LSI

OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

DIGITAL RESEARCH™

Please read the software license agreement before opening the diskette package. If you do not agree to the licensing contract, you may return the package to your distributor/dealer for refund as long as the diskette package remains unopened. Upon receipt of this registration card by Digital Research you will become a registered user and receive the following:

- Digital Research Newsletter
- Notices of updates and enhancements to Digital Research Software
- Digital Research Software performance reports and patches
- Discounts on updated versions of Digital Research Software

I have read the Digital Research Software Licensing Agreement and agree to abide by the terms contained in it:

Please Print

Your Name ________	Product ________
Company ________	Version Number ________
Street Address ________	Diskette
City ________	Serial Number ________
State/Zip ________	OEM/Dealer #
Country ________	or Name ________
Signature ________	Computer MFG ________
	Date ________

DIGITAL RESEARCH™

Please read the software license agreement before opening the diskette package. If you do not agree to the licensing contract, you may return the package to your distributor/dealer for refund as long as the diskette package remains unopened. Upon receipt of this registration card by Digital Research you will become a registered user and receive the following:

- Digital Research Newsletter
- Notices of updates and enhancements to Digital Research Software
- Digital Research Software performance reports and patches
- Discounts on updated versions of Digital Research Software

I have read the Digital Research Software Licensing Agreement and agree to abide by the terms contained in it:

Please Print

Your Name ________ Product ________
Company ________ Version Number ________
Street Address ________ Diskette
City ________ Serial Number ________
State/Zip ________ OEM/Dealer #
Country ________ or Name ________
Signature ________ Computer MFG ________
Date ________

FIRST CLASS PERMIT

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 180 PACIFIC GROVE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL RESEARCH™
Post Office Box 579, Pacific Grove, California 93950

FIRST CLASS PERMIT

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 180 PACIFIC GROVE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL RESEARCH™
Post Office Box 579, Pacific Grove, California 93950

DIGITAL RESEARCH LANGUAGES END USER LICENSE AGREEMENT

Use and possession of this software package is governed by the following terms.

1. DEFINITIONS - These definitions shall govern:

A. "DRI" means DIGITAL RESEARCH INC., P.O. Box 579, Pacific Grove, California 93950, the author and owner of the copyright on this SOFTWARE.

B. "CUSTOMER" means the individual purchaser and the company CUSTOMER works for, if the company paid for this SOFTWARE.

C. "COMPUTER" is the single microcomputer on which CUSTOMER uses this program. Multiple CPU systems may require supplementary licenses.

D. "SOFTWARE" is the set of computer programs in this package, regardless of the form in which CUSTOMER may subsequently use it, and regardless of any modification which CUSTOMER may make to it.

E. "LICENSE" means this Agreement and the rights and obligations which it creates under the United States Copyright Law and California laws.

F. "RUNTIME LIBRARY" is the set of copyrighted DRI language subroutines, provided with each language compiler, a portion of which must be linked to and become part of a Customer program for that program to run on the COMPUTER

2. LICENSE
DRI grants CUSTOMER the right to use this serialized copy of the SOFTWARE on a single COMPUTER at a single location so long as CUSTOMER complies with the terms of the LICENSE, and either destroys or returns the SOFTWARE when CUSTOMER no longer has this right. CUSTOMER may not transfer the program electronically from one computer to another over a network. DRI shall have the right to terminate this license if CUSTOMER violates any of its provisions. CUSTOMER owns the diskette(s) purchased, but under the Copyright Law DRI continues to own the SOFTWARE recorded on it and all copies of it. CUSTOMER agrees to make no more than five (5) copies of the SOFTWARE for backup purposes and to place a label on the outside of each backup diskette showing the serial number, program name, version number and the DRI copyright and trademark notices in the same form as the original copy. CUSTOMER agrees to pay for licenses for additional user copies of the SOFTWARE if CUSTOMER intends to or does use it on more than one COMPUTER. If the microcomputer on which CUSTOMER uses the SOFTWARE is a multi-user microcomputer system, then the license covers all users on that single system, without further license payments, only if the SOFTWARE is used only on that microcomputer. This is NOT a license to use the SOFTWARE on mainframes or emulators.

3. TRANSFER OR REPRODUCTION
CUSTOMER understands that unauthorized reproduction of copies of the SOFTWARE and/or unauthorized transfer of any copy may be a serious crime, as well as subjecting CUSTOMER to damages and attorney fees. CUSTOMER may not transfer any copy of the SOFTWARE to another person unless CUSTOMER transfers all copies, including the original, and advises DRI of the name and address of that person, who must sign a copy of the registration card, pay the then current transfer fee, and agree to the terms of this LICENSE in order to use the SOFTWARE. DRI will provide additional copies of the card and LICENSE upon request. DRI has the right to terminate the LICENSE, to trace serial numbers, and to take legal action if these conditions are violated.

4. COMPOSITE PROGRAMS

As an exception to Paragraph 3, CUSTOMER is granted the right to include portions of the DRI RUNTIME LIBRARY in CUSTOMER developed programs, called COMPOSITE PROGRAMS, and to use, distribute and license such COMPOSITE PROGRAMS to third parties without payment of any further license fee. CUSTOMER shall, however, include in such COMPOSITE PROGRAM, and on the exterior label of every diskette, a copyright notice in this form: "Portions of this program, ©1982 DIGITAL RESEARCH INC." In cases where such COMPOSITE PROGRAM is contained in READ-ONLY-MEMORY (ROM) chips, a copyright notice in the form listed above, must be displayed on the exterior of the chip and internally in the chip (in ASCII literal form). As an express condition to the use of the RUNTIME LIBRARY, CUSTOMER agrees to indemnify and hold DRI harmless from all claims by CUSTOMER and third parties arising out of the use of COMPOSITE PROGRAMS.

5. LIMITED WARRANTY

The only warranty DRI makes is that the diskette(s) on which the SOFTWARE is recorded will be replaced without charge, if DRI in good faith determines that the media was defective and not subject to misuse, and if returned to DRI or the dealer from whom it was purchased, with a copy of the original registration card, within ten days of purchase. Customer will receive support from the Vendor from whom customer has purchased the software. In addition, support is available from DRI directly, for qualified, registered customers under DRI's then current support policies. DRI reserves the right to change the specifications and operating characteristics of the SOFTWARE it produces, over a period of time, without notice.

6. DRI MAKES NO OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED, AND DRI SHALL NOT BE LIABLE FOR WARRANTIES OF FITNESS OF PURPOSE OR MERCHANTABILITY, NOR FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES SUCH AS LOSS OF PROFITS OR INABILITY TO USE THE SOFTWARE. SOME STATES MAY NOT ALLOW THIS DISCLAIMER SO THIS LANGUAGE MAY NOT APPLY TO CUSTOMER. IN SUCH CASE, OUR LIABILITY SHALL BE LIMITED TO REFUND OF THE DRI LIST PRICE. CUSTOMER MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE. CUSTOMER and DRI agree that this product is not intended as "Consumer Goods" under state or federal warranty laws.

7. MISCELLANEOUS

This is the only agreement between CUSTOMER and DRI and it cannot and shall not be modified by purchase orders, advertising or other representations of anyone, unless a written amendment has been signed by one of our company officers. When CUSTOMER opens the SOFTWARE package or uses the SOFTWARE, this act shall be considered as mutual agreement to the terms of this LICENSE. This LICENSE shall be governed by California law, except as to copyright matters which are covered by Federal laws, and is deemed entered into at Pacific Grove, Monterey County, CA by both parties.

DIGITAL RESEARCH™

by Gary A. Kildall
President

SAVE THIS LICENSE FOR FUTURE REFERENCE

DIGITAL RESEARCH
OPERATING SYSTEM END USER LICENSE AGREEMENT

Use and possession of this software package is governed by the following terms.

1. DEFINITIONS - These definitions shall govern:

A. "DRI" means DIGITAL RESEARCH INC., P.O. Box 579, Pacific Grove, California 93950, the author and owner of the copyright on this SOFTWARE.

B. "CUSTOMER" means the individual purchaser and the company CUSTOMER works for, if the company paid for this SOFTWARE.

C. "COMPUTER" is the single microcomputer on which CUSTOMER uses this program. Multiple CPU systems may require supplementary licenses.

D. "SOFTWARE" is the set of computer programs in this package, regardless of the form in which CUSTOMER may subsequently use it, and regardless of any modification which CUSTOMER may make to it.

E. "LICENSE" means this Agreement and the rights and obligations which it creates under the United States Copyright Law and California laws

2. LICENSE

DRI grants CUSTOMER the right to use this serialized copy of the SOFTWARE on a single COMPUTER at a single location so long as CUSTOMER complies with the terms of the LICENSE, and either destroys or returns the SOFTWARE when CUSTOMER no longer has this right. CUSTOMER may not transfer the program electronically from one computer to another over a network. DRI shall have the right to terminate this license if CUSTOMER violates any of its provisions. CUSTOMER owns the diskette(s) purchased, but under the Copyright Law DRI continues to own the SOFTWARE recorded on it and all copies of it. CUSTOMER agrees to make no more than five (5) copies of the SOFTWARE for backup purposes and to place a label on the outside of each backup diskette showing the serial number, program name, version number and the DRI copyright and trademark notices in the same form as the original copy. CUSTOMER agrees to pay for licenses for additional user copies of the SOFTWARE if CUSTOMER intends to or does use it on more than one COMPUTER. If the microcomputer on which CUSTOMER uses the SOFTWARE is a multi-user microcomputer system, then the license covers all users on that single system, without further license payments, only if the SOFTWARE was registered for that microcomputer. This is NOT a license to use the SOFTWARE on mainframes or emulators.

3. TRANSFER OR REPRODUCTION

CUSTOMER understands that unauthorized reproduction of copies of the SOFTWARE and/or unauthorized transfer of any copy may be a serious crime, as well as subjecting a CUSTOMER to damages and attorney fees. CUSTOMER may not transfer any copy of the SOFTWARE to another person unless CUSTOMER transfers all copies, including the original, and advises DRI of the name and address of that person, who must sign a copy of the registration card, pay the then current transfer fee, and agree to the terms of this LICENSE in order to use the SOFTWARE. DRI will provide additional copies of the card and LICENSE upon request. DRI has the right to terminate the LICENSE, to trace serial numbers, and to take legal action if these conditions are violated.

4. ADAPTATIONS AND MODIFICATIONS

CUSTOMER owns any adaptations or modifications which CUSTOMER may make to this SOFTWARE, but in the event the LICENSE is terminated CUSTOMER may not use any part of the SOFTWARE provided by DRI even if CUSTOMER has modified it. CUSTOMER agrees to take reasonable steps to protect our SOFTWARE from theft or use contrary to this LICENSE.

5. LIMITED WARRANTY

The only warranty DRI makes is that the diskette(s) on which the SOFTWARE is recorded will be replaced without charge, if DRI in good faith determines that the media was defective and not subject to misuse, and if returned to DRI or the dealer from whom it was purchased, with a copy of the original registration card, within ten days of purchase. Customer will receive support from the Vendor from whom customer has purchased the software. In addition, support is available from DRI directly, for qualified, registered customers under DRI's then current support policies. DRI reserves the right to change the specifications and operating characteristics of the SOFTWARE it produces, over a period of time, without notice.

6. DRI MAKES NO OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED, AND DRI SHALL NOT BE LIABLE FOR WARRANTIES OF FITNESS OF PURPOSE OR MERCHANTABILITY, NOR FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES SUCH AS LOSS OF PROFITS OR INABILITY TO USE THE SOFTWARE. SOME STATES MAY NOT ALLOW THIS DISCLAIMER SO THIS LANGUAGE MAY NOT APPLY TO CUSTOMER. IN SUCH CASE, OUR LIABILITY SHALL BE LIMITED TO REFUND OF THE DRI LIST PRICE. CUSTOMER MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE. CUSTOMER and DRI agree that this product is not intended as "Consumer Goods" under state or federal warranty laws.

7. MISCELLANEOUS

This is the only agreement between CUSTOMER and DRI and it cannot and shall not be modified by purchase orders, advertising or other representations of anyone, unless a written amendment has been signed by one of our company officers. When CUSTOMER opens the SOFTWARE package or uses the SOFTWARE, this act shall be considered as mutual agreement to the terms of this LICENSE. This LICENSE shall be governed by California law, except as to copyright matters which are covered by Federal laws, and is deemed entered into at Pacific Grove, Monterey County, CA by both parties.

DIGITAL RESEARCH™

by Gary A. Kildall
President

SAVE THIS LICENSE FOR FUTURE REFERENCE

LSI OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

LSI

OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

OCTOPUS SYSTEM GUIDE

This manual describes the operation of and available features on the Octopus range of micro computers. For additional information or technical assistance contact LSI Computers at the address below.

The Octopus is designed and produced in the UK by:

LSI Computers Limited, St. Johns, Woking, Surrey.
Telephone: 04862 23411

INTRODUCTION

CONTENTS

INTRODUCTION

The Octopus computer system was developed specifically for the business user and incorporates many advanced features and facilities. These advanced facilities are designed to make the system easy to use and easy to apply to many tasks within a business environment.

This documentation is designed to take you through the first few steps of setting up your system, and provide a technical and general reference guide to the hardware and software facilities provided.

The Introduction provides a basic explanation of the principal elements within the system. It will take you through the procedure necessary to set up the system ready for use. Follow through this procedure, step by step. It will provide an initial familiarity with the system, and cover a number of points which will check that your supplier has set up the system correctly for your use.

The reference sections on, the Hardware, the Operating System and the BASIC Primer are designed to add to your understanding and technique as your confidence and competence grows. They also provide further information should you have any difficulty with the basic procedures covered in the introduction.

The final sections contain details of various hardware expansion options, software availability and technical specifications. For example it contains details of the application software which is available for use on the system, and of various standard add-on facilities, all of which can be obtained from your supplier.

YOUR SYSTEM EXPLAINED

2.1 The Processor Unit

The Processor Unit houses the processors, the disk drives, and the electronic systems which control the transfer of information and data between the various elements within the system.

The Main Logic Board provides an advanced dual processor capability. The system has a 16 Bit Intel 8088 processor, and an 8 Bit Zilog Z80B processor. This means that you can use application software which is produced in either 16-Bit or 8-Bit code. The system decides which processor it will use as the software is loaded. You are not required to provide any additional commands.

The programs and data used and produced by the system are stored on either floppy diskettes, or a Winchester hard disk drive.

