF y2(i%) <= yoff2% THEN
                y2(i%) = y2(i%) + rate
                IF y2(i%) >= yoff2% THEN y2(i%) = yoff2%
            ELSEIF y2(i%) > yoff2% THEN
                y2(i%) = y2(i%) - rate
                IF y2(i%) < yoff2% THEN y2(i%) = yoff2%
            END IF
        END IF
    END IF

    IF y2(i%) > 80 AND y2(i%) < 400 THEN
        PSET (x2, y2(i%)), col2%
    END IF
    rate = rate + stepp2 * reso2(1)
NEXT i%
NEXT p%
END SUB

SUB dro3 'To add wave 1 and wave 2
    SHARED ydummy1(), ydummy2(), y(), yoff3%, xoff%, erase%
    SHARED ampl1, ampl2, col3%, xlen%, reso%, reso2, freq1%, freq2%

    m = 0: n = 0
    xdummy = xlen% / 720

    IF erase% THEN
        CALL eraser3
    END IF
    CALL Xaxis3
    reso3(1) = 1
    FOR i = 0 TO 720 STEP reso3(1)
        x = xdummy * i + xoff%
        j = m * freq1%
        IF j > 720 THEN m = 1: j = 0
        h = n * freq2%
        IF h > 720 THEN n = 1: h = 0
        y(i) = y1(j) + y2(h)
        y = y(i) - yoff3%
        IF y > 80 AND y < 400 THEN
            PSET (x, y), col3%
        END IF
        m = m + reso3(1): n = n + reso3(1)
    NEXT i
END SUB
```

```
SUB eraser1 'To erase wave 1 .See dro1 for relevant comments.
SHARED ampl1, offset1%, col1%, freq1%, xlen%, phase1%, eraze%
SHARED wtype1%, ydummy1(), y(), xoff%
```

```
IF period1(1) = 0 THEN EXIT SUB
```

```
xdummy = period1(1) / 720
```

```
freq% = xlen% / period1(1)
```

```
nwaves% = freq%
```

```
FOR p% = 1 TO nwaves%
```

```
  nextcycle = (p% - 1) * period1(1)
```

```
  FOR i% = 0 TO 720 STEP reso1(1)
```

```
    x1 = i% * xdummy + nextcycle + xoff%
```

```
    IF y1(i%) > 80 AND y1(i%) < 400 THEN
```

```
      PRESET (x1, y1(i%))
```

```
    END IF
```

```
  NEXT i%
```

```
NEXT p%
```

```
END SUB
```

```
SUB eraser2 'To erase wave 2. See dro1 for relevant comments.
```

```
SHARED ampl2, offset2%, col2%, freq2%, xlen%, phase2%, eraze%
```

```
SHARED wtype2%, ydummy2(), y(), yoff2%, xoff%
```

```
IF period2(1) = 0 THEN EXIT SUB
```

```
xdummy = period2(1) / 720
```

```
freq% = xlen% / period2(1)
```

```
nwaves% = freq%
```

```
FOR p% = 1 TO nwaves%
```

```
  nextcycle = (p% - 1) * period2(1)
```

```
  FOR i% = 0 TO 720 STEP reso2(1)
```

```
    x2 = i% * xdummy + nextcycle + xoff%
```

```
  IF y2(i%) > 80 AND y2(i%) < 400 THEN
```

```
    PRESET (x2, y2(i%))
```

```
  END IF
```

```
NEXT i%
```

```
NEXT p%
```

```
END SUB
```

```
SUB eraser3 'To erase wave 3
```

```
SHARED y(), xlen%, xoff%, yoff3%, reso3()
```

```
xdummy = xlen% / 720
```

```
FOR i = 0 TO 720 STEP reso3(1)
```

```
  x = xdummy * i + xoff%
```

```
  y = y(i) - yoff3%
```

```
  IF y > 80 AND y < 400 THEN
```

```
    PRESET (x, y)
```

```
  END IF
```

```
NEXT i
```

```
END SUB
```

```
SUB fkeydisplay 'To display functions of selected keys.
```

```
VIEW PRINT
```

```
LOCATE 30: COLOR 4: PRINT "M"; : COLOR 7: PRINT "erge/separate"; SPC(10); :
```

```
COLOR 4: PRINT "R"; : COLOR 7: PRINT "ub_graphics"; SPC(10);
```

```
COLOR 7: PRINT "Deca"; : COLOR 4: PRINT "y"; SPC(10);
```

```
COLOR 4: PRINT "C"; : COLOR 7: PRINT "ipping"
```

```
LOCATE 1
```

```
COLOR 4: PRINT " A"; : COLOR 7: PRINT "mplitude"; SPC(5);
```

```
COLOR 4: PRINT " W"; : COLOR 7: PRINT "avetype"; SPC(5);
```

```
COLOR 4: PRINT " F"; : COLOR 7: PRINT "requency"; SPC(5);
```

```
COLOR 4: PRINT " P"; : COLOR 7: PRINT "hase"; SPC(5);
```

```
COLOR 4: PRINT " L"; : COLOR 7: PRINT "ibrary"; SPC(5);
```

```
PRINT " F"; : COLOR 4: PRINT "o"; : COLOR 7: PRINT "urier"; SPC(5);  
VIEW PRINT 3 TO 28  
END SUB
```

```
SUB formula      'To print formula of the maths eqn of each wave  
SHARED cosex, y, a, wtype1%, wtype2%, phase1%, phase2%  
SHARED i%, freq1%, freq2%, offset1%, offset2%, ampl1, ampl2
```

```
SELECT CASE wtype1%  
CASE 1  
  fmltyp$ = "sin"  
CASE 2  
  fmltyp$ = "cos"  
CASE 3  
  fmltyp$ = "saw"  
CASE 4  
  fmltyp$ = "squ"  
CASE 5  
  fmltyp$ = "tri"  
END SELECT
```

```
LOCATE 10, 2: PRINT "Phase1 = "; phase1%  
LOCATE 10, 72: PRINT "y1 ="  
LOCATE 11, 68: PRINT CINT(ampl1); fmltyp$; freq1%; "x"
```

```
SELECT CASE wtype2%
```

```
CASE 1  
  fmltyp$ = "sin"  
CASE 2  
  fmltyp$ = "cos"  
CASE 3  
  fmltyp$ = "saw"  
CASE 4  
  fmltyp$ = "squ"  
CASE 5  
  fmltyp$ = "tri"  
END SELECT
```

```
LOCATE 23, 2: PRINT "Phase2 = "; phase2%  
LOCATE 23, 72: PRINT "y2 ="  
LOCATE 24, 68: PRINT CINT(ampl2); fmltyp$; freq2%; "x"
```

```
LOCATE 16, 72: PRINT "y ="  
LOCATE 17, 68: PRINT "y1 + y2 "
```

```
END SUB
```

```
SUB frequency      'To change value of frequency. See  
SHARED col1%, col2%, freq1%, freq2%      'sub amplitude for relevant  
SHARED ch1%, ch2%                          'comments  
VIEW PRINT 3 TO 6
```

```
LOCATE 3, 10: COLOR col1%; PRINT "Frequency of 1 = "; freq1%  
LOCATE 3, 40: COLOR col2%; PRINT "Frequency of 2 = "; freq2%
```

```
COLOR 7
```

```
LOCATE 4: PRINT "Change 1 or 2 ? (select)";
```

```
DO
```

```
DO WHILE NOT INKEY$ = "" : LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
```

```
a$ = INPUT$(1): LOCATE 4, 25: PRINT a$; SPC(5);
```

```
a% = VAL(a$)
```

```
IF a% = 1 THEN
```

```
  PRINT "press d-decrease i-increase esc-quit"
```

```
  freq% = freq1%
```

```
  DO
```

```
    DO WHILE NOT INKEY$ = "" : LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
```

