1 Introduction

The microprocessor has become commonplace in our technological society. Everything from dish washers to astronomical telescopes have chips controlling their operation. While the development of applications for computers has been in constant flux since their introduction, the principles of computer operation and of their use in sensing and control have remained stable. Those are the primary subjects of this book. Once a basic understanding of the principles has been built, further detailed knowledge can be acquired later as the need arises.

This book is designed to be accompanied by extensive laboratory work. Over the years the engineering curriculum has focused more and more on the lecture/recitation format. This has led to an ever increasing emphasis on theoretical developments and a loss of contact with the physical basis of engineering and science. The laboratory provides a vital experience in linking theory with physical reality. It also provides the satisfaction of building something and making it work.

Not all computers are suitable for laboratory use. Large mainframe computers are fast and can handle large amounts of data but are awkward to connect to laboratory equipment. At the other end of the scale, microprocessors are included in many laboratory devices but are programmed to perform only a restricted set of duties. Mini and microcomputers have enough speed and memory for all but the most demanding applications but yet are small enough to be dedicated to individual projects and therefore are widely used in the laboratory.

With the technological strides of recent years, microcomputers (or personal computers) have prodigious capabilities. Since they are also used in business, many languages and programs are available; some of which are even useful in a laboratory. With a single microcomputer, an engineer or scientist can acquire data and control an experiment, analyze the data, display the data and analysis as graphs or tables, and write a report or journal article. Remember that it can't do the thinking!

Microcomputers come with many built-in features. Those included depend on the designers' decisions as to what will sell the most computers. Since the demands of the laboratory are so varied, no computer when taken out of the box can hope to fulfil them. Hence to be useful, a computer must be able to change its capabilities after manufacture. This is done in two ways.

One is to provide a method of communication (serial or parallel) between the computer and the laboratory devices (analog to digital converters, voltmeters, etc.) and rely on the devices to be intelligent enough to communicate. Generally this means that the devices need a microprocessor built in which is preprogrammed to communicate with a certain protocol. The other way is to have slots (connectors) in the back of the computer so that circuit boards can be inserted to perform the desired tasks such as analog to digital conversion of serial communication. The computer can then be configured exactly as needed for a particular application. Even the video display or microprocessor can be changed when desired. Further, 'slot machines' are generally the least expensive way to computerize the laboratory.

The IBM-PC (upon which this book is based) and the APPLE IIe (which is the subject of a companion volume) are 'slot machines'. The APPLE IIe (and its predecessors, the APPLE II, the APPLE II+ and its successor APPLE IIGS) has proved its usefulness innumerable times. Its simple yet versatile architecture makes the computer easy to use and understand but limits its analysis and data volume capabilities. The IBM-PC is a newer design which is faster and has more memory but is slightly more complicated internally. They are both quite suitable for laboratory use. Beware of the APPLE IIc which is a slotless version of the IIe; it will not be able to accept the circuit cards which are necessary for the exercises in this book. IBM-PC clones (design copies) can be used instead of the IBM-PC as long as they have a slot where the data acquisition board can be placed and can run the programs. IBM-XT and AT designs can also be used. See Appendix A.

A computer can be treated as a black box which responds in a predictable way to an input; however, that type of use requires a complete knowledge of the possible inputs and responses. An understanding of how it works inside allows the user to figure out how the computer will respond to an input, or even if it can respond. The capabilities and limitations become transparent. Throughout the book a gradual understanding of what goes on inside a computer will be developed.

Other devices which are used are various sensors, analog to digital converters, digital to analog converters, timers, digital input and output devices, optical encoders, stepping motors, and analog amplifiers. They provide the interface between the computer's digital world and the physical phenomena being studied.