Your system will have one of three disk drive arrangements;

MODEL 1	Single Floppy Disk Drive
MODEL 2	Two Floppy Disk Drives
MODEL 3	One Floppy Disk Drive and a Winchester Disk Drive.

You will find, later in this Introduction, separate instructions for organising your system depending upon the disk drive arrangment.

2.2 The Monitor

The Monitor provides the visual reference for the system. It has a green phosphor screen and is normally mounted on top of the Processor Unit. It can however be mounted beside the unit if that is more convenient.

The Processor unit also has a TV Jackpoint. This means that using a UHF modulator the system can also be operated in conjunction with a conventional television receiver. Your Business Computer can therefore double very effectivly as a Personal (or Home) Computer - and a very powerful personal computer at that.

The actual character formation on the screen is controlled by programmable 'fonts'. A standard font is supplied with your system. The procedure for generating your own fonts is explained in the Operating System section under System Utilities.

2.3 The Keyboard

A number of standard keyboards can be used with your system. Check the layout and specific functions of your keyboard against the diagrams and descriptions in the Hardware section of this manual. Generally the key functions are the same for all keyboards but their arrangement within the keyboard layout differ slightly. In this introductory session you will only use the keys which are common to all keyboards.

The keyboards have three types of keys. The main keyboard has a conventional set of alpha-numeric keys just like those of an ordinary typewriter keyboard. This is called the QWERTY set. Look at the first six characters of the top row and you will see why it is called QWERTY.

It also has one additional key called the CONTROL (or CNTRL) key. The significance of this key will become apparent when you follow the first instructions.

There is a numeric key pad for fast entry of numeric data, your accounting information for example. There is also a groups of keys called 'function' keys. They have an F and a number on them, F1, F2 etc. These keys can be 'programmed' to perform specific functions which normally require more than one key stroke. The procedure for programming these function keys is explained in the Operating System section of this manual.

2.4 The Printer

Your systém will also have a printer. There is a wide variety of printer types which your supplier can provide.

Your supplier will normally set up the system to suit the printer supplied.

We will describe later how to test your printer and how to change the means by which the system and the printer communicate.

2.5 The Software

Your Computer System is supplied with the **CP/M 86-80 PLUS Operating System**, and the **PBASIC Interpreter** on a single diskette. If you have at least two disk drives with your system, you will also have been supplied with the **AXIS Accounting System**. The latter is supplied on two additional diskettes making three in all.

You will need to complete the Registration Forms for these software programs and return them as instructed on the forms. You should do that now before you start.

The Operating System is a very special program. It provides a range of management facilities for your system; it is the system housekeeper. It controls the disk operations, and transfer of instructions and data between the other elements of the system. It is the Operating System which provides the facility to run a wide variety of software from different sources.

The Operating System supplied with your computer is the very latest version of CP/M, the industry standard operating system for microcomputers.

This new version called **CP/M 86-80 PLUS** still afford's access to the vast range of micro-computer application software designed to operate under the original versions of CP/M. You can add all types of programs at a later date; word processing, financial planning systems, data base systems, and some very specialised systems for professional use.

There are also a number of other operating system programs called **System Utilities** some of which are standard CP/M utility programs whilst others are provided specifically for use with the equipment described in this manual. You will use some of these utilities to set up your working diskettes in this introduction.

The **PBASIC Interpreter** has three uses. First is will enable you to create your own programs, and provide you with first class calculation facilities either directly from the keyboard, or via simple programs for repetitive work. Secondly it will enable you to write, or have written for you, your own application programs. Thirdly, since PBASIC is compatible with Microsoft's MBASIC, you will be able to run any other application programs which require an MBASIC Interpreter to be present for the programs to function.

SETTING UP YOUR SYSTEM

3.1 Power Supply and Connections

Your Computer System can be operated from a conventional 240 volt 13 amp fused power supply. Special power arrangements are **NOT** normally required.

If the smoothness of your power supply is doubtful, for example you experience voltage fluctuations, there are regulation devices which can be included in the circuit which will overcome any operating problems. Your supplier will be pleased to provide further details and advice if required.

In this introduction we will assume that your supplier has set up the connections between the Keyboard, the Monitor and the Processor Unit. If this has not been done then consult the Hardware section of this manual for details of the three simple connections which have to be made using the special connectc:s and cables provided.

The printer can be connected to one of the large sockets on the rear of the Processor Unit. These are marked PARALLEL and RS232. The PARALLEL connector is called the **parallel port** and the RS232 the **serial port**. Many printers can be set for either Parallel or Serial operation. Where you have a choice the parallel port is recommended.

Your supplier should have matched the settings within the system, to the printer. If the printer does not respond to the procedure detailed later in this introduction, first contact your supplier to check whether it is set up for 'parallel' or 'serial' operation, and that your operating system has been set accordingly. Also check that the printer is connected to the appropriate 'port'.

With all the connections made as above, the system can be powered on:

The Monitor is switched on independently and should be switched on first using the top **On/Off** knob. A small red indicator lamp on the top right hand side of the unit will glow.

There are two adjustments on the Monitor - Contrast and Brightness. These are controlled via the bottom two knurled knobs on the right hand side of the monitor facia. If after switching on the screen is blank, turn the these knobs clockwise to achieve the required result.

The switch for the Processor Unit is at the rear.

3.2 Data Storage Media

Two types of data storage medium are provided with the system - Floppy Diskettes and Winchester Hard Disks. All models have at least one floppy disk drive. Your operating system programs, and other software will be supplied on floppy diskette.

If you have a Winchester system, you will still need to understand the function, use and preparation of floppy diskettes.

3.3 Care and Use of Floppy Diskettes

Handled with care, floppy diskettes are remarkably durable. But they must be treated with respect. A floppy diskette is a sheet of plastic film contained in a soft lined hard case. The information which your system uses and produces is stored on the surface of the diskette.

All diskettes have a centre boss. This fits onto a cone inside the disk drive which rotates the disk at high speed whenever the system needs to access the information on the diskette, or transfer newly processed information back to the diskette. You can see when the diskette is being 'accessed'. The red indicator light on the drive will glow. Do not remove diskettes from the drive when this indicator is glowing.

The surface of the disk is 'read', through a 'window' cut in the hard case, by a head inside the disk drive. The 'window' makes the diskette very vulnerable to fingers when the diskette is being handled. Always therefore:

- **Hold a diskette lightly by one of the top corners**
- **Do not exert pressure**
- **Do not write on a label which is already fixed to the diskette**

The information stored on the surface is also sensitive to random electrical impulses. Always therefore:

Keep floppy diskettes well away from telephone equipment, photocopiers, and any other device which has either a discharge lamp, (this includes your display monitor) or a powerful electric motor, for example cleaners and polishers used by the office cleaners.

Floppy diskettes are sensitive to heat, dust and liquids:

Do not smoke in the vicinity of floppy diskettes
Do not subject them to heat from radiators, electric fires or sunny window ledges
Do not drink coffee, tea or any other liquids whilst handling the diskettes

Treated in a sensible and careful way your floppy diskettes will cause few problems. **Maltreat them at your peril.**

3.4 Preparing New Floppy Diskettes

You can use most proprietary brands of good quality floppy diskette.

There are two procedures which you must follow before you can use a diskette in the system.

All floppy diskettes must be **formatted** by the system. We will take you through the process of **formatting** a diskette in the final section of this Introduction.

3.5 Use of Winchester Hard Disks

A Winchester disk is a permanently sealed unit. Once set up and operating it should require no attention. In the event that you do experience operating problems, it is imperative that the use of the Winchester disk system is supported by a regular back up operation. This transfers the contents of the Winchester onto floppy diskettes. Ideally this should be done daily.

The operating system utility programs include a BACKUP program which will transfer the data contained on the Winchester disk onto floppy diskettes. (See Operating System section).

3.6 Preparing a Winchester Disk System

Under normal circumstances, your Winchester disk will have been prepared for use by your supplier. There are two stages to this process.

First the disk is **formatted** in a similar way to a floppy diskette but using a different utility program. This procedure is normally undertaken by your supplier. You should check with your supplier whether or not the Winchester has been formatted.

If it has not then refer to the WDFORMAT utility program described in the Operating System section of this manual.

3.7 The Operating System

Your CP/M Operating System is the system's housekeeper. In addition to coordinating the hardware and software elements within the system its principal function is to organise and manage all the **files** used within the system.

An appreciation of the filing system is a pre-requisite for effective use of the system.

All the programs and data stored by the system are organised within a filing system. It is the operating system which manages the organisation and maintenance of the filing system.

There are basically two principal types of file which you must learn to recognise:

1. **Program files** - these contain the coded instructions which tell the computer to perform specific operations. The operating system itself is a series of program files.

2. **Data files** - these files contain the data entered and stored within the system.

3.8 File Recognition

Each file has a name, or label, so that it can be recognised by you and the computer. These filenames are **symbolic** names. That is, they are assigned file names which are repesentative of the contents. Program files, and files created within say an accounting program, are all assigned names by the manufacturers. If you intend using your system for word processing, or financial planning, or for writing your own programs you will need to understand and appreciate how files names are created. You will find further information on File Naming in the Operating System section.

Basically all you need to know at this stage is that file names have an eight character descriptive **filename** and a three character optional **filetype** which indicates the type of file. The **filename** and the **filetype** are separated by a full stop.

The **filename** can be numbers (0-9) or characters (A-Z). Some punctuation marks cannot be used in either the name or the type. (See the Operating System section).

Some **filetypes** have a special meaning to your system. Filenames which end with **.CMD** for example are program files in 16 Bit code. The file called BASIC.CMD is the PBASIC program. A program written in 8 Bit code will have a **.COM** filetype. For example a spread sheet program called SuperCalc has a file name called SC.COM.

You can use either upper or lower case characters for filenames; the operating system converts all lower case characters into upper case characters automatically.

The different types of file are principally recognised by the **filetype**. There are certain conventions which are used which apply designated types to particular files. (See the Operating System section for further details).

You will learn how to identify which files are on which diskettes in the final section of this introduction when you start to create your own working programs.

YOUR FIRST STEPS

We will demonstrate the essential management functions of your system by taking you through the steps required **before** you put your system into use. FOLLOW THESE STEPS THROUGH ON THE SYSTEM.
To complete this section you will need at least two blank floppy diskettes. If you have the AXIS Accounting System, you will also need at a further four blank diskettes.

If you have Winchester system, we will assume that the Winchester disk has been **formatted** by your supplier. If it has not, you will need to use the FDFORMAT utility program. Details of the procedure are included in the Operating System section. Read the instructions carefully. If you do not understand them consult your supplier.

4.1 Booting the system

To complete this procedure, you will 'boot' your system, and check the contents of each of the diskettes supplied before creating working copies of the software for everyday use.

If you have a single or twin floppy disk system, as you switch on the screen will display:

```
Testing ...
             Main Processor
             PROM
             DMA Controllers
             RAM
             Interrupt Controllers
             Floppy Discs

*** Self Test Complete ***

Firmware versions:

SYSTEM    XX

Insert System Disc
```

This self test procedure is performed each time the system is switched on.

If you have a floppy disk drive system, the final message reads:

```
Insert System Disc
```

Load the diskette marked **CP/M 86-80 PLUS** into the left hand disk drive. The label must be facing upwards, and the drive must be closed using the small lever provided.

If you have a Winchester disk system the self test procedure will also check the Winchester disk. The final message reads:

```
Winchester=W  Floppy=F
```

Until the Winchester has been set up, you will key F, to use the floppy disk drive.

Load the diskette marked **CP/M 86-80 PLUS** into the left hand disk drive, (the label must be facing upwards, and the drive must be closed with the small lever provided), and enter <F> to tell the system that it must refer to the floppy disk drive for its next instruction.

For both types of system a sequence similar to the following will be executed and displayed:

```
CP/M86/80 Plus Level 2 (Date)
LSI Computers Ltd., Woking
256K Octopus

Available drives :-
A: L.H. drive, 5" double sided, double density
B: R.H. drive, 5" double sided, double density
M: 64K Memory Disk

A>submit autoexec

A>loadkey ws5
Key table loaded
A>loadfont chrgen
Font loaded
A>
```

The A> is called the system prompt. It tells you the operating system is in control. If you have a hard disk system, the prompt is C>.

This procedure of starting the system from cold is called a **cold boot**. What has happened is that the operating system has been loaded from the diskette into the computers memory.

The A, in the system prompt, tells you that you are **logged** onto Drive A:. In a single or twin floppy diskette system, the left hand drive is called Drive A and the A> means that the system is ready to **read** files from the diskette in Drive A:. In this case Drive A: is called the **logged** drive.

If you have a Winchester disk system the left hand floppy drive is called Drive C:, and the Winchester is split into two logical drives – Drive A: and Drive B:

Note the colon following the drive letter. This is the standard CP/M convention for designating a drive.

4.2 Displaying a Directory

You can check which files and programs are contained on a disk or diskette using an operating system function called DIR. DIR stands for DIRectory and is the name of a utility program DIR.CMD.

To execute the utility program DIR.CMD and display the directory of the floppy diskette enter the following on your system:

```
A>DIR<CR>
```

DIR is called a **command** and is entered at the A> (or C>). It can be DIR or dir. The <CR> means press the key marked   i.e. the carriage **return** or **enter** key. Throughout the system this <CR> key is used to confirm that a keyboard entry is complete.

If you have entered the DIR command and pressed carriage return the screen will display all the files/programs contained on the disk in drive A: (or Drive C:). A typical directory display is shown on the next page.

```
A>DIR
A: CHRGEN   FNT  : CPM3     SYS  : AUTOEXEC SUB : SUBMIT   CMD
A: SHOW     CMD  : BASIC    CMD  : LOADKEY  CMD : LOADFONT CMD
A: DISCOPY  CMD  : BACKUP   CMD  : DATE     CMD : PARMGEN  CMD
A: FONTGEN  CMD  : WS5      KEY  : FORMAT   CMD : LOADPARM CMD

A>
```

You can also print, or produce a **hard copy** of the contents of the directory by entering the following command line;

```
    A>DIR<↑P><CR>
```

The < ↑ P> means hold down the key marked CONTROL (or CNTRL), depress the key P, and then release the CONTROL key. As you press <CR> the same directory produced on screen will be output to the printer. This will enable you to keep a physical record of the disk contents.