```
    B$ = INPUT$(1)
```

```
    GOSUB frq: freq1% = freq%
```

```
LOCATE 3, 27: PRINT freq1%
LOOP UNTIL B$ = CHR$(27)
PRINT : PRINT : PRINT
ch1% = 1
EXIT SUB
ELSEIF a% = 2 THEN
PRINT "press d-decrease i-increase esc-quit"
freq% = freq2%
DO
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
B$ = INPUT$(1)
GOSUB frq: freq2% = freq%
LOCATE 3, 56: PRINT freq2%
LOOP UNTIL B$ = CHR$(27)
PRINT : PRINT : PRINT
ch2% = 1
EXIT SUB
ELSEIF a$ = CHR$(13) THEN
EXIT SUB
ELSE
BEEP
END IF
LOOP
```

```
frq:
IF LCASE$(B$) = CHR$(ASC("i")) THEN 'CHR$(0) + CHR$(H48)
freq% = freq% + 1
IF freq% > 70 THEN BEEP: freq% = 70
ELSEIF LCASE$(B$) = CHR$(ASC("d")) THEN 'CHR$(0) + CHR$(H5)
freq% = freq% - 1
IF freq% < 1 THEN BEEP: freq% = 1
END IF
RETURN
END SUB
```

```
SUB graphpage 'To print graphics page
SHARED xoff%, xlen%
```

```
FOR y = 100 TO 400 STEP 10
FOR x = xoff% TO xlen% + xoff% STEP 10
PSET (x, y), 1
NEXT x
NEXT y
```

```
FOR y = 100 TO 400 STEP 50
FOR x = xoff% TO xlen% + xoff% STEP 6
PSET (x, y), 1
NEXT x
NEXT y
```

```
FOR x = xoff% TO xoff% + xlen% STEP 90
FOR y = 100 TO 400 STEP 5
PSET (x, y), 1
NEXT y
NEXT x
END SUB
```

```
SUB klean 'To rub off the graphics area of screen
SHARED stat%, stp%, kln%, dror%
LINE (stat%, 100)-(stp%, 400), 0, BF
CALL graphpage
eraze% = 0
dror% = 3
END SUB
```

```
SUB library 'To set conditions for printed selected wave effects.
```



```
SHARED ampl1, ampl2, freq1%, freq2%, stepp1, limit1%, dror%, stat%, stp%
SHARED xlen%, yoff1%, xoff%
a% = 0
VIEW PRINT 3 TO 6
PRINT "1.Modulation 2.Beats 3.decay/damping 4.Clipping (select number)"
DO
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
a$ = INPUT$(1): a% = VAL(a$)
IF a% >= 0 AND a% < 5 THEN
    stat% = 0: stp% = 640 'reset portion of screen to clean.
    CALL klean
    eraze% = 0
    VIEW PRINT 3 TO 28
    SELECT CASE a%
    CASE 1
        ampl1 = 10: ampl2 = 18: yoff1% = 150
        freq1% = 3: freq2% = 22
        LOCATE 12, 2: PRINT " lower "
        LOCATE 13, 2: PRINT " signal"
        CALL dro1
        LOCATE 22, 2: PRINT "carrier"
        LOCATE 23, 2: PRINT " wave "
        CALL dro2
        LOCATE 15, 2: PRINT "Modulated"
        LOCATE 16, 2: PRINT " wave"
        CALL dro3
        EXIT DO
    CASE 2
        ampl1 = 13: ampl2 = 13: yoff1% = 150
        freq1% = 25: freq2% = 22
        LOCATE 12, 2: PRINT " higher "
        LOCATE 13, 2: PRINT " signal"
        CALL dro1
        LOCATE 22, 2: PRINT "slightly"
        LOCATE 23, 1: PRINT "lower wave "
        CALL dro2
        LOCATE 15, 2: PRINT "Beat"
        LOCATE 16, 2: PRINT "signal"
        CALL dro3
        EXIT DO
    CASE 3
        yoff1% = 250: ampl1 = 40: freq1% = 12
        stepp1 = ampl1 / (3 * freq1% * 720)
        LOCATE 15, 2: PRINT "Damped"
        LOCATE 16: PRINT "Oscillation"
        CALL dro1
        stepp1 = 0: yoff1% = 150
        EXIT DO
    CASE 4
        ampl1 = 30: freq1% = 8: yoff1% = 250
        limit1% = 20
        LOCATE 15, 2: PRINT "Clipped"
        LOCATE 16, 2: PRINT " wave"
        CALL dro1
        limit1% = 0: yoff1% = 150
        EXIT DO
    END SELECT
ELSEIF a$ = CHR$(27) THEN
    EXIT DO
ELSE
    BEEP
END IF
LOOP

VIEW PRINT 3 TO 6
PRINT : PRINT : PRINT : PRINT
```

```
LOCATE 3: PRINT "press any key to continue"
DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
B$ = INPUT$(1) *****To allow viewing of the screen for some time.
PRINT : PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
LOCATE 12: PRINT " "
LOCATE 13: PRINT " "
LOCATE 22: PRINT " "
LOCATE 23: PRINT " "
LOCATE 15: PRINT " "
LOCATE 16: PRINT " "
stat% = xoff% - 5: stp% = xoff% + xlen% + 5 'reset portion to clean.
IF a% = 3 OR a% = 4 THEN CALL klean
CALL formula
END SUB
```

```
SUB loader1 'Retrieve values from relevant
SHARED wtype1%, ydummy1() 'tables (on file ) and store in memory
SHARED dror%, ydummy2(), wtype2%
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7
SELECT CASE wtype1%
CASE 1
OPEN "sine.tbl" FOR INPUT AS #4
CASE 2
OPEN "cosine.tbl" FOR INPUT AS #4
CASE 3
OPEN "sawtooth.tbl" FOR INPUT AS #4
CASE 4
OPEN "square.tbl" FOR INPUT AS #4
CASE 5
OPEN "triangle.tbl" FOR INPUT AS #4
END SELECT
FOR i% = 0 TO 720
INPUT #4, ydummy1(i%)
NEXT i%
CLOSE #4
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
SUB loader2 'see loader one for relevant comments
SHARED wtype1%, ydummy1(), dror%, ydummy2(), wtype2%
```

```
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7

SELECT CASE wtype2%
CASE 1
OPEN "sine.tbl" FOR INPUT AS #5
CASE 2
OPEN "cosine.tbl" FOR INPUT AS #5
CASE 3
OPEN "sawtooth.tbl" FOR INPUT AS #5
CASE 4
OPEN "square.tbl" FOR INPUT AS #5
CASE 5
OPEN "triangle.tbl" FOR INPUT AS #5
END SELECT
FOR i% = 0 TO 720
INPUT #5, ydummy2(i%)
NEXT i%
```

```
CLOSE #5
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
    SUB merge
    SHARED ch2%, yoff2%, dror%, yoff1%, ch1%
    ch1% = 1: ch2% = 1
    CALL klean
    CALL graphpage
    IF yoff1% = 150 AND yoff2% = 350 THEN
        yoff1% = 250: yoff2% = 250
    ELSE
        yoff1% = 150: yoff2% = 350
    END IF
    dror% = 3
    END SUB
```

```
    SUB phase                                'To change phase of waves.
    SHARED phase1%, phase2%, ch1%, ch2%, col1%, col2% 'See sub amplitude for
                                                'relevant comments.
```

```
    VIEW PRINT 3 TO 6
    LOCATE 3, 10: COLOR col1%: PRINT "Phase of 1 = "; phase1%
    LOCATE 3, 40: COLOR col2%: PRINT "Phase of 2 = "; phase2%
```