Turbo Pascal (version 4.0 or greater) is used for programming throughout the book. It has the advantage of being a structured language making the logical structure of programs more prominent. In addition, many students will have learned Pascal (or another structural language) in introductory programming courses. Turbo Pascal has additional capabilities which are convenient for data acquisition and control. These are the direct memory access instructions (such as Port and Mem) and the Inline statement which allows machine language statements to be inserted into the middle of Pascal

programs. Turbo Pascal also has an integral editor for writing programs and is inexpensive. Turbo Pascal version 3 can also be used with minor changes in the text. The DOS program DEBUG is used to write some of the assembly language programs. A graphing/plotting program is also necessary; see Appendix B for guidelines in choosing one.

The text refers to the 8088 microprocessor. The details inside 8086/80286 based machines are different, however, programming is exactly the same at the level used in the book.

All of this computer work is done in the context of doing physics experiments. These experiments cover subjects not usually emphasized in introductory courses but which have a wide applicability. They show that, with computer control, conceptually sophisticated experiments can be performed with simple apparatus. In particular, the physics of activation temperature, heat diffusion and motion in fluids are explored.

1.1 How to use this book

In this book much of the programming material will be presented by way of example. Programs will be given from which you will be expected to deduce the essence of what is going on and thereby proceed to write your own programs. After using instructions in programs, the more precise description of instructions given in the manuals is easier to comprehend.

This book is written in a tutorial manner in which the exercises and experiments are distributed throughout the text. It would be nice to read up to an exercise and then sit down at the computer to do it. However, the time in the laboratory is short so that this becomes impossible. Before going to the laboratory, read through the text you expect to cover and organize your thoughts about what you will be doing. Also, jot down flow charts and write out programs which you will enter in the computer at the laboratory. Even if they do not run the first time they can be easily changed once the program is in a file. The essence of learning is going through the struggle of getting things to function properly, whether it be in writing programs, building experimental apparatus, or understanding theoretical descriptions.

Details of machine operation can be found in the operator's manual for the machine you are using. This is not a beginner's programming book; an introductory programming course or some experience in computer programming should precede the use of this book.

The appendices contain reference material and extended discussions. They are separated from the main text to improve the flow but contain important information and so should be perused once in a while.

1.2 Chapter summary

Chapter 2 begins with an introduction to the operation of the IBM-PC computer and Turbo Pascal programming. It also contains exercises in using the graphing program you have available.

Chapter 3 introduces the first Input/Output (I/O) device, the Analog to Digital Converter (ADC). It is used to measure the temperature/resistance characteristics of a thermistor. Further Pascal programming is used to do a least squares fit to the data. The I/O capabilities of an 8255 Programmable Peripheral Interface (PPI) chip are used to control a HEXFET switch on a heater to make a temperature controller.

In Chapter 4, simple time delays are used to generate control signals for a stepping motor and the effect of truncation errors is explored. The internal TimeOfDay clock of the IBM is used to make an internal timer.

Chapter 5 concerns an experiment in thermal diffusion. A heater at one end of a copper rod is turned on for a set interval under program control. The flow of this heat pulse down the rod is then measured at two locations for about 30 s. A theoretical model is fitted to these data to determine the thermal conductivity and heat capacity of copper. An analog amplifier is used to boost the signal from the thermistor to the ADC.

Chapter 6 is an introduction to assembly language programming and the internal structure of the IBM-PC. Simple programs are written using Debug and Turbo Inline statements. The efficiency of the Turbo compiler is examined. A Digital to Analog Converter (DAC) is used to make an X–Y plotter.

In Chapter 7, an experiment is constructed which measures the viscosity of glycerine by measuring the speed of a falling sphere. The physics of turbulent as well as smooth fluid flow is discussed. LEDs and photocells are used as position sensors to measure the speed of the sphere.

Chapter 8 introduces the concept of interrupt processing. Programs using software and hardware interrupts are written. Serial data communication is described and instated between two computers.

Chapter 9 contains various topics which are important but do not have a direct bearing on the experiments done in the previous sections.