Performing this function will check that your printer has been set up correctly. If it does not respond to this command, first check that it is switched on and loaded with paper. If it still fails to respond you will need to check with your supplier whether the system has been set up for the particular printer in use.

Remember that you need to check whether your printer has been set for serial or parallel operation, and whether it has been **assigned** to the appropriate output port of the computer on the back of the processor unit. (See Operating System section, CP/M Transient Programs - DEVICE and System Utilities - PARMGEN).

If you have the AXIS Accounting System, or any other programs on diskette, repeat the above procedure with the other diskettes loaded in the floppy disk drive.

4.3 Formatting a new diskette

DO NOT USE THE SUPPLIED MASTER DISKETTES AS WORKING COPIES. Replacements will take several days to arrive and are chargeable.

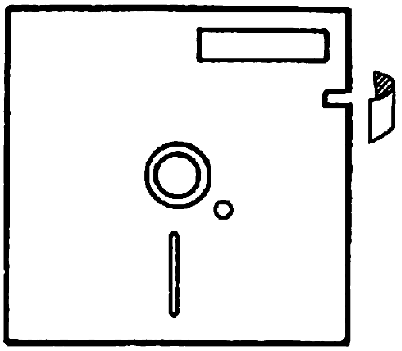
There are two steps involved in copying the supplied **Master disks** to create working copies for everyday use. First we will FORMAT a diskette, then we will use a program called DISCOPY. This will produce a perfect copy of the contents of the master disk (or any other disk) onto a blank **formatted** disk.

Formatting is a process in which the the surface of a diskette is 'mapped' out by the system so that it can maintain a record of the location on the disk of the contents of any file. There are different procedures for the Winchester disk and for floppy disks. We will deal with floppy disks first since this is procedure is required for all systems.

All new diskettes must be formatted before use. The procedure is quite straight forward and quite automatic.

Note also that FORMAT can be used to create blank diskettes from previously used diskettes, but the contents of the used diskette will be destroyed irrevocably. You must therefore be careful that your **drive to be formatted** specification is correct. To cover the possibility of error you should first **protect your master diskette**.

You will find that there are small metallic labels supplied with new diskettes. Place one of these labels over the side notch of the master diskette as in the diagram. This is called **write protection**. It will allow the system to **read** the contents of the diskette, but prevent any attempt to **write** onto the diskette.



For twin floppy disk systems, load the diskette marked **CP/M 86-80 PLUS**, with the notch covered as indicated, into the left hand disk drive and insert a new blank diskette into the right hand drive. The right hand drive is called Drive B:.

Enter the following command:

```
A>FORMAT B:<CR>
```

FORMAT.CMD is a program which formats a floppy disk. Note the space after the 'FORMAT' command (you don't need to add the '.CMD' extension). The B: means **the diskette in Drive B**. The process is quite automatic, just respond to the display as requested.

For a single floppy disk system or a Winchester disk system the procedure is slightly different. The Master disk **must be removed** from the drive **before** the formatting commences.

For a single floppy disk system, the command line is:

```
A>FORMAT A:<CR>
```

For a Winchester disk system the command line is:

```
A>FORMAT C:<CR>
```

In the last two cases you must remove the Master disk from the floppy drive and replace it with the new disk. The screen display will tell you when to change disks. When the process is completed, the screen display will again tell you when to replace the Master disk.

The formatting procedure also checks the surface of the disk being formatted and will report any problems. The FORMAT program allows you to format additional disks, if required, before returning control to the operating system - just follow the instructions on screen.
From this point on the procedures for floppy, and Winchester disk systems are quite different. If you have a Winchester disk system then continue at 4.6 below.

4.4 Creating Work Diskettes

For a twin floppy disk system leave the diskettes as they are; the **CP/M 86-80 PLUS** diskette in Drive A:, and a newly formatted diskette in Drive B:. Enter the command:

```
A>DISCOPY<CR>
```

This command invokes a program called DISCOPY.CMD and will copy the contents of the diskette in Drive A: onto the diskette in Drive B:. Again the procedure is fully automatic.

You can now use the copy of the Master disk as a working program disk, and store the Master disk itself in some safe place together with the printed directories produced above.

Repeat this step to create working copies of your other Master disks. Format a new diskette for each Master disk to be copied and use DISCOPY to copy over the contents. Label each diskette clearly using the diskette labels supplied.

For a single floppy disk system ensure that the **CP/M 86-80 PLUS** diskette is in Drive A:. Invoke the DISCOPY command as for twin disk systems and follow the instructions displayed. You will be asked to swop the Master disk and the formatted disk several times so that the contents can be copied from one disk to the other via the system's main memory.

4.5 Selective File Copying

There are two other Operating System programs which you must learn to use. These are PIP and SHOW. PIP stands for Peripheral Interchange Program, SHOW is a disk status program.

PIP is primarily used to transfer individual files between disk drives (or diskettes). SHOW is used to check the available space on a particular disk drive (or diskette). Their other functions are covered in the Operating System section.

The working copy of the **CP/M 86-80 PLUS** diskette contains both the Operating System programs and PBASIC. In practice you will only need PBASIC on other diskettes either to run programs written in BASIC, or to write your own programs.

Similarly disks to run your application programs will not require all the files present on the **CP/M 86-80 PLUS** diskette.

YOUR FIRST STEPS

The following procedure will create some handy disks for daily use. First ensure that the working copy of the **CP/M 86-80 PLUS** diskette is in Drive A:.
Take a directory using DIR as below:

```
A>DIR<CR>
```

The system will present the directory on screen.

You can find out how much space is available on the disk for new program or data files by using the program SHOW. You enter a SHOW command as follows:

```
A>SHOW<CR>
```

The response to this command will be in the form:

```
A: RW, Space:        38k

A>
```

The available space on Drive A: is only 38 KBytes, that is 38,000 characters of program or data. That does not leave much space for regular usage, so we will transfer PBASIC to a new diskette.

FORMAT a blank diskette just as you have earlier.

Leave the diskette in drive B:. Enter the following command:

```
A>PIP B:=A:BASIC.CMD<CR>
```

This command line says PIP, i.e. transfer, to drive B:, a program on Drive A: called BASIC.CMD. The <CR> says that the command line is complete. The command must be precise. There must be a space after the PIP command, there must be a colon after the drive specifications, i.e. A: and B:. There must be no other spaces in the command line except that after the PIP.

If you do make a mistake the system will present an **error message**. Check the error message and then re-enter the command.

There is another way to use PIP which we will demonstrate. If you have more than one file to transfer, the commands can be entered in turn without reloading the PIP program each time. Your PBASIC disk will benefit from having the programs PIP, SHOW, FORMAT and DISCOPY available for future use. Enter the next command sequence:

```
A>PIP<CR>
*B:=A:PIP.CMD<CR>
*B:=A:SHOW.CMD<CR>
*B:=A:FORMAT.CMD<CR>
*B:=A:DISCOPY.CMD<CR>
*<CR>
A>
```

As PIP is entered and confirmed, the first * is presented ready for a command instruction. When that command has been successfully completed, the second * is presented ready for the next instruction and so on. The <CR> command at the final * says that the sequence is complete.

You will now have a PBASIC working diskette with five files present, BASIC.CMD, PIP.CMD, SHOW.CMD. FORMAT.CMD and DISCOPY.CMD. This you can now use for development purposes.

You cannot however **cold boot** the system from the new PBASIC diskette you have created. Before a diskette can be used to boot the system from a cold start, it needs to contain two functions on disk. The first is what is called the **loader track**. The loader track is transferred during the format process. It is therefore already on your diskette. The second is the CP/M 86-80 PLUS Operating System programs. These are held in a file called CPM3.SYS.

You must therefore transfer this file to the new PBASIC diskette using PIP as follows:

```
A>PIP B:=A:CPM3.SYS<CR>.
```

The procedure which you have followed to create the PBASIC diskette is one which you will use to create any other application program disks for use on your system.

4.6 Setting up a Winchester Disk System

There are two other Operating System programs which you must learn to use. These are PIP and SHOW. PIP stands for Peripheral Interface Program, SHOW is a disk status program.

PIP is primarily used to transfer individual files between disk drives (or diskettes). SHOW is used to check the available space on a particular disk drive (or diskette). Their other functions are covered in the Operating System section.

To start this procedure load the working copy of the **CP/M 86-80 PLUS** diskette into Drive C:. the floppy diskette drive.

Take a directory using DIR as below:

```
A>DIR <CR>
```

The system will present the directory on screen.

You can find out how much space is available on the disk for new program or data files by using the program SHOW. You enter a SHOW command as follows:

```
A>SHOW<CR>
```

The response to this command will be in the form:

```
A: RW, Space:        38k

A>
```

The available space on Drive A: is only 38 KBytes, that is 38,000 characters of program or data. However there is much more space available on the Winchester disk.

We will now use the program PIP to transfer the contents of the **CP/M 86-80 PLUS** diskette to the Winchester disk. There is a facility in CP/M called **file matching**. This is covered in detail in the Operating System section of this documentation. The most commonly used aspect of file matching is the use of the '*' to represent any character, or number of characters.

With the **CP/M 86-80 PLUS** diskette loaded in the floppy diskette drive enter the following command line:

```
C>PIP A:=C:*.*<CR>
```

This command line says PIP, i.e. transfer, **all** the files on drive C: onto Drive A:. The *.* means files of all filenames and all filetypes. The command must be precise. There must be a space after the PIP command, there must be a colon after the drive specifications, i.e. A: and C:. There must be no other spaces in the command line except that after the PIP.

As each file is transfered its name will be displayed on the screen.

If you do make a mistake the system will present an **error message**. The error messages for PIP are included in the Operating System section. Check the error message and then re-enter the command.

You may remember that when you first started up your system, you were presented with an option which read:

```
Winchester=W Floppy=F
```

Before you can use the Winchester disk to boot the system from a cold start, two functions need to be present on drive A: of the Winchester. The first is what is called the **loader track**. The second is the CP/M 86-80 Operating System programs. These are held in a file called CPM3.SYS.

The CPM3.SYS will already be present on the Winchester. It will have been transferred during the PIP operation you have just completed. The **loader track** is added using the program called FORMAT.CMD. This is the same program used to format floppy diskettes.

In the case of the Winchester disk we use the format program in a special way. You will have the C> on the screen. So enter the following command:

```
C>FORMAT A:/T<CR>
```

The /T says 'transfer the loader track only - do not format'.

Remove your **CP/M 86-80 PLUS** diskette and file it in a safe place.

You can now check that the system will start up from the Winchester. You can do this in one of three ways. Switch off the power for a few seconds – then switch it on again, press the RESET button on the back of the machine, or press down the CONTROL SHIFT and DELete keys simultaneously. This will automatically purge any programs in the computers memory and restart the system.

You can now ask the system to boot from the Winchester by selecting W for Winchester at the end of the self test display. If you have set up the system correctly, then the A> will be returned to the screen.

You can now load the two AXIS Master Program diskettes into the floppy drive in turn and PIP the contents over onto drive A: of the Winchester using the following command line:

```
A>PIP A:=C:*.*<CR>
```

Your system is now set up for use.

4.7 Further Familiarisation

This introduction has used the commands which you will use most frequently in the day to day use of your system.

There are however other facilities which will improve its overall utility. They are covered in the Operating System section. If you have a Winchester disk system you must read and understand the importance of the utility program BACKUP. This is the facility you will use to copy your data and programs from the Winchester onto floppy disks for security.

You should also read the Expansion Options section. This includes general details on various other application programs available for use on your system, and information on the expansion facilities which are available and which you will find increasingly important as your system usage increases.

LSI

OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

THE HARDWARE

CONTENTS

THE PROCESSOR UNIT

This is the central part of the computer. As well as the disk drives, it holds the processors which do the work and the electronics which control the flow of information to and from the various parts of the system - the screen, the keyboard, the disks and the printer.

To you, who will be using the computer, this unit will appear as a large rectangular box, and you will seldom, if ever, need to know what goes on inside it.

The box may sit on top of your desk, or you may prefer to save desk space by mounting it on its plinth by the side of your desk. See Expansion Options for further details on the vertical mounting kit.

1.1 The Processors

Your computer system actually has two central processors. One is an Intel 8088, one of the new generation of 16-bit processors for which fast and powerful applications programs can be written. The other is the popular and well-established Zilog Z80 8-bit processor, which lets you make full use of the wide range of established software designed to run under the CP/M operating system.

1.2 The Disk Drives

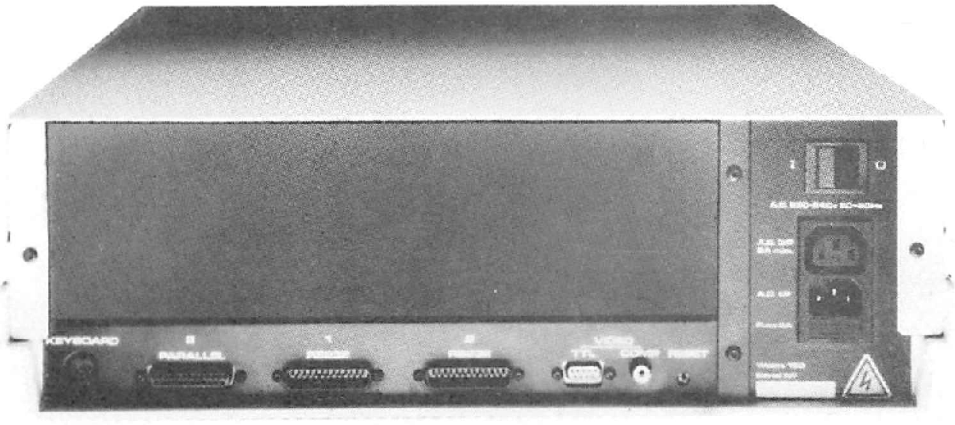
In the front of the unit you will see the diskette drives. These are the slots which will hold the disks when you are using the computer. Your system will have one of three arrangements of disk drives:

MODEL 1	Single Floppy Disk Drive
MODEL 2	Two Floppy Disk Drives
MODEL 3	One Floppy Disk Drive and a Winchester Disk Drive.

If your system has a Winchester drive, then you will not need to change it, and the mechanics need not detain us here. You will need to know something about the Diskette drives though, as you will often need to put disks in and take them out again.

Take a look at the front of the floppy disk drive. There is a horizontal slot, and a lever which turns through 90 degrees to point either to the right or downwards. Before you put a disk in the drive, you should turn the lever to point to the right, thus opening the drive. Remove the diskette from its protective cover and slide it into the drive with the large oval hole pointing into the machine, and the label uppermost. You should feel the disk seat itself positively. Then close the drive by turning the lever downwards through 90 degrees.