```
    COLOR 7
    LOCATE 4: PRINT "Change 1 or 2 ? (select)";
    DO
    DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
    a$ = INPUT$(1): LOCATE 4, 25: PRINT a$: SPC(5);
    a% = VAL(a$)
    IF a% = 1 THEN
        PRINT "press d-decrease i-increase esc-quit"
        phs% = phase1% * 2
        DO
            DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
            B$ = INPUT$(1)
            GOSUB phs: phase1% = phs% / 2
            LOCATE 3, 23: PRINT phase1%
            LOOP UNTIL B$ = CHR$(27)
            PRINT : PRINT : PRINT
            ch1% = 1
            EXIT SUB
        ELSEIF a% = 2 THEN
            PRINT "press d-decrease i-increase esc-quit"
            phs% = phase2% * 2
            DO
                DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
                B$ = INPUT$(1)
                GOSUB phs: phase2% = phs% / 2
                LOCATE 3, 52: PRINT phase2%
                LOOP UNTIL B$ = CHR$(27)
                PRINT : PRINT : PRINT
                ch2% = 1
                EXIT SUB
            ELSEIF a$ = CHR$(13) THEN
                EXIT SUB
            ELSE
                BEEP
            END IF
        LOOP
```

```
    phs:
    IF LCASE$(B$) = CHR$(ASC("i")) THEN
        phs% = phs% + 2
        IF phs% > 360 THEN BEEP: phs% = 360
```

```
ELSEIF LCASE$(B$) = CHR$(ASC("d")) THEN
  phs% = phs% - 2
  IF phs% < -360 THEN BEEP: phs% = -360
END IF
RETURN
END SUB

SUB ruboff
SHARED erase%, dror%, ch1%, ch2%
ch1% = 1: ch2% = 1
CALL klean
erase% = 0
END SUB

SUB wavetype ' To change type of wave
SHARED wtype1%, wtype2%, ch1%, ch2%

VIEW PRINT 3 TO 6
LOCATE 3, 5
PRINT "1.sin 2.cos 3.sawtooth 4.square 5.triangular...(select number) ";
DO
  DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
  typ$ = INPUT$(1): PRINT typ$
  t% = VAL(typ$)
  IF t% = 1 OR t% = 2 OR t% = 3 OR t% = 4 OR t% = 5 THEN
    PRINT "Change type of wave 1 or wave 2 ? (select number) "
    DO
      DO WHILE NOT INKEY$ = "": LOOP 'TO EMPTY KEYBOARD INPUT BUFFER
      a$ = INPUT$(1): LOCATE 4, 54: PRINT a$
      IF a$ = CHR$(ASC("1")) THEN
        ch1% = 1 'Flag to show that
        wtype1% = t% 'change has taken
        CALL loader1
        EXIT DO
      ELSEIF a$ = CHR$(ASC("2")) THEN
        ch2% = 1
        wtype2% = t%
        CALL loader2
        EXIT DO
      ELSEIF a$ = CHR$(27) THEN EXIT DO
      ELSE BEEP
      END IF
    LOOP
  EXIT DO
ELSEIF typ$ = CHR$(27) THEN EXIT DO
ELSE BEEP
END IF
LOOP
VIEW PRINT 3 TO 5
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB

SUB Xaxis1 'To draw equilibrium line for wave 1
SHARED yoff1%, xlen%, xoff%
FOR i% = 0 TO xlen% STEP 6
  PSET (i% + xoff%, yoff1%), 1
NEXT i%
END SUB

SUB Xaxis2 'To draw equilibrium line for wave 2
SHARED yoff2%, xlen%, xoff%
FOR i% = 0 TO xlen% STEP 6
  PSET (i% + xoff%, yoff2%), 1
NEXT i%
END SUB
```