2 Instrumentation structures and using the IBM-PC

The purpose of an instrument is to make measurements of a particular parameter in a physical process. This requires at least a sensor which responds to the parameter and a display which lets the user record readings which are in some way proportional to the parameter being measured. A thermometer is an instrument which indicates the temperature by quantitatively showing the expansion of a liquid with a temperature increase. A more complete description of the measurement process is shown in Figure 2.1. The arrows show possible but not necessary routes for the flow of information. The computer is able to do many of the tasks which were formerly done by separate units of an instrument. This lets the designer reduce the number of components required to a bare minimum as the experiments in this book show. Many times all that is needed is a sensor to translate the process into an electrical signal.

Another way to think of the computer is as an interface between the experimenter and the experiment (or the user and the measurement). It is able to translate the unintelligible signals from the sensor into a form which is understandable using human senses. One of the best ways of communicating information is by picture. 'A picture is worth a thousand words.' (In fact, it takes roughly a thousand words of computer memory to display a video graphics screen.)

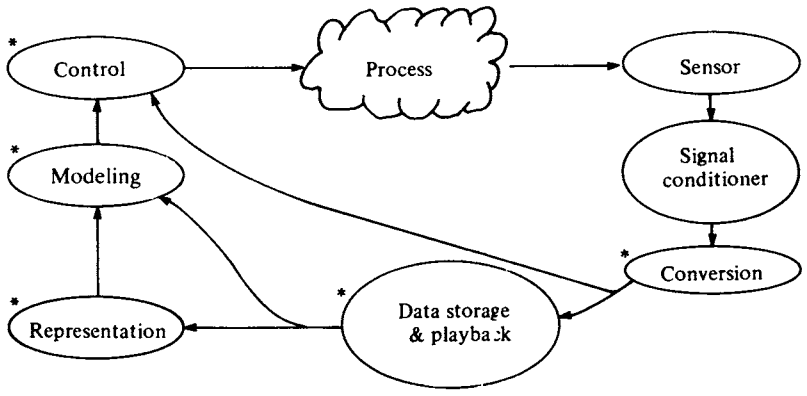

Fig. 2.1. Instrumentation structures
Process: e.g. temperature as function of time, position as function of temperature.
Sensor: e.g. temperature or position converted to voltage.
Signal conditioner: e.g. amplifier, filter.
* Conversion: analog to digital.
* Storage/playback: e.g. silicon memory, magnetic, papertape.
* Representation: e.g. numbers, pictures (1000 words of memory!).
* Modeling: mathematical fit to data.
* Control: e.g. change temperature or position.
The computer has a part in all items with a *.

2.1 First program and graphs

Exercise 2.1.1 Starting out

(*a*) To get started: Insert the SYSTEM START disk into the drive unit marked 'A' and turn the computer power switch on. The disk is set up so that the computer will wake up in the Turbo Pascal system. To see what files are on the disk type 'F' (for File) and 'D' (for Directory) and then the RETURN key (CR, to accept the *.* wildcard mask). In addition to the Turbo files there are other files and programs such as 'COMMAND.COM', the operating system program (DOS) and a graphing program. Type the 'Esc' key twice or 'F10' once to get back to the main menu.

(*b*) Editing: To start writing a program, the program must be in the 'Edit' mode. Type 'E' at the main menu. The computer is now in Edit mode waiting for you to enter some text. 'F5' expands the Edit portion of the screen to full size. 'F5' again contracts it.

(*c*) First program: Type in program CALCCOS.PAS which follows. Each line should be ended with a CR. Use the Editing commands described in the Turbo manual to correct any mistakes you make. The comments can be abbreviated if desired but some should be included as reminders of why the statements are made.

```
program CalcCos;          {all words between brackets are comments}
                          {which are ignored by the compiler}

var                       {all variables used in the program must be}
                          {declared with a type, eg real, integer, string}
    y,x : real;           {all program statements end with a ; symbol}

begin                     {this announces that the main program
                            follows}
  writeln('     Radians     cosine');
                          {type header on the screen}
  x := 0.0;               {assigns the value 0.0 to the variable x}
  repeat                  {starts a repeat-until loop}
    y := cos(x);          {assign y, cos argument is in radians}
    writeln( x : 12 : 4, y : 12 : 4);
                          {type results on screen in format specified}
    x := x + 30*pi/180;   {step in 30 degree increments}
  until x > 2*pi;         {test end of repeat–until loop}
end.                      {end of program has a period not a
                            semicolon}
```

(*d*) Printing: One quick way to get a listing of the program is to use the Print Screen utility. Use 'F5' so that the full Edit screen is displayed. Make sure the printer is on and then press 'Shift PrtSC'. A copy of the screen should be printed on the printer.