THE PROCESSOR UNIT



1.3 The Power Supply

At the back of the box you will see a variety of plugs and switches. Look at the diagram opposite and compare it with your own system.

The main on/off switch is on the right hand side, at the top. Underneath that are two sockets, one above the other. The lower one of the two takes the mains lead, the higher is an auxiliary mains output socket (for the monitor). Below these two is the fuse.

1.4 Connections for the System Components

Still at the back, and looking along the bottom from the left, there is a row of different shaped sockets. The first is a 5 pin DIN connector, into which you will plug the keyboard. Next are three sockets which are connectors for printers and other extra units. Your supplier will tell you which socket will take your particular type of printer.

For the technically minded these are a bidirectional parallel port and two asynchronous RS232 serial ports.

Next come the video connectors; one marked TTL which you will use if you have a colour monitor, the second is a phono socket for a monochrome monitor.

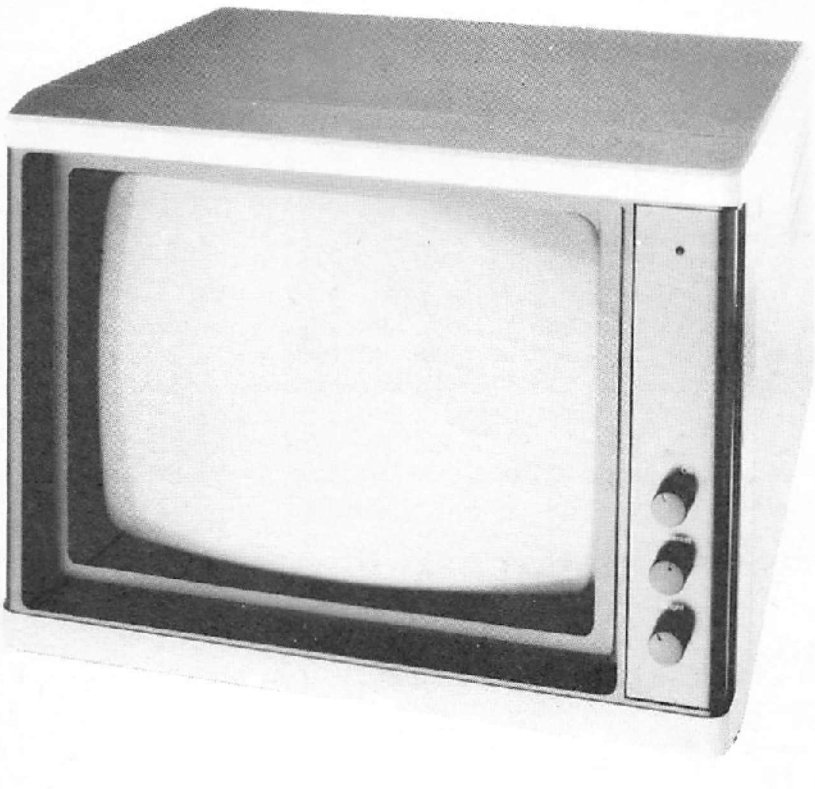
Finally there is the red **RESET** button. If you press this button while the machine is working, it will start the computer all over again, as if you had just switched on. By doing this, you will lose everything that was in the computer's memory before you pressed it, so you should only use the **RESET** button as a last resort.

1.5 Room for Expansion

Inside the box, room has been provided for extra circuit boards, which will allow your system to expand as your requirements for more computer power increase. Options for expansion include additional memory, a communications board to permit direct transfer of data between computers, a graphics board, a connection to the telephone line, and a network controller.

The Expansion Options section contains further details. Your supplier will also have additional information on the latest facilities.

THE MONITOR



THE MONITOR

The monitor which comes as standard with the system has a 12" screen and normally displays green characters on a black background. Together with the keyboard, this is your main way of communicating with the computer.

2.1 Connecting the Monitor

Packed with the monitor you should find a cable with a single-pin (phono) plug at either end. One end of this goes into the socket marked **IN** at the back of the monitor: the other goes to the phono socket at the rear of the disk drive unit.

The power cable may have a standard three pin mains plug or it may have a plug designed to fit into the auxiliary mains socket on the disk drive unit. If you use the auxiliary mains socket you will be able to leave the monitor's on/off switch set to **ON** all the time, and turn the whole system on and off by using the single mains switch on the rear of the disk drive unit.

2.2 Colour

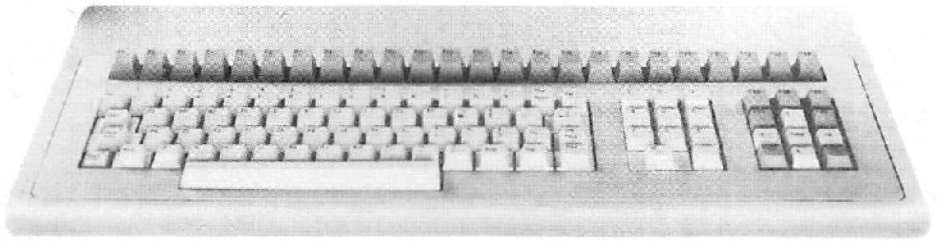
If you want to take advantage of the system's colour display facilities, you will need a colour monitor. In that case you will find that instead of a single pin phono plug at one end of the cable, there is a D-shaped plug. This will fit into the socket marked TTL to the left of the phono socket on the rear panel of the processor unit.

2.3 Screen Attributes

In use, the screen displays 25 lines of 80 characters. Of the 25 lines, one is reserved for special messages. Normally the screen shows green characters on a black background, but this may be changed by the software you are using. The foreground characters may be bright or dim, and it is possible to show black letters on a green background.

2.4 Character Fonts

The form of the characters on the screen is controlled by fonts. **Font** is a printer's term for the stock of letters and characters in a particular typestyle. Your system can hold a number of fonts of different typestyles. The system is supplied with a standard font, but it is possible to create new fonts using the system utility called FONTGEN. Each font consists of 256 characters which, for convenience, is split into two equal parts of 128 characters each. These are referred to as **font 1** and **font 2**. To find out more about FONTGEN, you should refer to the section on the Operating System.



THE KEYBOARD

There are currently two types of keyboard available with the system. One is the standard LSI keyboard, the other is similar to that on the IBM Personal Computer. Both keyboards have three types of keys - a block of QWERTY keys, as on a normal typewriter; a numeric keypad for fast entry of data; and a number of special keys called **function keys**. Compare your keyboard with the diagrams opposite as you read through this section.

3.1 Connecting the Keyboard

Both keyboards have a coiled cable ending in a five-pin DIN plug, which goes into the DIN socket at the rear of the processor unit. Note that if you are reaching behind the unit to plug it in 'blind', then the socket will be on the right hand side as you look at it, and the groove on the plug goes at the top. Your system is able to tell which kind of keyboard is plugged in, and even if there is none there at all.

3.2 The QWERTY Cluster

This section of the keyboard is very similar to a standard typewriter, except that there are a few extra keys which need some explanation. These are:

The **CONTROL** or **CTRL** key

This acts in a similar way to the SHIFT key. If you hold the CONTROL key down and type any of the 26 alphabetic keys, or some of the special character keys, a separate code will be produced. These do not normally print on the screen, but they can be given a special meaning by the software you are using.

The **ALT** key (only on the IBM keyboard)

The IBM keyboard has this special key which works in the same way as CONTROL and produces yet another set of codes. These may sometimes be used by applications software designed for use on the IBM Personal Computer.

The **CAPS LOCK** key

Press this down once and an amber light in the key comes on: press it again and the light goes out. When it is on, the alphabet keys will give only UPPER CASE letters. It has no effect on the keys with two symbols marked on them; you will still have to press SHIFT to get the upper character.

The **SHIFT LOCK** key (Only on the LSI keyboard)

This works in the same way as CAPS LOCK, but as well as upper case letters on the alphabet keys, it produces the upper character on the other keys.

The **DEL** key (delete)

Use this key to delete the character which comes immediately before the cursor on the screen.

The **ESC** key (for **escape**)

This key always gives one particular code. This key is used by the AXIS accounting system. It may also be used for special purposes by other software packages.

3.3 The Numeric Keypad

As well as having numbers in the usual place in the QWERTY cluster, they are also arranged in a special block to the right, together with a decimal point key, to enable fast and efficient entry of numeric data. With the standard LSI keyboard, this is as far as it goes.

The Numeric Keypad on the IBM-type Keyboard - If you have the IBM type keyboard there are some extra keys in the numeric keypad. These are NUM LOCK, SCROLL LOCK, '+' and '-'. NUM LOCK and SCROLL LOCK deserve some extra comments.

NUM LOCK (Numeric lock)

This key is an alternative action key like CAPS LOCK. When it is down, and its amber light is glowing, the keys produce numbers when pressed. When the key is up, the keys in the keypad become special function keys, as described in the next section.

SCROLL LOCK

The SCROLL LOCK key is a special function key sometimes used by applications software.

3.4 Special Function Keys

Both keyboards have a number of special keys which can be programmed to produce any character or sequence of characters you need. This will often be handled by the particular software you are using, but if you want to program keys to give codes or words you use a lot, you can do this with the utility program called KEYGEN. You will find the details of how to use KEYGEN in the section relating to the Operating System.

The way in which the Special Function Keys are laid out is the major difference between the two types of keyboard. They are handled separately here, so you should go on to read the section for your particular keyboard.

The Standard LSI keyboard - Along the top of the keyboard is a row of 24 keys marked F1 to F24. To the right of the numeric keypad is a further block of twelve keys, seven of them marked F25 to F31, one marked HOME, and four marked with arrows. Another key is at the bottom right hand corner of the QWERTY cluster, marked F32.

That makes a total of 37 keys which can be programmed for special purposes. In addition, the 24 keys at the top can be set up to generate two codes; one with the **shift** key held down, and one **unshifted.**

There are therefore 61 different special function sequences that can be produced.

The IBM-type keyboard - The special keys are in two blocks. To the left of the QWERTY cluster is a block of ten keys marked F1 to F10. In addition, when the NUM LOCK key is off, the **scroll lock**, **7**, **9**, **1**, **3** and **0** keys on the numeric keypad can be programmed for special functions.

Also with the NUM LOCK key off, some other keys on the keypad assume special, but fixed, functions. The '.' key becomes the DEL key, and the '8', '4', '6' and '2' keys are used to move the cursor about the screen. They are marked with arrows under the numbers to indicate the direction of movement.

ROUTINE MAINTENANCE

The only components of your system which need routine maintenance are the disk drives.

The complexity of the disk drives means that you should not attempt to maintain them yourself, and we recommend that the drives are serviced annually by an authorised service engineer.

TECHNICAL SPECIFICATION

5.1 The Processor Unit

Mechanical

- Polyurethane foam moulded base and facia
- Folded aluminium lid
- Sectional steel rear plates
- Optional additional EMI screening
- Optional UL fire retardancy
- Size 400mm (depth) x 450mm (width) x 145mm (height)
- Weight approx 7kg depending on configuration

Environmental

Storage
-30°C to +70°C
0% to 90% relative humidity

Operational
0°C to +40°C
15% to 70% relative humidity

Electrical

- 210-250V AC; 45-65Hz
- Optional 100-110V AC; 45-65Hz
- 170 watts max (full options); 100 watts typically
- Fan cooled. Less than 31 dBA noise at 50Hz (35dBA at 60Hz)
- 3 pin mains inlet
- 3 pin 2A mains outlet (unfused) for monitor
- Fuse and switch
- Switching power supply unit
- All electrical specifications meet BSI, CSA, VDE and UL requirements

TECHNICAL SPECIFICATION

Electronics

- Intel 8088 CPU
- Zilog Z80B CPU
- 128 kbyte DRAM + parity
- Optional 128 kbyte DRAM + parity
- Up to 32 kbytes EPROM
- Two serial asynchronous ports (RS232)
- Bidirectional parallel port (Centronics compatible)
- Real time clock with battery backup
- Floppy disk interface with DMA
- Optional Winchester interface with DMA
- Serial TTL keyboard interface
- LSI expansion bus interface
- CRT controller - 25 rows of 80 characters (7x9 character in 9x13 cell)

VDU Attributes

Colour
  8 foreground
  8 background

Monochrome
  Reverse video
  High intensity foreground
  Grey background
  Blank
  Two user attributes

Both
  Underline
  Blink
  Double height rows
  Double width rows

TECHNICAL SPECIFICATION

VDU Features

Dynamic slow scroll (3 speeds plus jump scroll)
Split screen (for status message line)
Soft font of 256 characters (split into 2 fonts of 128 characters each)

5.2 The Monitor

Monochrome – this is a high quality 12" monitor with green P34 phosphor. The refresh is approximately 52Hz and provides a good flicker-free image.

Colour – a 12" direct drive 8 colour monitor similar in style to the above.

5.3 The Keyboard

LSI layout – low profile moulding with 109 sculptured keys, 37 programmable by the user and all programmable at system level for language changes.

IBM-PC layout – low profile moulding with 88 keys similar in style to the IBM Personal Computer keyboard. 16 keys are programmable by the user.

5.4 The Printer Interface

Printers used with your system may have serial or Centronics compatible parallel interfaces.

The operating system uses the parallel interface by default. To use a printer with a serial interface it is necessary to set up the correct Logical/Physical Device Assignment using the CP/M command **DEVICE** and/or the system utilities **PARMGEN** and **LOADPARM**.

In addition the baud rate, parity, number of stop bits, etc. must be set up using the system utility **PARMGEN** to match the parameters required by the printer. The parameter file created by **PARMGEN** must be loaded into your computer's memory, prior to use of the printer, using the utility **LOADPARM**. This may be done automatically by including the appropriate **LOADPARM** command in the **AUTOEXEC** command file.

Use the CP/M **DEVICE** command to display the current assignments for the list device **LST:**.

TECHNICAL SPECIFICATION

Printer Parallel Port

A standard 25 way female **D-type** connector is provided on the rear panel for connection to a Centronics type printer. Pin connections at the computer end of the interface cable are listed below:

Pin No.	Signal
1	*DATA STROBE
2	DATA BIT 0
3	DATA BIT 1
4	DATA BIT 2
5	DATA BIT 3
6	DATA BIT 4
7	DATA BIT 5
8	DATA BIT 6
9	DATA BIT 7
10	*ACKNOWLEDGE
11	BUSY
12	PAPER EMPTY
13	not connected
14	not connected
15	*ERROR
16	not connected
17	not connected
18	GROUND (0 volts)
19	SCREEN
20-25	GROUND (0 volts)

* denotes negative true.