```
SUB Xaxis3 'To draw equilibrium line for wave 3
SHARED xlen%, yoff3%, xoff%
FOR i% = 0 TO xlen% STEP 6
PSET (i% + xoff%, yoff3%), 1
NEXT i%
END SUB
```

APPENDIX E

ASCII TEXT FILE FOR TUTORIAL SECTION

WAVE

TUTOR

page1
CHAPTER 1

=====

INTRODUCTION
=====

Probably the most essential object to life as we know it is the SUN. It enables us to see during the day as well as providing invisible ultraviolet light, which is essential for our skin to manufacture vitamin D. The sun produces light (and other radiations) which travels for to get to us on earth. This is achieved by wave motion.

Sound and consequently speech is also transferred from one place to another by wave motion. Wireless communication via radio, television, and mobile telephones, is achieved by wave propagation.

However, despite this preponderance of waves around us, many are still intrigued by the nature of waves. One of the reasons for this, is that wave motion involves atoms or molecules which are microscopic (too small to be seen).

Another reason lies in the nature of the wave itself. A wave exists only when there is motion. Any passive method of learning it cannot suffice.

page2
Displacement
=====

The most basic type of motion, is movement from one place to another in a particular direction. Press any key.

A <----- X -----> B

The block has moved from A to B in a straight line. It has been displaced from A to B by an amount X metres.

If the body had moved from A to B following an irregular path, (curved path), then it would have moved by a distance Y metres instead. The basic difference between displacement and distance is the direction. Displacement is a vector quantity (it has magnitude and direction), while distance is a scalar quantity (it has magnitude only).

page3

circular motion

=====

A satellite (white circle) is shown stationary in its orbit (big red circle). Press any key.

This is an example of circular motion. The basic difference between this and linear motion is that, the trajectory can be traversed repeatedly. Any point along the trajectory is accessible to the satellite after a period of time. The earth moves round the sun in an approximate circular path. There are numerous other examples in nature.

page4

Vibratory motion

=====

Another type of motion involves a body or object moving to and fro about a fixed position. The diagram above illustrates a small heavy mass (red ball), suspended from a rigid support (blue line) by a light inextensible string. The arrangement is called a pendulum. If the mass is displaced from its rest position (press any key), and released (press another key), it swings from one side to another.

This type of motion is called OSCILLATORY or VIBRATORY motion.

page5

The motion is repeated continuously so far as the oscillating system possesses (or continuously receives) energy to sustain the motion. However, the energy possessed is gradually exhausted (or stops in case of an external source).

The maximum displacement of the bob from its rest position reduces accordingly. This reduction is called damping. In the case of the pendulum, the energy of the system is derived from the gravitational potential energy of the bob (white ball), when it is initially displaced (press any key). Damping is mainly due to work done against air resistance. Therefore damping is not easily noticed. However, when left to oscillate for a sufficiently long time, the pendulum comes to rest at its equilibrium position.

In this example damping has been exaggerated, for easy comprehension.

page6

In some cases, 'heavy' damping is desirable. An example is the shock absorbing mechanism of automobiles. When a car runs into a pothole, the tyres are displaced from their equilibrium position. The shock absorbers attempt to pull back the tyres. In that bid, oscillation results. Consequently, the car begins jerk up and down. This can be quite uncomfortable for the occupants of the car. Hence the system is designed to stop the oscillation (heavy damping) as rapidly as possible.

page7

atoms

=====

Natural constraint to free motion or oscillation is exemplified in the arrangement of the atoms. Atoms are the 'building blocks' of matter. They are so small that they cannot be seen. The specific position of an

atom in relation to the others is maintained because of interatomic forces depending on the natural arrangement of atoms for a particular substance.

page8

At room temperature, each atom vibrates about its rest position. The amplitude of vibration increases as the room temperature increases. Below is an isolated atom. Its size and amplitude of vibration has been exaggerated to enhance understanding of how it vibrates. (Press any key).

page9

In a substance, the separation between one atom and another is very small. Any atom that is displaced towards another atom suffers repulsion while simultaneously, the atom on its other side attracts it. Hence, it is quickly restored to its original position. Unless the atom is displaced repeatedly, its oscillation is shortlived. Its oscillation is critically damped.

However, due to the thermal energy possessed by all the atoms of a substance, they all vibrate with minimal amplitude about their equilibrium positions. The oscillation is sustained by a continuous supply of energy drawn from the environment. If the energy increases, atomic thermal energy also increases which lead to higher amplitudes of vibration.

page10

If a substance is disturbed at a point, the disturbance is transmitted through the substance by transfer from one atom to another. The diagram below shows a layer of atoms. Imagine that the first atom is given an initial push (press any key). As it is displaced towards the second atom, they both repel one another. This retards the motion of atom 1 while setting atom 2 into motion too. Eventually atom 1 is turned back to its equilibrium position.

Meanwhile, atom 2 begins to experience repulsive forces as it is further displaced towards atom 3. The process continues until the last atom is affected. Apart from repulsive force, the restoration of the atoms to their equilibrium position is enforced by the attractive forces of the atoms around it. In fact, this is how the last atom is restored (since there is no repulsive force in the direction of its displacement). Press any key to stop the vibration.

page11

The successive displacement AND recovery of the atoms, has transmitted the initial push given to the first atom to the last, without the atoms completely leaving their position. They have merely been displaced for a very short period of time. This is an example of how energy can be transmitted from one point to another through a substance. A wave has been transmitted. In practice, a push is usually accompanied by a pull in the opposite direction. Hence each atom moves to and fro (vibrates) about its equilibrium position while the wave is transmitted through the substance.

page12

The type of wave described above is called a LONGITUDINAL wave because the direction of vibration is the same as the direction of propagation (transmission) of the wave. For a TRANSVERSE wave, the direction of vibration is perpendicular to the direction of propagation of the wave. That is each atom is displaced vertically (or horizontally) while the wave is propagated to the right of the (or left).

page13

At rest, the position of each molecule is regarded as zero displacement. Let us assume that displacements to the right of the rest position are considered to have positive values while those to the left of the rest position are negative. (Press any key).

By measuring and recording displacement at regular time intervals, (also recorded, the relationship between displacement and time as a wave is propagated through a medium can be better comprehended. The set of readings

obtained can be used to plot graph for figurative illustration of the relationship.

page 14
CHAPTER 2

=====

The diagram below is a graphical representation of the relationship between displacement and time as a wave is propagated through the medium. It can be represented by the mathematical equation

$$y = A \sin k(x)$$

where A = amplitude, k = 'frequency', x = variable
from 0 to 2π radians
or from 0 to 360
degrees .

page 15

The red curve is the displacement of y as x varies from 0 to 2π .
The blue line represents the 'equilibrium position i.e. $y = 0$.
 x increases linearly from left to right of the screen.

Now , press letter ' A ' , once (wait to observe the effect),
and twice again.

page 16

Notice how the curve bulged at its highest and lowest points. The vertical distance from the highest point, on the curve, to the equilibrium line, or , the vertical distance from the lowest point, on the curve to the equilibrium line, is called the

AMPLITUDE

of the wave.

page 17

The wave shown is for one cycle only. If x increases from 2π to 4π , another curve, exactly the same as the former is produced.
Press any key. Another wave will be produced.

The wave is continuously reproduced as the value of x increases . Many situations in nature give rise to these variations . A major difference exists however . Changes in x , are replaced by changes in time. That is, displacements, as represented by y , varies sinusoidally (like a sine function), with time.

page 18

Examples include:

- Sound waves,
- Mechanical waves,
- Electromagnetic waves,
- Electrical (alternating) current,
- and many more.

Each of the examples mentioned above can be further subdivided, depending on certain criteria .

Waves, in general , are means of transmitting energy from one point to another.

In practice, waves are more complicated than the basic one considered.

page 19

Let us examine more of the attributes of a wave.

The wave above is for one complete cycle. Notice that the wave is longer than the one shown previously . The wave was deliberately extended to aid further studies. In reality, this will correspond to increasing the

period of the wave.

PERIOD

of a wave is the time it takes to complete one cycle (oscillation).

page20

Another wave (green) has been introduced. How many cycles has the green wave?

(Type the number and press enter)

Good. The frequency of the green wave is twice that of the red wave.

frequency

of a wave is the number of complete cycles (oscillations) in one second.

page21

WAVELENGTH

of a wave is defined as the distance between two successive points in phase.

Consider the diagram shown below. The white line is the wavelength, for each wave. It has been drawn from several points in phase with one another. Note that the line is shorter for the red wave. Which of the two waves has the greater frequency, red or green?

Very good. The frequency of the red wave is greater but the wavelength is shorter. The frequency is inversely proportional to the wavelength.

The wavelength can be measured from other positions, on the wave(s) as shown below. It is clearer to observe the wavelength as the distance between two successive crests (or two successive troughs).

page22

When two waves are incident in the same medium, each one of them behaves as if the other is not there. Their overall effect is the addition of the two waves.

Let us see the addition of the two waves shown below. Press any key.

page23

This is called superposition of waves. The same principle holds for any number of waves in a medium.

Various complicated waves are produced when the attributes of one wave are varied with respect to another, as you will soon see. This is often how waves abound in nature, especially, sound waves.

page24

Now compare the two waves below.

One is exactly the opposite of the other. They are said to be π radians, or 180 degrees, out of phase with each other. What do you think will happen when added together? Press any key.

page25

The waves cancelled out. That is why the displacement of the blue wave is zero. This is an example of destructive interference. If the amplitude of one is different from that of the other, the resultant wave will have an amplitude which is the difference of the two.

If A_1 and A_2 are the amplitudes, then the resultant amplitude is,

$$A = A_1 - A_2$$

page26

You have probably discovered what will happen if two waves, exactly the same are added together. In case you have not, let us look at it.

page27

Notice that the amplitude of the blue wave is double that of any of the other two. In fact their amplitudes added up. This is a good example of constructive interference.

In general, if the amplitudes are different , A_1 and A_2 , for an example, then the resultant amplitude is,

$$A = A_1 + A_2$$

page28

A wave that is $\pi/2$ radians or 90 degrees out of phase with another is called a cosine of the other. The green is the cosine of the red wave as displayed below.

Notice that the wave did not start from the 'equilibrium' line. It started 'earlier', though you cannot see it.

page29

You can enhance your understanding of all the topics in the tutorial by practicing on your own. The next part of WORLD OF WAVES enables you to do this.

H A V E F U N

Press "b" to review the previous page , "esc" to quit WAVE TUTOR, any other key to proceed to the the WAVE DRAW environment for practice.

APPENDIX F

PROGRAM LIST FOR FOURIER-DRAW SECTION

```
REM /* program for the fourier-draw section of "world of waves" */
REM /* Filename is "wvfuria.drv" */
DECLARE SUB dro2 ()
DECLARE SUB eraser2 ()
DECLARE SUB Xaxis1 ()
DECLARE SUB Xaxis2 ()
DECLARE SUB Xaxis3 ()
DECLARE SUB klean ()
DECLARE SUB loader2 ()
DECLARE SUB loader1 ()
DECLARE SUB tria ()
DECLARE SUB squ ()
DECLARE SUB saw ()
DECLARE SUB fadd ()
DECLARE SUB fund ()
DECLARE SUB foumanual ()
DECLARE SUB fouauto ()

DIM SHARED ydummy1(721), ydummy2(721), period2(1), reso2(1), y2(721)
DIM SHARED y(721)

COMMON SHARED chain$

xlen% = 360: yoff1% = 150: yoff2% = 350: yoff3% = 250: xoff% = 140
col1% = 4: col2% = 2: col3% = 3
stat% = 0: stp% = 640: ampl2 = 0: wtype1% = 1: wtype2% = 1: last% = 20
phase1% = 0: phase2% = 0: xdummy = xlen% / 720: erase% = 0: kount% = 25