(*e*) Running the program: To compile and run the program, either (i) exit the editor with 'F10' and press 'R' for Compile and Run or (ii) press 'ALT R' to go directly to Compile and Run. The computer will compile the program (that is translate the text in the Editor to machine instructions the microprocessor can understand). If there are no errors detected, the program will be run. An error in compiling will put you back in the editor near the place of the error so that you can correct it. After correction, run the program again.

The term compile means to translate the Pascal program to machine language. This is a necessary step because the program is written in text form (letters and digits) and the computer executes programs as binary code. The program can be stored as text and compiled each time it is used or in its binary form in which case it can be run from DOS, ie, outside the Turbo Pascal system. More on this in later chapters.

Also note that the computer makes a distinction between integer numbers (whole numbers) and real numbers (numbers which can have decimals). In general, integers are used for counting and reals for calculations. A different number of bytes is used to store integers and reals. A variable needs to be declared as one or the other so the computer knows how many bytes to access when the variable is used.

In order to use a program more than once, the computer can store the text on a disk and retrieve it later. The results of a program (such as the x,y list of program CalcCos) can also be stored and retrieved as a disk file. However, a disk must be initialized before it can be used. The system program disk is already initialized but it is better to have a separate disk to store your programs. Initialization is done with the DOS FORMAT.COM program. The following exercise shows how to use the program and how to use the disk to store programs and data text files.

Exercise 2.1.2 Saving programs and writing text files

(*a*) Formatting: First exit from Turbo by typing 'F', 'Q' at the main menu (or 'ALT F','Q' from anywhere else). If there is a program in the editor, Turbo will give a warning that the program has not been saved. Answer 'N', you don't want it saved. You should now be at the DOS 'A>' prompt. (If not type 'A:' CR.) Place a new disk in drive B and type FORMAT B: and then CR at the prompt. Be sure to use B: not A: since the FORMAT program erases all the data on

a disk. You don't want to do that to the SYSTEM disk. The computer will spend a minute formatting the disk. When it is done take it out and label it with a felt tip marker.

(*b*) Another program: Put the new disk (call it the SAVE disk) back in drive B and type 'TURBO'. When Turbo has started, you need to tell it that disk operations will now be to drive B. Type 'F', 'C' for Change dir and then 'B : 'CR. Go to the editor and type in and run program WRITECOS . PAS which follows. This program will calculate the same data as program CalcCos but will also save it on disk. Take special note of the way the file statements work.

```
program WriteCos;          {calculates cosine and saves it on the disk}

var
      x,y : real;          {variables for the calculation}
      OutFile : text;      {variable for disk information assignment}
begin
  assign(OutFile, 'COS.DAT');
                           {make assignment to the disk information
                             variable}
  rewrite(OutFile);        {initialize the disk file for writing, this destroys
                           any file on the disk with the same name}
                           {use reset to open an existing file for reading}
  writeln('      Radians      Cosine');
                           {write to the screen}
  writeln( OutFile,'      Radians      Cosine');
                           {write the same header to the disk file}
  x := 0.0;                {same as before}
  repeat
    y := cos(x);
    writeln( x : 12 : 4, y : 12 : 4);
                           {write to the screen}
    writeln( OutFile, x : 12 : 4, y : 12 : 4);
                           {this writes the same information to the disk}
   x :=x+30*pi/180;
  until x>2*pi;
  close(OutFile);          {write the rest of the buffer to the file and}
                           {write an end of file marker}
end.
```

(*c*) Save the program: There are two ways to save a program file on the disk: if the file name which appears at the top of the edit window is the one you want, use 'F2' or 'ALT F','S' and the file will be saved under that name. Any previous file with that name will be renamed

with a .BAK extension. If you want to rename the file while saving it, use 'ALT F','W' which will ask you for a new name to use. Use the second method here and name the file WRITECOS.PAS. Note if you just write WRITECOS the .PAS will be added automatically.