TECHNICAL SPECIFICATION

RS232 Ports

These two standard interfaces are intended for serial communication between your computer and certain printers (e.g. daisy wheels), remote terminals, and other computers. The normal **handshake** signals are incorporated for use if required. Your computer is configured as Data Terminal Equipment (DTE). A standard 25 way male **D-type** connector is provided on the rear panel for each port. Pin connections are listed below:

Pin No.	Signal
1	Protective Ground (Chassis)
2	Transmitted Data (Output)
3	Received Data (Input)
4	Request To Send (Output)
5	Clear To Send (Input)
6	Data Set Ready (Input)
7	Signal Ground
11	Current Loop Tx+
13	Current Loop -ve supply
14	Current Loop Tx-
16	Current Loop Rx-
18	Current Loop +ve supply
20	Data Terminal Ready (Output)
25	Current Loop Rx+

TECHNICAL SPECIFICATION

Typical RS232 Interconnection

To Printer or VDU:

```
Computer (DTE)                VDU or Printer

  Pin                              Pin
  No.                              No.

   1  -----------------------   1
   2  ----------\ /----------   2
   3  ----------/ \----------   3
   4  -                     -   4
       |                   |
   5  -                     -   5
   6  ----------\ /----------   6
   7  ---------- * ----------   7
  20  ----------/ \----------  20
```

To Asynchronous Modem:

```
Computer (DTE)                  Modem (DCE)

  Pin                              Pin
  No.                              No.

   1  -----------------------   1
   2  -----------------------   2
   3  -----------------------   3
   4  -----------------------   4
   5  -----------------------   5
   6  -----------------------   6
   7  -----------------------   7
  20  -----------------------  20
```

THE OPERATING SYSTEM

CONTENTS

WHAT IS AN OPERATING SYSTEM

The Operating System is your computer system's housekeeper. It is in fact, a number of computer software programs which control the basic functions of the system.

There are two quite distinct series of programs which comprise the Operating System. The first are the CP/M programs or **commands**, the second are the System Utilities.

CP/M is the industry standard operating system for micro computers, and provides a range of system management functions which are common to the majority of micro computer systems. CP/M stands for Control Program for Microprocessors, and is designed and produced in the USA by Digital Research.

The System Utilities provide a range of functions which are specific to your computer system and are provided by the equipment manufacturer.

1.1 The CP/M System

As your system's housekeeper CP/M is responsible for many aspects of computer usage.

> It is responsible for managing the file directory, for all the system file handling. It is responsible for allocating the file space on the disk, and for keeping track of where it files, which parts, of which files.
>
> It is responsible for the smooth running of the various parts of your system. It controls the interchange of instructions and information between the individual elements within the system; the keyboard, screen, printer, disk drives and the central processor.

To the business user it is the CP/M File Management System which is the most important of its many facilities. Though its other internal functions are equally important to the correct functioning of your system they are essentially internal functions.

1.2 Built in Commands and Transient Programs

The CP/M programs are of two types. The Built in Commands are contained within the CP/M operating system and can therefore be accessed at any time the system prompt is displayed on your terminal screen. They include:

DIR – displays a directory of the disk file contents

ERA – allows files on disk to be erased

REN – allows files on disk to be renamed

TYPE – displays the contents of a file on the monitor screen

USER – sets a user number

The Transient Program Commands can only be used when the specific program is available to be loaded into memory from one of the disk drives. The principal Transient Program Commands are:

PIP – copies the contents of files between disk drives

SHOW – reports on disk drive status

SDIR – reports on file status

DEVICE – reports on logical/physical device assignment

SET – assigns drive and file attributes and manages password protection

SUBMIT – allows a sequence of CP/M commands to be executed automatically

BACK – initiates a **background** task

STOP – stops a **background** task

There are other less frequently required CP/M programs for the advanced user. If you are in this category then contact your supplier for a set of CP/M 86 PLUS manuals. The majority of users will find the information contained in this System Guide sufficient for most purposes.

1.3 System Utilities

The System Utilities provided by the manufacturer include:

FORMAT – for formatting floppy diskettes

WDFORMAT – for formatting a Winchester disk

DISCOPY – to produce a duplicate copy of an entire diskette

BACKUP – to copy the contents of a Winchester disk onto diskette(s)

KEYGEN – maintains **user specified** function key tables

LOADKEY – invokes **user specified** function key tables

PARMGEN – used to maintain system parameter details

LOADPARM – to load a system parameter table

FONTGEN – maintains **user specified** character font tables

LOADFONT – invokes **user specified** character font tables

TIME – to display or set the systems calendar/clock

THE CP/M FILING SYSTEM

An understanding of the CP/M Filing System is essential to the successful management of the system.

Every character, report, letter or other collection of related data for an accounting system, or word processor etc. is stored by CP/M in a **file**. To CP/M, a file is any collection of related facts, information and data. It has a beginning, and it has an end.

Each file must have a unique name. The form of that name must follow CP/M conventions. This we will discuss in detail under File Naming.

In some cases the computer program in use will set up its own file names. In the AXIS System for example the files containing your accounting data are created by the AXIS program itself when it is first installed. If you are writing your own programs, using PBASIC for example, or you are using a word processor then you will be responsible for defining your own file names.

Once a file name has been defined, CP/M records that name in the directory for the particular diskette (or disk drive) on which it is stored. The directory also records where on the disk it has stored that file.

Each time you format a diskette, CP/M puts a series of what can be regarded as grid lines on its surface similar to those used on maps. The grid lines form sectors. It uses this grid reference of these sectors to record where it stores the contents of each file.

One part of CP/M is entrusted with the management of how and where it stores the data and files you trust to its care. It does this via a part of CP/M called the **BDOS**. This stands for Basic Disk Operating System. If you have a disk problem, you may see CP/M display a message:

```
CP/M Error On A: Disk I/O
Bdos Function = 20 File = BASIC    .$$$
```

This means that CP/M has lost its place on the map. It may just be speck of dust on the surface of the diskette but that will be enough. So handle your diskette with care.

CP/M organises its filing system using a very simple naming structure. It is called a symbolic naming structure. It allows you to call a file by a descriptive name. It does not have to be referred to by a difficult to remember code. The rules controlling the naming of a file are discussed in the File Naming section below.

THE CP/M FILING SYSTEM

The rules of file naming must be understood by all operators.

There are other aspects of the filing system which will become progressively more important as your experience and understanding grows; how it stores data or text on the disk, and the special charactersics which you can assign to a file or group of files.

CP/M has a **dynamic** filing system. You don't have to tell CP/M before you start, the amount of space it must allocate for a new file. Instead CP/M organises its filing system to allow you can keep adding to a file providing their is enough space on the disk.

The volume or size of a file is measured in **bytes**. In everyday use a byte is one character of information. Because a character is a relatively small unit, file sizes are usually measured in units of 1000 characters (strictly 1024 characters). The shorthand for 1000 is **k**. So, a file which is 10k bytes contains approximately 10,000 characters (or exactly 10,240 characters).

2.1 File Naming

CP/M provides computer programs of all types with a consistent method for naming files. The standard format for a CP/M file name is:

DESCRIPTION.TYPE

The description may be up to eight characters long, either numbers or letters are valid. The type, which may be used to group together files for a common purpose, is optional. If present, a full stop separates the type from the description. The type may be up to three characters long. Some punctuation marks are allowed in file names, others are not. Those **not** allowed are:

```
[ ] ? * ; : , . < >
```

There are also some **file types which should be avoided** in general usage. These are:

```
CMD - 16 Bit program files
COM -  8 Bit program files
BAS - BASIC program files
SUB - CP/M Submit Files
OVR - program overlay files
OVL - program overlay files
BAK - text (word) processor back up files
$$$ - temporary system files
KEY - function key tables
FNT - font tables
PRM - system parameter tables
```

Either upper case or lower case letters can be used describe the file name in a CP/M command line. CP/M will translate all lower case letters to upper case (or capitals).

The CP/M file naming convention allows you to describe files using descriptive naming. That makes recognition very simple. But some form of orderly file naming is recommended.

Intelligent file naming allows the **file matching** facilities available within CP/M to be used to maximum effect.

2.2 File Matching

File matching is a CP/M facility which allows files with some common attribute in the file name to be accessed as a group. It allows a command line to be specified which will handle these groups of files.

It is the ability to access this CP/M facility which should encourage the use of logical file naming.

The two characters which allow file matching are the ? and the *. The ? and the * in filematching operate as **wild card** characters, just as the Joker card can in some card games. Their inclusion in a command line will cause CP/M to recognise any file which matches that general condition.

Why two characters? Because there are two conditions. The **?** is specific to a character. Say you wish to match all files with four characters commencing with an 'L'. The reference would be **L???**. If you required all files commencing with an 'L' regardless of the number of characters in the description, then the reference would be **L***.

The most common use of * is for block operations.

All files on drive B: can be referenced thus:

```
    B:*.*
```

All files of the COM type on drive C: can be referenced thus:

```
    C:*.COM
```

All files called LESSON?? on drive A: can be referenced thus:

```
    A:LESSON*.*
```

You can use this file matching with PIP, SDIR, SET, DIR, ERA. It **cannot be used** with TYPE, REN.

2.3 Drive Logging

Drive logging is a means of telling CP/M which drive it must use to look for the files specified.

The drive specification is required when a file is used which is not on the **current logged disk drive**.

CP/M indicates the current logged drive via the system prompt. For example if an A> is displayed on the screen the system is logged onto drive A:. If the system prompt is C> it is logged onto drive C:.

If you have a system with two floppy disk drives, the left hand drive is called A., the right hand drive is called B:.

If you have a Winchester disk system, the floppy disk drive is called C:, and the Winchester disk is split into two logical drives A: and B. .

To access a file on a drive other than the current logged drive, the file name will need the drive prefix.

For example, if the system prompt says A> and you wish to access a file called TEST.DAT on drive A: the filename TEST.DAT will be adequate. However, if TEST.DAT is on a diskette loaded in Drive B: the file must be described in the command line as B:TEST.DAT.

You can change the logged drive quite simply. The following sequence shows how to change the logged drive from A: to B:

```
A>B:<CR>
B>
```

Note: CP/M does not record the drive designation with the file name. Thus a file called TEST.DAT created when a diskette was in drive B: becomes A:TEST.DAT whenever the diskette is loaded into drive A:.

2.4 Write Protection

CP/M recognises two quite distinct file processes; it can **read** from a file; it can **write** to a file. Reading does not change the contents of a file, Writing does. There are some occasions where you may only wish to read a file, and to specifically prevent the system from writing or changing the contents of a file.

Write protection is a means of preventing the system writing to a file. Write protection can be applied either to an individual file, a group of files, or to an entire disk.

What benefits does write protection offer in normal daily use?

First it will stop you inadvertently copying over valuable information.

Secondly it may stop you editing a useful file. For example if you are reading a file on a write protected disk using a word processor. and you inadvertently make a change, the **write protection** will stop you or anyone else inadvertently writing over the existing data.

You **write protect** all files on a 5" diskette by applying a metallic label to the write protect notch of the diskette. Not only the Operating System but also the disk drive itself will sense the presence of the **write protection** and prevent any data from being written onto the diskette. All master disks should be protected in this way especially when security copies are being made; this prevents you from inadvertently corrupting your master disks should they be loaded into the wrong drive during the copying procedure.

You can assign **Read Only** status (RO) to a disk drive, a particular file or a group of files using the CP/M command SET. This will prevent another user changing the contents of the file.

For example:

```
A>SET B: [RO]            sets drive B: to Read Only status
A>SET *.CMD [RO]         sets all program files on A: to RO status
A>SET MESSAGE.TXT [RW]   sets the file to Read/Write status
```

2.5 Logical Devices

Logical devices are the means by which your computer system receives and presents data.

These devices, are called CONIN:, CONOUT:, AUXIN:, AUXOUT:, and LST:.

CONIN: is a console input device usually a keyboard
CONOUT: is a console output device, usually a monitor or VDU
AUXIN: is an auxiliary input device, (e.g. a reader)
AUXOUT: is an auxiliary output device, (e.g. a paper tape punch)
LST: is a listing device, a printer for example.

Note that the above logical device names are followed by a colon in the same way as logical disk drives, A:, B:, etc.

The AUXIN: is a user defined device which can only input data into the system. The AUXOUT: and LST: devices are receive only devices.

Each of these devices must be **assigned** within the system to one of the physical Input or Output connections provided on your computer. In practice the principal use of this facility is to define which outlet port of the system is to be used by the printer.

The standard convention used in your system is that the LST: device, the printer, is assigned to LPT (the line printer or parallel port). You can permanently change this for your system using the system utility program called PARMGEN.

You will only need to change the device assignment if you change the physical specification of your system. Normally the device assignment is pre-set for your system. See also the CP/M command DEVICE.

2.6 CP/M and Application Software

It is a fact that CP/M is sold by the industry as being the answer to all the users problems of selecting and using application sofware. However you should be aware some of the pitfalls you may encounter if you propose purchasing software from other than a reputable supplier. You cannot buy application software off the shelf and run it immediately unless it has been configured for your computer and keyboard/screen. There may even be problems with the printer you use unless you are very specific.

CP/M will only provide a consistent working environment for what is often called CP/M application software when, and only when, it has been installed or configured for your particular system.

The installation requirements for software packages vary considerably from software package to software package. The ease with which software can be adapted to run on your system is described by its **transportability**.

The key factors (but not the only ones) are as follows:

> The disk format for the master copy must be consistent with the format used by your own system. Any application programs which you purchase have to be read into your system's memory for execution. If your system cannot read the disk format supplied then you cannot run the programs.
>
> The control characters used by the program developers to achieve certain effects on the screen and printer must be compatible with or configured for your system. Some programs can be configured easily by non-technical users, others cannot.
>
> Check before you buy new software that the configuration is within your own capability. Ask the dealer to run the install or configurator program for you to check. If there isn't one, beware. Look for an install or configurator which relies on simple Yes/No type answers.

CP/M BUILT IN COMMANDS

3.1 DIR – Taking a Directory

The DIR function enables directories to be displayed either on screen, or on the printer. It can be used whenever the system prompt is displayed on the screen.