SCREEN 12
CALL loader1: CALL loader2
DO
CLS
VIEW PRINT
LOCATE 1, 30: COLOR 14: PRINT "FOURIER ANALYSIS"
COLOR 7
e% = 1
LOCATE 6, 10: PRINT "1. Auto-generate or 2. Manual ? (select number)";
DO
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1): W% = VAL(W$)
IF W% = 1 THEN
CALL fouauto: EXIT DO
ELSEIF W% = 2 THEN
CALL foumanual: EXIT DO
ELSEIF W$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
END IF
LOOP
COLOR 14
PRINT "Quit Fourier Analysis ? (y/n) ";
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1)
IF LCASE$(W$) = "y" THEN EXIT DO
LOOP

IF chain$ = "on" THEN
VIEW PRINT 27 TO 28
LOCATE 27: COLOR 15: PRINT "Returning to main module...."
VIEW PRINT 3 TO 28: COLOR 7
```

```
CHAIN "waves.drv"
END IF
END

SUB dro2 'To draw each harmonic before addition. No maths implication.
SHARED ampl2, col2%, freq2%, xlen%, phase2%, erase%
SHARED reso2, wtype2%, ydummy2(), y(), yoff2%, xoff%, yoff3%
SHARED stepp2, limit2%, dror%

IF erase% = 1 THEN 'erase previously drawn wave
CALL eraser2
END IF

IF phase2% < 0 THEN
faze% = 720 - ABS(2 * phase2%)
ELSE
faze% = 2 * phase2%
END IF

q = (ampl2 * SIN(70)) / (70 * freq2%) 'Set resolution of screen
IF q < .2 THEN q = .2 'Necessary for optimum speed
reso2(1) = 1 / q

k% = 0
i% = faze%
DO
z = ydummy2(i%) * ampl2
y2(k%) = z + yoff2%
k% = k% + 1
i% = i% + 1
IF i% > 720 THEN i% = 1
LOOP UNTIL k% = 721
rate = 0
nwaves% = freq2%
period2(1) = xlen% / freq2%
xdummy = period2(1) / 720
CALL Xaxis2
FOR p% = 1 TO nwaves%
nextcycle = (p% - 1) * period2(1)
FOR i% = 0 TO 720 STEP reso2(1)
x2 = i% * xdummy + nextcycle + xoff%
PSET (x2, y2(i%)), col2%
NEXT i%
NEXT p%
END SUB

SUB eraser2 'To erase previously drawn harmonic
SHARED ampl2, col2%, freq2%, xlen%, phase2%, erase%
SHARED wtype2%, yoff2%, xoff%
CALL Xaxis2
xdummy = period2(1) / 720
freq% = xlen% / period2(1)
nwaves% = freq2%
FOR p% = 1 TO nwaves%
nextcycle = (p% - 1) * period2(1)
FOR i% = 0 TO 720 STEP reso2(1)
x2 = i% * xdummy + nextcycle + xoff%
PRESET (x2, y2(i%))
NEXT i%
NEXT p%
END SUB

SUB fadd 'For addition of waves
SHARED ampl2, col3%, xlen%, reso, freq2%, e%
SHARED y(), wtype1%, wtype2%, xoff%, col1%, phase1%, yoff3%
SHARED phase2%, erase%, wty$, konstant, ampli, kount%, eqn$
```

```
xdummy = xlen% / 720
IF e% <> 1 THEN      'e% signifies the no. of harmonic
  VIEW PRINT 3 TO 28
  LOCATE 10, 2: PRINT "Last addition"
  CALL Xaxis1
  FOR i% = 0 TO 720
    x = xdummy * i% + xoff%
    y = y(i%) + 150
    PSET (x, y), col1%
  NEXT i%
END IF

LOCATE 22, 2: PRINT "Harmonic "; e%
LOCATE 22, 64
PRINT (ampl2); TAB(68); wty$; TAB(71); freq2%; TAB(75); "x"
eqn$ = eqn$ + "+" + " " + STR$(ampl2) + " " + wty$
eqn$ = eqn$ + STR$(freq2%) + "x" + " "
CALL Xaxis2
IF ABS(ampl2) < 1 THEN
  VIEW PRINT 26 TO 27
  LOCATE 26, 15: PRINT "Amplitude of this harmonic is visually amplified."
  END IF
  ampl = ampl2
  times = 5
  DO
  IF ampl2 = 0 THEN EXIT DO
  ampl2 = ampl2 * times
  IF ABS(ampl2) < 1 THEN
    times = times + 2
  ELSE
    EXIT DO
  END IF
  LOOP
  CALL dro2
  PRINT : PRINT : PRINT
  VIEW PRINT 3 TO 28

n% = 0
LOCATE 16, 2: PRINT "New addition"
CALL Xaxis3
FOR i% = 0 TO 720
  x = xdummy * i% + xoff%
  h% = n% * freq2%
  IF h% > 720 THEN n% = 1: h% = 0
  y(i%) = y(i%) + ydummy2(h%) * ampl
  y = y(i%) + 250
  PSET (x, y), col3%
  n% = n% + 1
NEXT i%
END SUB

SUB fouauto
  SHARED ampl2, col3%, xlen%, reso, freq2%, e%
  SHARED y(), wtype1%, wtype2%, xoff%, col1%, phase1%, yoff3%
  SHARED phase2%, erase%

  DO
  VIEW PRINT 5 TO 6
  PRINT : PRINT : PRINT
  LOCATE 6: PRINT "1.Square 2.Triangular 3.Sawtooth (select type to generate)";
  DO
  DO WHILE NOT INKEY$ = "": LOOP      'To empty keyboard input buffer
  W$ = INPUT$(1)
  IF W$ = "1" THEN
    CALL squ: EXIT DO
  ELSEIF W$ = "2" THEN
```

```
CALL tria: EXIT DO
ELSEIF W$ = "3" THEN
CALL saw: EXIT DO
ELSEIF W$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
END IF
LOOP
PRINT "Quit auto-generation ? (y/n)"
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1)
IF LCASE$(W$) = "y" OR W$ = CHR$(27) THEN EXIT DO
LOOP
```

END SUB

SUB foumanual

```
SHARED ampl2, col3%, xlen%, reso, freq2%, e%
SHARED y(), wtype1%, wtype2%, xoff%, col1%, phase1%, yoff3%
SHARED phase2%, eraze%, wty$, eqn$
e% = 1
DO
VIEW PRINT 5 TO 7
PRINT : PRINT : PRINT : PRINT
LOCATE 5
PRINT "1.sin 2.cos (Choose type of "; e%; : PRINT " harmonic) "
DO
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1): LOCATE 5, 50: PRINT W$
W% = VAL(W$)
IF W% = 1 OR W% = 2 THEN
IF W% = 1 THEN wty$ = "sin" ELSE wty$ = "cos"
IF wtype2% <> W% THEN wtype2% = W%: CALL loader2
DO
LOCATE 6: INPUT "Enter frequency "; W$
W% = VAL(W$)
IF W% > 1 AND W% < 50 THEN
freq2% = W%: EXIT DO
ELSE BEEP
END IF
LOOP
EXIT DO
ELSEIF W$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
END IF
LOOP
DO: LOCATE 7
INPUT "Enter amplitude "; W$
W = VAL(W$)
IF W > -70 AND W < 70 THEN
ampl2 = W: EXIT DO
ELSEIF W$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
END IF
LOOP
IF e% = 1 THEN
CALL fund
ELSE
CALL klean
CALL fadd
```