(*d*) Viewing text: To see the text created by the program you wrote, use the editor as follows. Type 'ALT F','L' for load and then the name of the file you wish to see, ie 'COS.DAT. Then enter the editor to see the numbers. To get the program back, use 'ALT F','L' again and enter the program name. Go back in the editor to make sure it is there.

(*e*) DOS commands: There is another way to view text. Exit the editor and type 'ALT L','Q' to exit Turbo. Now type 'TYPE COS.DAT' and the data file should appear on the screen. If it goes too fast, type 'Ctrl-S' to stop the text. Type any key to restart it. Another useful DOS command is PRINT. 'A : PRINT file' will print a copy of a file on the printer. Try printing your program and data file. Typing 'A : TURBO' will get you back to the Turbo Pascal program.

Graphing experimental data and mathematical expressions is an important aspect of the work outlined in this book. Many graphics programs are not suited to drawing graphs of data; they are written to draw pictures. On the IBM-PC there are several ways in which engineering graphs can be constructed. Turbo Pascal has some primitive graphing functions (PutPixel and LineTo) which allow a quick plot to be done from within a program. A library of functions can be constructed to add scales and text and make more elaborate plots. Another way to make graphs is to use a stand alone graphing program. The program reads data from text files created with an editor or a program and can plot it in several different ways with labeled axes, titles, etc. See Appendix B for a list of suitable programs.

Exercise 2.1.3 Simple graphing

Using the graphing method available on your computer, plot the curve $Y = X*X - 1$ from $X = -2$ to $+2$. You may need to write a program first to create a data file of the curve. Label the X and Y axis and make a dotted line grid at increments of 1 for X and Y. In addition, plot on your graph the data points X, Y as open circles for the following points:

X	Y
−1.8	3.5
−1.2	1.0
−0.5	−1.0
0.0	−1.3

0.5	0.3
1.2	0.5
2.0	3.5

The graphing program may have its own way of allowing you to enter this data or you could use the Turbo editor to make a file of the numbers.
Obtain a copy of the graph on the printer.

2.2 Addresses and data, RAM and ROM

Inside the IBM-PC there is an 8088 integrated circuit microprocessor which controls the operation of the computer. Connected to it are 16 primary address wires; eight of which also serve as data wires. There are also four secondary address wires. The 16 primary address wires allow the microprocessor to specify 65536 unique memory locations (addresses). During the data cycle of operation the eight data wires allow 256 unique numbers (or characters) to be represented. it is like a telephone system with 65536 telephone numbers and in which the caller can choose from 256 words to send a message. All the calls go through the central switchboard: the microprocessor.

The four secondary address wires allow addresses up to 16 × 65536 or 1 048 576. However, some special programming needs to be done in order to use those addresses. More about this later.

The address wires are connected to several different types of memory and to devices which allow communication between the computer and the outside world (for example the keyboard and screen). The microprocessor first places the binary representation of the location to be accessed on the address wires. Then after waiting for the computer circuits to select the unique location to which this refers, it either sends or receives a byte of data on the data wires during the data cycle. Along with performing some manipulation (eg, addition), this is all that a computer does.

Modern computers usually have several types of memory; early computers had only Random Access Memory (RAM). RAM is essential for any computer since the fundamental principles of computer operation require the Central Processing Unit (CPU) to repeatedly store and retrieve program instructions, data and memory addresses. The term 'Random Access Memory' means that it may be written to or read from in any order. A severe disadvantage of semiconductor RAM is that it doesn't remember anything after its power is turned off. Some computers have vital portions of their RAM protected by having a battery to provide the power in case of a power line failure.