To obtain a complete directory of a disk key **DIR** or **dir** at the system prompt. Unless you tell CP/M otherwise it assumes you require a directory of the logged drive. To obtain the directory of a drive other than the logged drive the drive designation must be specified. For example:

```
A>DIR B:<CR>
```

Note the space after DIR, the ':' after the drive letter and the <CR>.

The system responds by displaying the names of all the files on that drive and then returns the system prompt for the next instruction.

To print out a directory first check that the printer is connected and ready for use. Now key <CRTL><P>, i.e. Press and hold down the key marked CONTROL or CTRL and simultaneously key the letter 'P'. All characters displayed on the screen by CP/M will now also be transmitted to the printer. Enter the DIR command and the directory will be displayed and printed. To switch off the printer key <CRTL><P> again.

This print facility is also useful to obtain a **hard copy** output from other CP/M commands.

DIR can be used to confirm that a specific file is on a particular drive or diskette, or to produce a selective directory of files with common letters or numbers in the file name.

For example to check that a file called TRIAL.TXT is on drive B:

```
A>DIR B:TRIAL.TXT<CR>
```

If the file specified is resident on the drive specified, the system will echo the file name and return the A> for your next instruction as follows:

```
A>DIR B:TRIAL.TXT<CR>
B: TRIAL    TXT
A>
```

If the file specified is not resident on the drive specified the following response will be produced:

```
A>DIR SAMPLE2<CR>
File not found: SAMPLE2
A>
```

The CP/M file matching technique can be used to check the location of groups or related groups of files. The wild characters are ? and *.

To look for all files on drive B: which have the BAK extension key:

```
A>DIR B:*.BAK<CR>
```

To check all files with just four characters in the name we key:

```
A>dir ????<CR>
```

3.2 ERA – Erase File(s)

The ERA command is the CP/M Function which allows you to erase a file. It can be used at any time that the system prompt is displayed on the screen.

Unless told otherwise CP/M assumes that you require to erase a file on the Logged Drive.

For example, to erase the file TRIAL.TXT on drive B: you should key:

```
A>ERA B:TRIAL.TXT<CR>
A>
```

Note the space after the ERA and the colon after the drive letter.

There is no response if the file is erased except that the system prompt is returned for your next instruction.

If the file specified is not present on the drive specified CP/M will report an error as follows:

```
A>ERA B:TRIAL.TXT<CR>
File not found: TRIAL.TXT
A>
```

To summarise: to erase a file, specify the drive name if not the logged drive, and the file name. Remember the space after ERA, the colon after the drive name and the confirmatory <CR>.

Files can also be erased as groups with some common letters or numbers in the file name description or type. CP/M's two wild characters; the * and the ? are used. Remember, ? represents any single character. * can represent a single character, or a number of characters.

For example to erase all the files on Drive B: which have the common BAK extension:

```
A>ERA  B:*.BAK<CR>
```

To erase only those files on Drive A: with four characters and only four characters in the filename use the following command:

```
A>ERA ????<CR>
```

To erase all the files on Drive B: key:

```
A>ERA B:*.*<CR>
```

In response to this request, CP/M will present a cautionary option **Confirm delete all user files (Y/N)?**. This prompt requires a response of **y** for yes or **n** for no.

3.3 REN – Rename File(s)

REN is the CP/M Function which renames a file of your choice. REN can be used at any time when the system prompt is displayed.

Unless otherwise instructed CP/M assumes that you require to rename a file on the Logged Drive.

To rename the file WORK1 to a new name MANUAL.TXT enter the command as follows:

```
A>REN MANUAL.TXT=WORK1<CR>
```

Note the space after REN and the confirmatory <CR>. If the file WORK1 had been on drive B: then the drive specification **B:** would be inserted prior to the new file name thus **B:MANUAL.TXT**. Both file names (old and new) are obviously on the same drive.

There is no system response when the file is renamed. (Run DIR to confirm).

If a file with the new name exists on the drive designated, CP/M will report **File Exists**. If CP/M cannot find a file with the old name on the drive specified it will report **File not found**.

It is important to remember with the rename command that the new name is stated before the old name. This sequence will help you remember.

What is the current logged drive?
On which drive is the file to be renamed?
What is the new file name?
What is the old file name?

3.4 TYPE - Display a File

The command TYPE will display the contents of a text file on screen or on the printer.

For example to display the file TEST.DAT (which is on drive B:) on the terminal screen the command line is:

```
A>TYPE B:TEST.DAT<CR>
```

To output the file to the printer the command line would be:

```
A>TYPE B:TEST.DAT<↑P><CR>
```

Where < ↑ P> means **hold the CTRL key down and simultaneously press the P key**. When the function is complete the < ↑ P> must be repeated to switch off the output to the printer.

CP/M will assume the logged drive unless instructed otherwise.

The display of the file can be arrested and restarted at any time by keying < ↑ S> to freeze the display and < ↑ Q> to restart it.

If the file requested is not present on the stated drive, a **file not found** message is displayed.

3.5 USER – Define a User Area

The disk space available to users under CP/M may be divided into separate user areas. Each area will contain files belonging to the designated user. CP/M allows up to sixteen user areas to be defined. The standard default user area is **User 0**. If you are the only user of your system then all your files can quite happily reside in the default user 0 area.

However, if you have a Winchester disk system you may find user areas useful to group together files relating to different tasks. For example, you may choose to assign user 1 for word processing files and user 2 for spreadsheet operations. An otherwise unassigned user area is also very useful for copying the contents of a floppy disk onto another floppy disk via the Winchester. (See under PIP for details).

Files in any of the defined user areas (1-15) can only be accessed by an operator in that user area. They will **not** show up on another user's directory for example.

A user area can be opened at any time that the system prompt is displayed thus:

```
A>USER 1<CR>
1A>
```

Note the 1 preceding the A which shows the current user number. For user area 0 (the default user number) this position in the system prompt is blank

The user number must be in the range 0 to 15 inclusive.

DIR, ERA, REN and TYPE when used within a defined user area will only operate on files within that user area.

However, files described as System Files in USER area 0 **can be accessed** by all users.

PIP can be used to copy files between user areas.
Use the command:

```
A>SHOW [USERS]<CR>
```

to obtain a list of users and the numbers of files within each user area.

4.1 PIP – Copy a File

PIP stands for Peripheral Interchange Program, and is used to copy the contents of files within a drive, between drives, and from disk to the screen, and printer. **Note: when a file is PIPped, the contents are copied to a new file. The original remains unchanged.**

PIP is a transient CP/M program and can only be used when PIP.CMD is available on the current drive.

A PIP session is commenced at the system prompt as follows:

```
A>PIP<CR>
CP/M-86 Plus PIP Version 3.1
*
```

The * is displayed by transient program PIP. It tells you that PIP is awaiting your instruction. The command line is entered at the *. You can enter several command lines in turn, and PIP will return the after each command * ready for your next instruction.

If you only have a single command then it can be entered on the same line as the PIP call. A single line command to copy a file called TRIAL.TXT on drive B: to a file of the same name on drive A: would look like this (note the space after PIP):

```
A>PIP A:=B:TRIAL.TXT<CR>
```

If there is already a file called TRIAL.TXT on drive A: when the command is given, the above instruction would cause the new file to overwrite the file of the same name. **Caution – there is no warning.**

CP/M's file matching facilities can be used effectively to copy groups of files with some common attribute in the file name. For example:

```
A>PIP B:=A:*.*<CR>
```

will copy all files on drive A: to drive B:.

```
A>PIP B:=A:T???
```

will copy all files commencing with T which have four characters in the file name.

The file name may also be changed when the contents of the file are copied to the new file.

As an example:

```
A>PIP A:TEST.BAK=B:TEST.DAT<CR>
```

This will copy the contents of a file called TEST.DAT on drive B: to a file called TEST.BAK on drive A:.

To help you with the format of PIP command line remember:

What is the Destination Drive?
What is the Destination Filename?
What is the Source Drive?
What is the Source File Name?

As its name implies, PIP, Peripheral Interchange Program, can be used to copy files from a disk drive to your printer, or to your monitor screen. If your printer is assigned to the CP/M list device, LST:, you can print from disk to the printer using PIP as follows:

```
A>PIP LST:=B:LESSON1<CR>
```

In general terms you can use PIP as you would use the built in TYPE command. But PIP gives you pagination control. Under normal circumstances, your word processor looks after pagination. TYPE ignores these page breaks and just dumps the contents of the file onto the screen and/or printer.

With PIP, any number of lines per page required can be set. PIP's LST: command line provides this flexibility during listing without affecting the permanent data files themselves in any way. A typical command line would read:

```
A>PIP LST:=B:LIST.DAT[P40]<CR>
```

The characters in the [ ]'s say start a new page every 40 lines.

PIP can also join files together. This is called file **concatenation**.

Files can be concatenated to form one new file, on the same or another drive, or to **chain** a number of files to the printer to make for example **hard copy** backup's of a set of data files. The command below would create one file called LIST.DAT on drive A:, which would contain the contents of the three files on drive B:.

```
A>PIP A:LIST.DAT=B:LIST1.DAT,B:LIST2.DAT,B:LIST3.DAT[V]<CR>
```

Note: specify the drive, and the new file name to create the destination and that the file names to be joined are separated by commas. The [V] is a special code which says please verify the data transferred. A wise precaution for important data, even though the copy will take slightly longer.

PIP can be used to copy files between **user areas** on the same drive or on different drives.

```
A>PIP A:=B:TEST.DAT[G2]<CR>
```

This command line will copy a file called TEST.DAT from user 2 on drive B: to the current user area on drive A:. Note the program PIP.CMD must be available in the current user area **or PIP.CMD must be assigned SYStem status and reside in user area 0**. (See the SET command).

An otherwise unassigned user area is also very useful for copying the contents of a floppy disk onto another floppy disk via the Winchester.

To copy the contents of a floppy disk onto a second disk via user area 5 on the Winchester proceed as follows:

Insert the original floppy into drive C:
Use PIP to copy the floppy files onto user area 5 of drive B:

```
A>PIP B:[G5]=C:*.*<CR>
```

Insert a newly formatted floppy disk into drive C:
Use PIP to copy the files onto the new floppy:

```
A>PIP C:=B:*.*[G5]<CR>
```

Erase the files from the Winchester:

```
A>USER 5<CR>
5A>ERA *.*<CR>
```

4.2 SHOW - Display Disk Drive Information

The SHOW command is used to report on disk drive status. Use of this command will not affect your system or files in any way. The following options are available:

```
A>SHOW B:
```

Reports the access mode (Read Only or Read/Write) and the unused space available on drive B:

```
A>SHOW [USERS]
```

Displays the current user number, any other user areas on drive A:, and the number of files in each user area.

```
A>SHOW B:[DIR]
```

Returns the number of free directory entries on drive B:

```
A>SHOW [DRIVE]
```

Displays the drive characteristics for the logged drive.

```
A>SHOW [LABEL]
```

Displays label information for the logged drive.

```
A>SHOW [PROGRAM]
```

This option shows the names of all programs currently running on the system.

4.3 SDIR – Display File Information

The SDIR command is used to report on the files held on your system. SDIR can be used in the same way as DIR but provides additional information about each file. This information includes the file size in bytes and CP/M records (a CP/M record = 128 bytes), standard attributes (SYS = system file, DIR = user file, RO = Read Only, RW = Read/Write) and user attributes F1 to F4. Use of this command will not affect your system or files in any way. The following options are just some of those available:

```
A>SDIR                display all files on the logged drive
A>SDIR [SYS]          display SYS files only
A>SDIR [DIR]          display DIR files only
A>SDIR [RO]           display Read Only files
A>SDIR [RW]           display Read/Write files
A>SDIR [USER=n]       display files for user area n
                      (n may be 0 to 15 or ALL)
A>SDIR [DRIVE=x]      display files on the drive(s) specified
                      (x may be A to P or ALL)
```

For example:

```
A>SDIR [DRIVE=ALL,USER=ALL] LOSTFILE.DAT<CR>
```

will search for the file LOSTFILE.DAT on all drives in all user areas and will display its details if found.

More specific information is available from the SDIR command. It can provide the individual file sizes for all the files on disk, or for a selection of files with some common character positions in the file name. (Remember that there are two **wild card** characters in CP/M; the ? and the *).

4.4 DEVICE – Display System Devices and Assignments

Displays the physical devices and their current assignments to the system logical devices. This command can also be used to temporarily alter the assignments and device characteristics. For more information consult the Digital Research CP/M 86 Plus Manual. See also the system utility PARMGEN.

Do not attempt to change the assignments unless you know what you are doing.

4.5 SET – Set Special File Control Facilities

The SET command can be used to create disk labels, set drives Read Only, set file attributes, invoke password protection facilities and invoke time and date stamping of files. These facilities are of an advanced nature and will not normally be required by the majority of users. (i.e. Those without Winchester disk drives). For full details of the SET command and its implications consult the Digital Research CP/M 86 Plus Manual.

The following simple examples may be of more general application:

```
    A>SET B:TRIAL.CMD [DIR]<CR>
```

Set the file TRIAL.CMD to Read/Write and DIRectory (non system) status.

Winchester disk users who are partitioning the Winchester disk by defining USER areas can assign a special SYStem status to files in user area 0. A file with system status in user area 0 can be accessed by all users.

```
    A>SET B:TRIAL.CMD [RW DIR]<CR>
```

Set the file TRIAL.CMD to DIRectory (non system) status.

4.6 SUBMIT – A Batch Processing Command

SUBMIT is a CP/M Transient Program which enable repetitive tasks to be accomplished using a single command line.

CP/M commands can be stored in a command file to be executed on demand at a later time. The command file has a SUB type extension. It can contain PIP, SHOW, DIR, ERA, other CP/M commands and application programs in the desired sequence.

The format of the command file is simple. It defines the CP/M functions and the file or files on which the function is to operate. The file-matching characters ? and * can be incorporated, and we can define the statements in non-specific terms using the **$n** character (where n is any number from 1 to 9 inclusive). The **$n** character allows replacement of the **$n** by a parameter from the SUBMIT command line when the SUBMIT process is invoked.

The **$n** sign can be placed anywhere in the command lines within the SUB file. For example the **$n** can be a drive name, a file name, or part of a file name.