```
END IF

VIEW PRINT 5 TO 7
PRINT : PRINT : PRINT : PRINT
PRINT "press any key to proceed, 'esc' to quit"
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
a$ = INPUT$(1)
IF a$ = CHR$(27) THEN
  PRINT : PRINT : PRINT : PRINT
  VIEW PRINT 27 TO 28
  LOCATE 27: COLOR 14
  PRINT "View cumulative Fourier equation ? y/n";
  DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
  a$ = INPUT$(1)
  PRINT : PRINT : PRINT : PRINT
  IF LCASE$(a$) = "y" THEN
    VIEW PRINT 6 TO 24
    CALL klean
    COLOR 14: LOCATE 10, 10
    PRINT "Cumulative Fourier equation is:"
    COLOR 7
    PRINT eqn$
  END IF
  VIEW PRINT 5 TO 6
  COLOR 7
  PRINT "Quit manual generation ? (y/n)"
  DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
  a$ = INPUT$(1)
  IF LCASE$(a$) = "y" OR a$ = CHR$(27) THEN EXIT DO
ELSE
  e% = e% + 1
  PRINT : PRINT : PRINT : PRINT
END IF
LOOP
END SUB

SUB fund
SHARED ampl2, col3%, xlen%, reso, freq2%, e%
SHARED y(), wtype1%, wtype2%, xoff%, col1%, eqn$, wty$, konstant, ampli

xdummy = xlen% / 720
VIEW PRINT 3 TO 28
CALL klean
LOCATE 10, 2: PRINT "1st harmonic "
LOCATE 11, 2: PRINT "(fundamental)"
LOCATE 10, 64: PRINT ampl2; TAB(68); wty$; TAB(71); freq2%; TAB(75); "x"
eqn$ = STR$(konstant) + " + "+" + STR$(ampl2) + " + wty$ + " "
eqn$ = eqn$ + STR$(freq2%) + "x" + " "
CALL Xaxis1
n% = 0:
FOR i% = 0 TO 720
  x = xdummy * i% + xoff%
  h% = n% * freq2%
  IF h% > 720 THEN n% = 1: h% = 0
  y(i%) = ydummy2(h%) * ampl2 + konstant
  y = y(i%) + 150
  PSET (x, y), col1%
  n% = n% + 1
NEXT i%

END SUB

SUB klean
SHARED stat%, stp%, yoff1%, yoff2%
LINE (stat%, yoff1% - 50)-(stp%, yoff2% + 50), 0, BF
END SUB
```



```
SUB loader1
SHARED wtype1%, ydummy1(), dror%
```

```
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7
SELECT CASE wtype1%
CASE 1
OPEN "sine.tbl" FOR INPUT AS #6
CASE 2
OPEN "cosine.tbl" FOR INPUT AS #6
CASE 3
OPEN "sawtooth.tbl" FOR INPUT AS #6
CASE 4
OPEN "square.tbl" FOR INPUT AS #6
CASE 5
OPEN "triangle.tbl" FOR INPUT AS #6
CASE ELSE
BEEP
END SELECT
FOR i% = 0 TO 720
INPUT #6, ydummy1(i%)
NEXT i%
CLOSE
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
SUB loader2
SHARED dror%, ydummy2(), wtype2%
```

```
VIEW PRINT 27 TO 28
COLOR 15
LOCATE 27, 70: PRINT "Loading..."
COLOR 7
SELECT CASE wtype2%
CASE 1
OPEN "sine.tbl" FOR INPUT AS #7
CASE 2
OPEN "cosine.tbl" FOR INPUT AS #7
CASE 3
OPEN "sawtooth.tbl" FOR INPUT AS #7
CASE 4
OPEN "square.tbl" FOR INPUT AS #7
CASE 5
OPEN "triangle.tbl" FOR INPUT AS #7
CASE ELSE
BEEP
END SELECT
FOR i% = 0 TO 720
INPUT #7, ydummy2(i%)
NEXT i%
CLOSE
PRINT : PRINT : PRINT
VIEW PRINT 3 TO 28
END SUB
```

```
SUB saw
SHARED ampli2, col3%, xlen%, reso, freq2%, e%
SHARED y(), wtype1%, wtype2%, xoff%, col1%, phase1%, yoff3%
SHARED phase2%, eraze%, eqn$, wty$, konstant, kount%, ampli, last%
```

```
xdummy = xlen% / 720
DO
PRINT : PRINT : PRINT : PRINT
```

```
LOCATE 5, 5
PRINT "1.Build up 2.Break down to components (choose)";
DO
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1)
LOCATE 5, 60: PRINT W$
PRINT : PRINT : PRINT : PRINT
INPUT "Enter fundamental frequency ( not > 3 ! )"; fundfreq%
IF fundfreq% > 3 OR fundfreq% < 1 THEN BEEP: fundfreq% = 2
PRINT : PRINT : PRINT : PRINT

e% = 1: pi = 22 / 7
ampli = 12
freq2% = fundfreq%: konstant = .5
hamming = .54 + .46 * COS((pi * e% * fundfreq%) / kount%)
DO
ampl2 = -ampli * (1 / (pi * e% * fundfreq%)) * hamming
IF ABS(ampl2) < 12 THEN
ampli = ampli + 2
ELSEIF ABS(ampl2) > 15 THEN
ampli = ampli - 2
ELSE
EXIT DO
END IF
LOOP
IF W$ = "1" THEN
IF wtype2% <> 1 THEN wtype2% = 1: CALL loader2
wty$ = "sin"
CALL fund
ELSEIF W$ = "2" THEN
wtype2% = 3: CALL loader2
wty$ = "sin"
CALL fund
ampli = -ampli
wtype2% = 1: CALL loader2
ELSEIF W$ = CHR$(27) THEN
EXIT DO
ELSE
BEEP
END IF

LOCATE 6
PRINT "1.step by step 2.compute and display result (select number) "
DO
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
a$ = INPUT$(1): LOCATE 6
PRINT " "
IF a$ = "2" THEN
COLOR 1: LOCATE 6: PRINT "Adding harmonic "
FOR e% = 2 TO last%
LOCATE 6, 18: PRINT e%
COLOR 7
freq2% = e% * fundfreq%
hamming = .54 + .46 * COS((pi * e% * fundfreq%) / kount%)
ampl2 = -ampli * (1 / (pi * e% * fundfreq%)) * hamming
eqn$ = eqn$ + "+" + STR$(ampl2) + " " + wty$ + STR$(freq2%) + "x" + " "
FOR i% = 0 TO 720
h% = n% * freq2%
IF h% > 720 THEN n% = 1: h% = freq2%
y(i%) = y(i%) + ydummy2(h%) * ampl2
n% = n% + 1
NEXT i%
NEXT e%

VIEW PRINT 3 TO 28
LOCATE 16, 2: PRINT "Derived wave "
```

```
LOCATE 17, 2: PRINT "after adding"
LOCATE 18: PRINT e% - 1, " harmonics "

CALL Xaxis3
FOR i% = 0 TO 720
  x = xdummy * i% + xoff%
  y = y(i%) + 250
  PSET (x, y), col3%
NEXT i%

ELSEIF a$ = "1" THEN

  DO
    e% = e% + 1
    freq2% = e% * fundfreq%
    hamming = .54 + .46 * COS((pi * e% * fundfreq%) / kount%)
    ampl2 = -ampli * (1 / (pi * e% * fundfreq%)) * hamming
    CALL klean
    IF W$ = "1" THEN wty$ = "sin" ELSE wty$ = "cos"

  IF e% > last% THEN
    PRINT : PRINT : PRINT : PRINT
    PRINT "Limit of iteration. Cannot continue."
    CALL eraser2
    EXIT DO
  ELSE
    CALL klean: CALL fadd
    VIEW PRINT 5 TO 6
    PRINT : PRINT : PRINT : PRINT
    LOCATE 6: PRINT "Press any key to continue, 'esc' to quit."
    DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
    b$ = INPUT$(1)
    IF b$ = CHR$(27) THEN
      PRINT : PRINT : PRINT : PRINT
      CALL eraser2: EXIT DO
    END IF
  END IF
  LOOP
  ELSEIF a$ = CHR$(27) THEN EXIT DO
  ELSE BEEP
  END IF

VIEW PRINT 3 TO 28
LOCATE 22, 2: PRINT "Model of wave"
LOCATE 23, 2: PRINT " desired"
ampl2 = 14: freq2% = fundfreq%:
IF W$ = "1" THEN
  wtype2% = 3: CALL loader2
ELSE
  wtype2% = 1: CALL loader2
END IF
CALL Xaxis2: CALL dro2
VIEW PRINT 27 TO 28
LOCATE 27
COLOR 14
PRINT "View cumulative Fourier equation ? y/n";
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
a$ = INPUT$(1)
PRINT : PRINT : PRINT : PRINT
IF LCASE$(a$) = "y" THEN
  CALL klean
  VIEW PRINT 6 TO 24
  LOCATE 10, 10
  PRINT "Cumulative Fourier equation is:"
  COLOR 7
  PRINT eqn$
```

```
    END IF
    COLOR 7
    EXIT DO
LOOP
EXIT DO
LOOP
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT : PRINT
PRINT "Quit sawtooth wave ? (y/n)"
DO WHILE NOT INKEY$ = "": LOOP    'To empty keyboard input buffer
W$ = INPUT$(1)
CALL klean
IF LCASE$(W$) = "y" THEN EXIT DO
LOOP

END SUB

SUB squ
SHARED ampl2, col3%, xlen%, freq2%, e%
SHARED y(), wtype1%, wtype2%, xoff%, col1%, phase1%, yoff3%
SHARED phase2%, eraze%, eqn$, wty$, konstant, amplii, kount%, last%
    xdummy = xlen% / 720
DO
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT : PRINT
LOCATE 5, 5
PRINT "1.Build up 2.Break down to components (choose)";
DO
DO WHILE NOT INKEY$ = "": LOOP    'To empty keyboard input buffer
W$ = INPUT$(1)
LOCATE 5, 60: PRINT W$
PRINT : PRINT : PRINT : PRINT
INPUT "Enter fundamental frequency ( not > 3 ! )"; fundfreq%
IF fundfreq% > 3 OR fundfreq% < 1 THEN
    BEEP: fundfreq% = 2
END IF
PRINT : PRINT : PRINT : PRINT
e% = 1: pi = 22 / 7
ampli = 8: freq2% = (2 * e% - 1) * fundfreq%
konstant = 0
hamming = .54 + .46 * COS((pi * (2 * e% - 1) * fundfreq%) / kount%)
DO
ampl2 = ampli * (4 / (pi * (2 * e% - 1) * fundfreq%)) * hamming
IF ABS(ampl2) < 12 THEN
    ampli = ampli + 2
ELSEIF ABS(ampl2) > 18 THEN
    ampli = ampli - 2
ELSE
    EXIT DO
END IF
LOOP
IF W$ = "1" THEN
    IF wtype2% <> 1 THEN wtype2% = 1: CALL loader2
    CALL klean
    wty$ = "sin"
    CALL fund
ELSEIF W$ = "2" THEN
    wtype2% = 4: CALL loader2
    wty$ = "sin"
    CALL fund
    ampli = -ampli
    wtype2% = 1: CALL loader2
ELSEIF W$ = CHR$(27) THEN
    EXIT DO
ELSE
    BEEP
```

```
END IF

LOCATE 5
PRINT "1.step by step    2.compute and display result (select number) "
DO
  DO WHILE NOT INKEY$ = "": LOOP    'To empty keyboard input buffer
  a$ = INPUT$(1): LOCATE 5
  PRINT "
  IF a$ = "2" THEN
    COLOR 1: LOCATE 6: PRINT "Adding harmonic "
    FOR e% = 2 TO last%
      LOCATE 6, 18: PRINT e%
      COLOR 7
      freq2% = (2 * e% - 1) * fundfreq%
      hamming = .54 + .46 * COS((pi * (2 * e% - 1) * fundfreq%) / kount%)
      ampli2 = ampli * (4 / (pi * (2 * e% - 1) * fundfreq%)) * hamming
      eqn$ = eqn$ + "+" + STR$(ampli2) + " " + wty$ + STR$(freq2%) + "x" + " "
    FOR i% = 0 TO 720
      h% = n% * freq2%
      IF h% > 720 THEN n% = 1: h% = freq2%
      y(i%) = y(i%) + ydummy2(h%) * ampli2
      n% = n% + 1
    NEXT i%
  NEXT e%

VIEW PRINT 3 TO 28
LOCATE 16, 2: PRINT "Derived wave "
LOCATE 17, 2: PRINT "after adding"
LOCATE 18: PRINT e% - 1; " harmonics "

CALL Xaxis3
FOR i% = 0 TO 720
  x = xdummy * i% + xoff%
  y = y(i%) + 250
  PSET (x, y), col3%
NEXT i%
ELSEIF a$ = "1" THEN
  DO
    e% = e% + 1
    freq2% = (2 * e% - 1) * fundfreq%
    hamming = .54 + .46 * COS((pi * (2 * e% - 1) * fundfreq%) / kount%)
    ampli2 = ampli * (4 / (pi * (2 * e% - 1) * fundfreq%)) * hamming
  IF e% > last% THEN
    PRINT : PRINT : PRINT : PRINT
    PRINT "Limit of iteration. Cannot continue."
    CALL eraser2
    EXIT DO
  ELSE
    CALL klean: CALL fadd
    VIEW PRINT 5 TO 6
    PRINT : PRINT : PRINT : PRINT
    LOCATE 6: PRINT "Press any key to continue, 'esc' to quit."
    DO WHILE NOT INKEY$ = "": LOOP    'To empty keyboard input buffer
    b$ = INPUT$(1)
    IF b$ = CHR$(27) THEN
      PRINT : PRINT : PRINT : PRINT
      CALL eraser2: EXIT DO
    END IF
  END IF
END IF
LOOP
ELSEIF a$ = CHR$(27) THEN EXIT DO
ELSE
  BEEP
END IF

VIEW PRINT 3 TO 28
```

```
LOCATE 22, 2: PRINT "Model of wave"
LOCATE 23, 2: PRINT " desired"
AMPL2 = 14: FREQ2% = FUNDREQ%
IF W$ = "1" THEN
  WTYPE2% = 4: CALL LOADER2
ELSE
  WTYPE2% = 1: CALL LOADER2
END IF
CALL Xaxis2: CALL dro2
VIEW PRINT 27 TO 28
LOCATE 27
COLOR 14
PRINT "View cumulative Fourier equation ? y/n";
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
a$ = INPUT$(1)
PRINT : PRINT : PRINT : PRINT
IF LCASE$(a$) = "y" THEN
  VIEW PRINT 3 TO 28
  KLEAN
  LOCATE 10, 10
  PRINT "Cumulative Fourier equation is:"
  COLOR 7
  PRINT eqn$
END IF
COLOR 7
EXIT DO
LOOP
EXIT DO
LOOP
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT : PRINT
PRINT "Quit square wave ? (y/n)"
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1)
IF LCASE$(W$) = "y" THEN EXIT DO
LOOP

END SUB

SUB tria
SHARED AMPL2, COL3%, XLEN%, RESO, FREQ2%, E%
SHARED Y(), WTYPE1%, WTYPE2%, XOFF%, COL1%, PHASE1%, YOFF3%
SHARED PHASE2%, ERAZE%, EQN$, WTY$, KONSTANT, KOUNT%, AMPLI, LAST%
  XDUMMY = XLEN% / 720
DO
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT : PRINT
LOCATE 5, 5
PRINT "1.Build up 2.Break down to components (choose)";
DO
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
W$ = INPUT$(1)
LOCATE 5, 60: PRINT W$
PRINT : PRINT : PRINT : PRINT
INPUT "Enter fundamental frequency ( not > 3 ! )"; FUNDREQ%
IF FUNDREQ% > 3 OR FUNDREQ% < 1 THEN BEEP: FUNDREQ% = 2
PRINT : PRINT : PRINT : PRINT
E% = 1: PI = 22 / 7
FREQ2% = (2 * E% - 1) * FUNDREQ%
AMPLI = 10: KONSTANT = PI / (2 * FUNDREQ%)
HAMMING = .54 + .46 * COS((PI * (2 * E% - 1) * FUNDREQ%) / KOUNT%)
DO
AMPL2 = -AMPLI * (4 / (PI * ((2 * E% - 1) * FUNDREQ%) ^ 2)) * HAMMING
IF ABS(AMPL2) < 14 THEN
  AMPLI = AMPLI + 2
ELSEIF ABS(AMPL2) > 18 THEN
```