A command file might read:

```
DIR $1:
DIR $2:*.BAK
PIP $1:=$2:*.*
ERA $2:*.BAK
↑P
DIR $1:
DIR $2:
↑P
```

In the above the $1 and $2 are drive names. This command file says display the drive $1 directory, then drive $2 directory. Copy all the main files from the second drive to the first. Delete the backups, then print the new directories. Remember that Control-P is the CP/M printer switch.

To initiate the process we simply invoke the command file as below, defining $1 and $2 as the drive names A and B.

```
A>SUBMIT ROUTINE A B<CR>
```

Where ROUTINE is the name of the command file. Notice that we don't need to add SUB to the filename, and that there are just spaces between the A and the B.

For further details on the SUBMIT command consult the Digital Research CP/M 86 Plus Manual.

4.7 BACK – Place a Program in the Background

The BACK command places the specified program in the background. Options provide for console input (INFILE) and console output (OFILE) to be designated. If a console output file is not specified then any console output is automatically written to a .LOG file. There is a NOLOG option to suppress console output if desired.

Once the background program has been loaded the console will prompt you for other work.

For example:

```
A>BACK PIP LST:=*.TXT<CR>

Background output in file: 0 LOG
```

Places the program PIP in the background to print all the files with extension TXT on the printer. The console log output is written to the file 0.LOG.

```
A>BACK SUBMIT MAILSHOT [INFILE=MAILSHOT.DAT,OFILE=MAILSHOT.LOG]<CR>
```

This time the program SUBMIT is started in the background. The command file MAILSHOT.SUB is invoked. Console input is taken from the file MAILSHOT.DAT and console output written to the file MAILSHOT.LOG.

```
A>BACK PIP LST:=B:REPORT [NOLOG]
```

Print the file REPORT on the printer. Suppress console output.

4.8 STOP – Stop a Background Program

Use this command to display the programs currently running on your system and/or to stop one of the programs.

For example:

```
A>STOP<CR>
```

or

```
A>STOP PIP<CR>
```

SYSTEM UTILITIES

5.1 FORMAT – Format a Floppy Diskette

This program is used to map out the **soft sectors** on blank diskettes and to verify the read/write capability of the diskettes's data area.

Note – FORMATTING A DISKETTE WILL DESTROY ANY PREVIOUSLY STORED DATA.

To format a diskette, proceed as follows:

Ensure the file FORMAT.CMD is available on the current drive.

Key in FORMAT followed by a space and the name of the drive to be used to perform the formatting. e.g. A: B: C: or D:

```
A>FORMAT B:<CR>
```

This command will format the diskette on Drive B:

The drive reference may optionally be followed by **/X** where **X** is one of the following:

V	verify the disk only (this operation is non-destructive)
T	transfer the loader track from the logged drive onto the diskette in the specified drive. No formatting or verifying is performed.
I	format an IBM-PC type disk
S	format a single sided IBM-PC type disk

In the absence of any switch, the disk is formatted, then verified and then the loader track is transferred.

The loader track is transferred following successful format (except when formatting IBM format disks). It is not necessary to use the /T switch to achieve this.

No verification is possible on Winchester disks.
Formatting and verification of a Winchester disk is achieved using the fixed disk format program FDFORMAT.

However the FORMAT program is used to transfer a CP/M loader track onto the Winchester drive and to initialise the CP/M directory area.

5.2 WDFORMAT – Format a Winchester Disk

This program is used to map out the **soft sectors** on a Winchester disk and to verify the read/write capability of the Winchester's data area.

Note – FORMATTING A WINCHESTER DISK WILL DESTROY ANY PREVIOUSLY STORED DATA.

To format a Winchester disk, proceed as follows:

Ensure the program WDFORMAT is available on the current drive.

Key in WDFORMAT.

Follow the instructions given on the screen.

If in any doubt contact your supplier

5.3 DISCOPY – Copy a Floppy Diskette

This program enables the operator to copy the entire contents of one diskette to another diskette. It should be used regularly to produce security copies of working diskettes.

To use the DISCOPY program, proceed as follows:

Ensure the file DISCOPY.CMD is available on the current drive.

Key in DISCOPY

Follow the instructions given on the screen.

5.4 BACKUP – Copy a Winchester Disk

This program is intended to 'back-up' all or selected files from a Winchester disk onto one or more floppy diskettes.

Although, when copying from a Winchester disk, the back-up diskette is usually blank (formatted and verified but containing no files), any diskette with files already on it can be used, as the existing files will not be overwritten. In addition to this, if there is insufficient storage space for a file to be completely stored on one diskette, the file will be split and stored on two or more diskettes (if the file is very large).

To back-up a diskette, proceed as follows;

Ensure the program BACKUP is available on the current drive

Key in BACKUP

Depress the RETURN key

Follow the instructions given on the screen

5.5 KEYGEN – Re-define Function Key Table

The keyboard's function keys can be programmed to generate any desired code, or code sequence, when they are depressed. This programming is handled by KEYGEN.

N.B. There are 511 character 'slots' available to the function keys and although they do not have to be shared equally amongst the keys, a complex sequence for one key will deprive the remainder.

To use the KEYGEN program, proceed as follows

Ensure the program KEYGEN is available on the current drive.

Key in KEYGEN.

Follow the instructions given on the screen.

5.6 LOADKEY – Load Function Key Table

This program is used to overlay the resident keyboard function codes with those held in a function key table file generated by KEYGEN.

To use the LOADKEY program, proceed as follows:

Ensure the program LOADKEY is available on the current drive.

Key in LOADKEY followed by a space and the name of the key file to be loaded.

Depress the RETURN key.

5.7 PARMGEN – Re-define System Parameter Table

This program is used to set up the system's parameters, e.g. permanent device assignments, Port Tx/Rx baud rate, etc.

To use the PARMGEN program, proceed as follows:

Ensure the file PARMGEN.CMD is available on the current drive.

Key in PARMGEN.

Follow the instructions given on the screen.

5.8 LOADPARM – Load System Parameter Table

This program is used to alter your system's parameters to those held in a parameter file which has been generated by PARMGEN.

To use the LOADPARM program, proceed as follows:

Ensure the file LOADPARM.CMD is available on the current drive.

Key in LOADPARM followed by a space and the name of the parameter file to be loaded.

Depress the RETURN key.

5.9 FONTGEN – Re-define Screen Character Font Table

This program enables the user to generate his own character fonts for special purposes. Two fonts of 128 characters each may be resident in the system memory at any one time. FONTGEN is used to create or modify font files (of type .FNT) each containing a single 128 character font.

To use the FONTGEN program, proceed as follows:

Ensure the program FONTGEN is available on the current drive.

Key in FONTGEN.

Follow the instructions given on the screen.

5.10 LOADFONT – Load Font Table

This program is used to alter the system's character set to that held in selected font files which have been generated by FONTGEN.

To use the LOADFONT program, proceed as follows:

Ensure the program LOADFONT is available on the current drive.

Key in LOADFONT followed by a space and the name of the font file to be loaded.

Depress the RETURN key.

The file name may be optionally followed by a number of switches to control the font loading; these are :

/1	load font 1 (first 128 characters) – **this is the default**
/2	load font 2 (second 128 characters)
/U	underline the loaded font
/R	reverse the loaded font

If /U is entered as the last item, it may optionally be followed by +n where n is a digit between 0 and 9. This generates the underlining line n 'dots' above the base of the characters.

e.g

```
LOADFONT AXIS/U     loads font 1 underlined
LOADFONT AXIS/1R    loads font 1 in reverse video
LOADFONT AXIS/1U+3  loads font 1 underlined with an underline 3 dots
                    from the bottom of the characters
```

N.B. Once loaded, fonts 1 and 2 form a single 256 byte character set with the second 128 byte font starting at byte 128. i.e. The characters in the second font have the high order bit (bit 7) set to 1.

6.11 TIME - Display/Set Time and Date

This program displays and optionally sets your system's date and time of day clock.

To display the current time and date enter:

```
A>TIME<CR>
```

To set the date and time enter:

```
A>TIME /S<CR>
```

and follow the instructions given on the screen. Note the space between TIME and

LSI

OCTOPUS

SYSTEM
GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

BASIC PRIMER

CONTENTS

BASIC PRIMER

CONTENTS

INTRODUCTION

A computer takes its instructions in the form of a stream of 1's and 0's called binary code. Unfortunately, this sort of code isn't too easy for humans to follow, so a number of languages have been devised over the years to allow the computer's human operators to talk to it. The **English like** instructions of these **high level** languages are translated inside the computer to the binary code that the computer understands.

BASIC was invented in the late 1950's as a teaching aid, designed to help students with the principles of other languages. It proved so simple and easy to learn that it became one of the most popular programming languages, and a standard on the new small computers that started to appear in the 1970's. At the same time, it was developed from its simple beginnings into a powerful language that remains easy to learn.

This primer is designed to be used in conjunction with the Personal BASIC (PBASIC) language provided with your system. Personal BASIC is produced by Digital Research.

BASIC IS A DESK TOP CALCULATOR

2.1 Getting started

In the BASIC on your computer, you have, if nothing else, a very powerful calculator at your fingertips.

We assume that you have your BASIC disk loaded in the logged disk drive, and that the screen is showing the prompt:

```
A>
```

Note that your prompt will show your logged disk drive, B C etc.

Enter the following command:

```
A>BASIC<CR>
```

Remember that pressing <CR> is your way of saying to the computer **Over to you!**

After a few seconds, a copyright message appears on the screen, followed by **Ok**. Just like this:

```
---------------------------------------------------
Personal Basic                          Version 1.0
Serial No XXX-XXXX-XXXXXXX      All rights reserved
Copyright (c) 1983          Digital Research, Inc.
---------------------------------------------------
Ok
```

Whenever you see **Ok** then BASIC is waiting for new instructions from you.

2.2 Some simple calculations

To start with, type in:

```
PRINT 2 + 3<CR>
```

The computer replies immediately with:

```
 5
Ok
```

which, of course, is the right answer.

But this is poor stuff for such a powerful calculator though. Try another example, this time:

```
PRINT 333.462 + 77.829<CR>
```

Once again, the computer responds straight away with:

```
 411.291
Ok
```

Which is once more the right answer. BASIC has all the usual arithmetic functions. Try:

```
PRINT 301 - 87<CR>
```

and the answer comes

```
 214
Ok
```

To multiply and divide, BASIC uses symbols which are different from those you would use on paper. Use an asterisk * to multiply, and a slash / to divide. So

```
PRINT 163.87 * 0.15<CR>
```

gives you

```
 24.5805
Ok
```

and

```
PRINT 447.20 / 25
```

yields

```
 17.888
Ok
```

2.3 Mixed calculations

BASIC can perform mixed calculations very easily. Just remember that BASIC does multiplication and division before it does addition and subtraction. Also it performs any calculation enclosed in brackets before anything outside the brackets. This is the same as the usual rules of arithmetic, and you write out the problem as you would on paper. So now try:

```
PRINT (400 + 2 * 12.80 + 21.60) / 25<CR>
 17.888
Ok
```

First BASIC looks inside the brackets. It works out 2 * 12.80, then adds the result to 400, then adds 21.60 to the sum. Finally, BASIC looks outside the brackets, and divides the total by 25.

If you had missed out the brackets, and written

```
PRINT 400 + 2 * 12.80 + 21.60 / 25<CR>
```

the result would be quite different. Firstly, BASIC would have worked out 2 * 12.80, and then 21.60 / 25. Then the three numbers would have been added together. Try it and see that the answer without the brackets would have been 426.464

BASIC can also raise a number to a power. The special symbol for this is the **up arrow** or **caret** symbol ↑. So to find the value of 7 cubed you could type

```
PRINT 7 ↑ 3<CR>
 343
Ok
```

As a practical example, lets assume that you often need to work out the area of a circle. The well known formula for the area of a circle is pi times the square of the radius. The familiar constant pi has the approximate value 3.14159. So to calculate the area of a circle whose radius is 5 metres, type in:

```
PRINT 3.14159 * (5 ↑ 2)<CR>
 78.5398
Ok
```

That is the area is 78.5398 sq metres.

2.4 How to get out of BASIC

At this point you may have had enough so how do you tell BASIC that you wish to get back to the Operating System prompt. Just enter SYSTEM in response to the **Ok** prompt as follows:

```
SYSTEM<CR>
A>
```

BASIC HAS A MEMORY

3.1 Introducing variables

In fact, unlike most pocket calculators, BASIC has, not one, but as many memories as you are likely to need.

These are called **variables**. Think of them as a set of compartments in a filing cabinet in which you can keep information until you need it. Just as you would label the sections of a filing cabinet to keep track of what they contain, so you have to put a label on the variables that you use in BASIC.

Let's see how this would work with our last calculation.

3.2 The LET statement

Going back to circles, if you do this kind of calculation a lot then it can be tedious to type out **3.14159** every time you want to use the constant pi. It would help to store this value in a variable, so that it is there when you want it. Put 3.14159 in a variable called PI now. This is how we do it:

```
LET PI = 3.14159
```

You can call a variable anything you like, with a few exceptions. It always helps if the name you use gives you some idea of the contents. So PI contains the value of pi, which you aren't likely to forget.

The rules of variable names are quite simple. They must be made up of letters and numbers. The first character must be a letter, not a number. You can't have spaces in the name, though you can separate words with a full stop. And there are a few words that you can't use, because BASIC uses them for its own purposes. You can't, for example, call a variable **PRINT** or **LET**.

By the way, the instruction **LET** is used so often in BASIC that you can miss it out altogether. So to put pi in its box like this:

```
Ok PI = 3.14159<CR>
Ok
```

would work just as well.

If we now try to ınd the area of the circle, with radius 5 metres, we can do it this way:

```
PRINT PI * (5 ↑ 2)<CR>
 78.5398
```

and you can use PI over and over again for other calculations.

BASIC IS A PROGRAMMABLE CALCULATOR

4.1 The idea of a program

Until now, all the instructions you have given to the computer have been carried out immediately. This isn't always very helpful. If you have a number of similar calculations to do, then it can be hard work, and a waste of time, to keep typing out the same formula over again.

Instead, using BASIC, you can set up the formula once and store it away to be used later.

4.2 Line numbers

To defer the execution of instructions in BASIC, we put a number in front of them. We can then store a number of instructions together, each with different line numbers in front of them. A set of deferred instructions is a **program**. When the computer follows the instructions it does so starting with the lowest line number, and going on to the next higher number unless told otherwise. If this sounds confusing we hope it will become clear shortly.