```
    ampli = ampli - 2
ELSE
    EXIT DO
END IF
LOOP

IF W$ = "1" THEN
    wtype2% = 2: CALL loader2
    CALL klean
    wty$ = "cos"
    CALL fund
ELSEIF W$ = "2" THEN
    wtype2% = 5: CALL loader2
    CALL klean
    wty$ = "cos"
    CALL fund
    ampli = -ampli
    wtype2% = 2: CALL loader2
ELSEIF W$ = CHR$(27) THEN
    EXIT DO
ELSE
    BEEP
END IF

LOCATE 6
PRINT "1.step by step    2.compute and display result (select number) "
DO
    DO WHILE NOT INKEY$ = "": LOOP    'To empty keyboard input buffer
    a$ = INPUT$(1): LOCATE 6
    PRINT "                "
    IF a$ = "2" THEN
        COLOR 1: LOCATE 6: PRINT "Adding harmonic "
        FOR e% = 2 TO last%
            LOCATE 6, 18: PRINT e%
            COLOR 7
            freq2% = (2 * e% - 1) * fundfreq%
            hamming = .54 + .46 * COS((pi * ((2 * e% - 1) * fundfreq%) / kount%))
            ampl2 = -ampli * (4 / (pi * (2 * e% - 1) ^ 2)) * hamming
            eqn$ = eqn$ + "+" + STR$(ampl2) + "" + wty$ + STR$(freq2%) + "x" + ""
            FOR i% = 0 TO 720
                h% = n% * freq2%
                IF h% > 720 THEN n% = 1: h% = freq2%
                y(i%) = y(i%) + ydummy2(h%) * ampl2
                n% = n% + 1
            NEXT i%
        NEXT e%

VIEW PRINT 3 TO 28
LOCATE 16, 2: PRINT "Derived wave "
LOCATE 17, 2: PRINT "after adding"
LOCATE 18: PRINT e% - 1; " harmonics "

CALL Xaxis3
FOR i% = 0 TO 720
    x = xdummy * i% + xoff%
    y = y(i%) + 250
    PSET (x, y), col3%
NEXT i%
ELSEIF a$ = "1" THEN
    DO
        e% = e% + 1
        freq2% = (2 * e% - 1) * fundfreq%
        hamming = .54 + .46 * COS((pi * ((2 * e% - 1) * fundfreq%) / kount%))
        ampl2 = -ampli * (4 / (pi * ((2 * e% - 1) * fundfreq%) ^ 2)) * hamming

IF e% > last% THEN
```

```
PRINT : PRINT : PRINT : PRINT
PRINT "Limit of iteration. Cannot continue."
CALL eraser2
EXIT DO
ELSE
CALL klean: CALL fadd
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT : PRINT
LOCATE 6: PRINT "Press any key to continue, 'esc' to quit."
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
b$ = INPUT$(1)
IF b$ = CHR$(27) THEN
PRINT : PRINT : PRINT : PRINT
CALL eraser2: EXIT DO
END IF
END IF
LOOP

ELSEIF a$ = CHR$(27) THEN EXIT DO
ELSE BEEP
END IF

VIEW PRINT 3 TO 28
LOCATE 22, 2: PRINT "Model of wave"
LOCATE 23, 2: PRINT " desired"
ampl2 = 14: freq2% = fundfreq%
IF W$ = "1" THEN
wtype2% = 5: CALL loader2
ELSE
wtype2% = 2: CALL loader2
END IF
CALL Xaxis2: CALL dro2
VIEW PRINT 27 TO 28
LOCATE 27
COLOR 14
PRINT "View cumulative Fourier equation ? y/n";
DO WHILE NOT INKEY$ = "": LOOP 'To empty keyboard input buffer
a$ = INPUT$(1)
PRINT : PRINT : PRINT : PRINT
IF LCASE$(a$) = "y" THEN
VIEW PRINT 6 TO 24
CALL klean
LOCATE 10, 10
PRINT "Cumulative Fourier equation is:"
COLOR 7
PRINT eqn$
END IF
COLOR 7
EXIT DO
LOOP
EXIT DO
LOOP
VIEW PRINT 5 TO 6
PRINT : PRINT : PRINT
PRINT "Quit triangular wave ? (y/n)"
DO WHILE NOT INKEYS = "": LOOP 'To empty keyboard input buffer
WS = INPUT$(1)
IF LCASE$(WS) = "y" THEN EXIT DO
LOOP
END SUB

SUB Xaxis1
SHARED yoff1%, xlen%, xoff%
FOR i% = 0 TO xlen% STEP 6
PSET (i% + xoff%, yoff1%), 1
```



```
NEXT i%  
END SUB
```

```
SUB Xaxis2  
SHARED yoff2%, xlen%, xoff%  
FOR i% = 0 TO xlen% STEP 6  
PSET (i% + xoff%, yoff2%), 1  
NEXT i%  
END SUB
```

```
SUB Xaxis3  
SHARED yoff3%, xlen%, xoff%  
FOR i% = 0 TO xlen% STEP 6  
PSET (i% + xoff%, yoff3%), 1  
NEXT i%  
END SUB
```

APPENDIX G

PROGRAM LIST FOR GENERATING THE DATA TABLES

```
REM /* program for generating the five data tables */
REM /* The command system on line 4 of this program is included */
REM /* to safeguard against running the program since */
REM /* the tables are already created. */
```

```
***** NOT MEARNT TO BE RUN SINCE FILES ARE ALREADY CREATED *****
```

```
SYSTEM
x = 0: pi = 22 / 7
OPEN "sine.tbl" FOR OUTPUT AS #4
FOR x = 0 TO 720
sine = SIN(x * pi / 180)
WRITE #1, sine
x = x + .5
NEXT x
```

```
x = 0
OPEN "cosine.tbl" FOR OUTPUT AS #5
FOR x = 0 TO 720
cosine = COS(x * pi / 180)
WRITE #1, cosine
x = x + .5
NEXT x
```

```
OPEN "triangle.tbl" FOR OUTPUT AS #1
stepp = 0
FOR i% = 0 TO 360
tri = stepp
WRITE #1, tri
stepp = stepp + (1 / 360)
NEXT i%
FOR i% = 361 TO 720
stepp = stepp - (1 / 180)
tri = stepp
WRITE #1, tri
NEXT i%
```

```
'square wave values *****
OPEN "square.tbl" FOR OUTPUT AS #2
```

```
stepp = 0
FOR i% = 0 TO 10
squa = stepp
WRITE #2, squa
stepp = stepp + .1
NEXT i%
FOR i% = 11 TO 345
squa = 1
WRITE #2, squa
NEXT i%
FOR i% = 346 TO 355
stepp = stepp - .1
squa = stepp
WRITE #2, squa
NEXT i%
FOR i% = 356 TO 710
```

```
squa = -1
WRITE #2, squa
NEXT i%
FOR i% = 711 TO 720
squa = stepp
WRITE #2, squa
stepp = stepp + .1
NEXT i%

' sawtooth wave values *****
OPEN "sawtooth.tbl" FOR OUTPUT AS #3
stepp = 0
FOR i% = 0 TO 720
sotut = stepp
WRITE #3, sotut
stepp = stepp + (1 / 720)
NEXT i%
CLOSE
system
```