Here, step by step, is a short program which converts a length in inches to a length in millimetres. Type in the program as you go along.

```
10 PRINT "This program converts inches to millimetres"<CR>
```

The first line will display the message between the quotation marks every time the program is run. A constant which appears between quotes is called a **string**. A string need not be a number, but can contain any character except quotation marks ".

```
20 PRINT<CR>
```

If you tell BASIC to PRINT without telling it to print anything, it will simply skip a line on the screen.

4.3 The INPUT instruction

```
30 INPUT "What is the length in inches";INCHES<CR>
```

INPUT asks BASIC to wait for you to type something in. Before it does that, it will display the string in quotes, followed by a question mark which BASIC supplies free of charge. If you missed out the string, BASIC would simply print the question mark before waiting.

When you type in a number, that number is stored away in a variable called INCHES.

```
40 MILLIMETRES = INCHES * 25.4<CR>
```

There are 25.4 millimetres to the inch. Here, BASIC will multiply the value in INCHES by the conversion factor, and store the result in another variable called MILLIMETRES

```
50 PRINT<CR>
```

Another blank line.

4.4 String constants

```
60 PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"<CR>
```

Here BASIC is being asked to print four items, one after the other on the same line. The first is the contents of the variable INCHES, which is what you originally typed in. Next is a string constant, then the contents of the variable MILLIMETRES - the result of the calculation, and finally another string constant.

Note that the separate items must be separated by semicolons ;

4.5 The END instruction

```
70 END
```

As you might expect, END ends the program. You don't always have to put this in, as BASIC will end the program itself when it finds the last instruction. But if you are interested enough to go on to more advanced programming, you will find you will need END, so it is always good practice to put it in.

4.6 The LIST and RUN commands

To check that everything is there, type:

```
LIST<CR>
```

As soon as you hit <CR>, the whole program is presented for your inspection, like this:

```
     10    PRINT "This program converts inches to millimetres"
     20    PRINT
     30    INPUT "What is the length in inches";INCHES
     40    MILLIMETRES = INCHES * 25.4
     50    PRINT
     60    PRINT INCHES;"inches is equal to";MILLIMETRES; millimetres"
     70    END
  Ok
```

When you want to see the program in action, type:

```
RUN<CR>
```

The result will look something like this:

```
  This program converts inches to millimetres

  What is the length in inches? 7

  7 inches is equal to 177.8 millimetres
 Ok
```

Run this program a few times to get the feel of it. Type

```
RUN<CR>
```

each time.

4.7 Unconditional branches – the GOTO instruction

If you want do do a series of these calculations, then you can be spared the effort of typing **RUN** each time. We mentioned earlier that BASIC always goes on to the next higher line number unless you tell it otherwise. This is the time to tell it otherwise, before it reaches the end.

So add a new line to the program. Key in:

```
65 GOTO 20<CR>
```

Then type;

```
LIST<CR>
```

The program you have entered will be re-listed thus:

```
      10  PRINT "This program converts inches to millimetres"
      20  PRINT
      30  INPUT "What is the length in inches";INCHES
      40  MILLIMETRES = INCHES * 25.4
      50  PRINT
      60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
      65  GOTO 20
      70  END
   Ok
```

Notice that BASIC has obligingly put your new line in its proper place, before line 70.

When BASIC has printed the result of your calculation, it will encounter the instruction **GOTO 20**, which means go directly to line number 20.

The GOTO instruction is what is known as an **unconditional branch**, meaning **go to line 20 whatever happens.**

Now type **RUN** and the operation will be repeated:

```
This program converts inches to millimetres

What is the length in inches? 7

7 inches is equal to 177.8 millimetres

What is the length in inches? 33

33 inches is equal to 838.2 millimetres

What is the length in inches? 20

20 inches is equal to 508 millimetres

What is the length in inches?
```

And so on. Each time you give it a calculation to do, the program comes back for more.

4.8 Control-C – the panic button

So how do you stop it when you have finished? There isn't an obvious way. Something is clearly wrong here; you are in an endless loop!

Fear not! All is not lost. If you find yourself in a situation like this, you can stop a program at any point by holding down the <CONTROL> key, and typing **C**. We will write this as CTRL-C in future, to be read as **control C**. Do this now.

```
-- Break -- at line 30
Br
```

and the program holds. Key <CR> and the program continues. Key CTRL-C twice, and the **Ok** symbol returns.

4.9 Conditional branches – IF.....THEN instructions

What went wrong? LIST your program again.

```
     10  PRINT "This program converts inches to millimetres"
     20  PRINT
     30  INPUT "What is the length in inches";INCHES
     40  MILLIMETRES = INCHES * 25.4
     50  PRINT
     60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
     65  GOTO 20
     70  END
Ok
```

The problem lies in that GOTO instruction in line 65. Every time BASIC reaches line 65 of the program, it is told to jump back to line 20. It never gets a chance to reach the END in line 70.

This is not what we want. We want to be given the choice of whether we want to go on or to finish. We can do this by replacing GOTO 20 with a **conditional branch** – where the computer only jumps back under certain conditions.

We would like the computer to stop and ask us if we want to go on, and only jump if the answer is **yes**.

Type in a new line 65 thus:

```
65 INPUT "Again";ANSWER$<CR>
```

4.10 String variables

This time the INPUT instruction is asking for a **string**, and not a number. Strings can be held as variables too, but they have to be handled slightly differently. The variable which will hold your reply is called **ANSWER$**. Notice the dollar sign **$** at the end of the name, which is the special mark of a string variable.

Another line is needed:

```
67 IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20<CR>
```

The **IF** instruction sets the conditions for a branch. First, BASIC will check your reply to see if it is **Y** (for yes) or its lower case equivalent. If the condition isn't true, then the rest of the line will be ignored, and BASIC will go on to the next higher line number, which is 70 END.

If, and only if, the condition ANSWER$="Y" OR ANSWER$="y" is true will BASIC go on to finish the line, and GOTO line 20.

LIST your program once more, before running it.

```
      10   PRINT "This program converts inches to millimetres"
      20   PRINT
      30   INPUT "What is the length in inches";INCHES
      40   MILLIMETRES = INCHES * 25.4
      50   PRINT
      60   PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
      65   INPUT "Again";ANSWER$
      67   IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
      70   END
   Ok
```

Notice two things here. First of all, your new line 65 has completely replaced the old one. A word of caution here: if ever you write a new line, any other line which had that same number will vanish for good, and BASIC won't warn you. So just be careful.

Notice, too, that the new line 67 has once again been placed in its rightful place.

RUN the program. This time the result should look something like this:

```
This program converts inches to millimetres

What is the length in inches? 7

 7 inches is equal to 177.8 millimetres
Again?Y

What is the length in inches? 33

 33 inches is equal to 838.2 millimetres
Again?Y

What is the length in inches? 20

 20 inches is equal to 508 millimetres
Again?N
Ok
```

That's much better. A GOTO instruction, by itself, should be avoided where possible..Many experienced programmers never use it, and they say that if you are tempted to use GOTO you can always find a better way.

BASIC KEEPS YOUR PROGRAMS FOR EVER

5.1 Disk storage

The program you have just created will stay in the computer's memory until you leave BASIC, or you switch off the machine.

When you do one of those things, your program will vanish, without trace. If you want to use it again, you will have to rewrite it.

Unfortunately, programs won't wait around while you go off to use some other software. So we have to have a way of permanently storing a program conveniently. You have just such a storage area, on your floppy disk.

5.2 The SAVE command

To record this program on your disk, type in

```
SAVE CONVERT<CR>
```

For a few seconds there will be a whirring of disk drives while BASIC puts your program in a disk file called CONVERT.BAS. You don't need to add the CP/M file extension **.BAS**, BASIC does that for you. All programs that are saved in this way will have the file type **.BAS**.

5.3 Starting a NEW Program

Have you saved the program? If you have, type in

```
NEW<CR>
```

The NEW command removes all program lines from the computer's memory. Check this by typing

```
LIST<CR>
```

There is nothing there to list.

5.4 The OLD command

Now reassure yourself that your program is still their, by reading it back into memory from the disk. Do this by typing

```
OLD CONVERT<CR>
```

Again the disk drives will whirr for a few seconds, and then the **Ok** prompt comes back. Type

```
LIST<CR>
```

again and your program is back again just as you saved it.

BASIC PROGRAMS ARE EASILY CHANGED

6.1 Programs within programs – the GOSUB instruction

It's all very well, you may think, to have a program to convert inches to millimetres. But if you think in terms of feet and inches then you may well be wondering why you should manually convert feet and inches to inches before you can start.

BASIC can cope with this, and BASIC programs are very easily adapted. Take another look at the program listing.

```
      10  PRINT "This program converts inches to millimetres"
      20  PRINT
      30  INPUT "What is the length in inches";INCHES
      40  MILLIMETRES = INCHES * 25.4
      50  PRINT
      60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
      65  INPUT "Again";ANSWER$
      67  IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
      70  END
  Ok
```

This time we will change the program so that instead of asking for the measurement in inches, it will ask for feet and inches, and convert your input to inches for you.

This is the point where we introduce a very useful instruction - GOSUB. If you tell BASIC to GOSUB a line number, it will branch off to another part of the program, and follow the instructions there until told to RETURN. When it comes back, it obeys the instruction immediately following the GOSUB.

Type in a new line 30 as follows:

```
30 GOSUB 100<CR>
```

Then you will need to write the routine that begins at line 100. Here it is, step by step:

```
100 INPUT "How many feet";FEET
110 INPUT "How many inches";INCHES
```

These input statements are the same as those you have already encountered.

```
120 INCHES = INCHES + FEET * 12
```

At this point, experts in algebra will throw up their hands in horror. This expression, they may say, is meaningless unless FEET = 0. The point to remember is that the statement is not to be read as an algebraic expression, but should be read as **Add twelve times the value in the variable FEET to the value in the variable INCHES. Replace the value in INCHES with the result.**

It is FEET * 12, of course, because there are 12 inches to the foot.

6.2 The RETURN instruction

```
    130 RETURN
```

RETURN says go back to the instruction after the GOSUB.

The program listing now looks like this.

```
    10  PRINT "This program converts inches to millimetres"
    20  PRINT
    30  GOSUB 100
    40  MILLIMETRES = INCHES * 25.4
    50  PRINT
    60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
    65  INPUT "Again";ANSWER$
    67  IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
    70  END
    100 INPUT "How many feet";FEET
    110 INPUT "How many inches";INCHES
    120 INCHES = INCHES + FEET * 12
    130 RETURN
Ok
```

RUN the program again. This time the screen should show:

```
This program converts inches to millimetres

How many feet? 6
How many inches? 8

 80 inches is equal to 2032 millimetres
Again?N
Ok
```

Although we get a correct answer in the end, there are some details not quite right about this.

For one thing, the message displayed at the beginning isn't quite right now. What it should say is "This program converts feet and inches to millimetres".

6.3 The EDIT command

We can edit program lines very easily with the EDIT command. Type in

```
     EDIT 10
```

BASIC responds with

```
     10   PRINT "This program converts inches to millimetres"
  Ed
```

The **Ed** prompt shows that BASIC is in **edit mode**. It will stay in this mode until you key a <CR> to save any changes you have made to the line. Once in edit mode you can modify the line using various **edit commands**. (See Appendix E for more details).
In the following example we shall use just a few of the **edit commands.**

You will notice that the cursor is positioned below the first character of the contents of line 10. Use the space bar to move the cursor along until it is positioned beneath the **i** of the word **inches**. When you reach the **i**, stop and press the <i> key. You will now be able to insert new material into the line. If you space too far in error use the backspace key to move back to the correct position.

Type in the following, **without pressing <CR> afterwards.**

```
      ifeet and
```

Note that the **i**, for insert, is displayed but will **not** be entered into the line. Don't forget to type in a space after the **and**.

Now end the insertion by pressing the key marked ESC. This will display a dollar sign. If you follow this with a <CR>, the complete line containing the new text inserted will be displayed. The screen should look as follows:

```
    10   PRINT "This program converts inches to millimetres"
Ed                                    ifeet and $
    10   PRINT "This program converts feet and inches to millimetres"
Ed
```

This looks okay so we have finished editing and need to get back to the **Ok** prompt. We do this by keying <CR> again.

Finally take a look again at the program listing

```
    10  PRINT "This program converts feet and inches to millimetres"
    20  PRINT
    30  GOSUB 100
    40  MILLIMETRES = INCHES * 25.4
    50  PRINT
    60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
    65  INPUT "Again";ANSWER$
    67  IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
    70  END
    100 INPUT "How many feet";FEET
    110 INPUT "How many inches";INCHES
    120 INCHES = INCHES + FEET * 12
    130 RETURN
Ok
```

Although the conversion to inches has been successful, and the conversion is right, it would be nice if the final display line showed

```
6 feet 8 inches is equal to 2032 millimetres
```

instead of

```
80 inches is equal to 2032 millimetres
```

To do this, we need a way of converting inches back to feet and inches. The best way is to GOSUB another subroutine before the final message is printed.

Put in a new line 55

```
55 GOSUB 200
```

6.4 The INT function

Then start a new routine at line 200

```
200 FEET = INT(INCHES / 12)
```

Here's something new. **INT** is a function built in to BASIC. The expression following it in brackets is called the **argument** of the function. INT is short for **integer**, and means **take the whole number part of the expression in the brackets.** So INT (13.625) would be 13. Here it means divide the number of inches by twelve and store the whole number part of the result in a variable called FEET.

6.5 The MOD function

```
210 INCHES = INCHES MOD 12
```

If INT was strange, MOD is likely to be even stranger. MOD, which is short for **modulo**, is a special kind of arithmetic operator, related to +, -, * and /. X modulo Y means the remainder after X has been divided by Y. So here, INCHES MOD 12 means **divide the value in INCHES by 12, keep the remainder and put it in the variable INCHES.**

6.6 The STR$ function

```
220 FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
```

This is a good example of how we can build up a single string variable from several strings. The process is known as **concatenation**.

We can't put numeric values in a string as they are; they have to be converted to strings themselves. That's what the STR$ function does. Notice the dollar signs that are the trademark of string variables.

The four strings here are joined together by the + sign, which should not be confused with the plus sign in arithmetic.

```
230 RETURN
```

The end of this subroutine.

Your program should now look like this:

```
    10  PRINT "This program converts feet and inches to millimetres"
    20  PRINT
    30  GOSUB 100
    40  MILLIMETRES = INCHES * 25.4
    50  PRINT
    55  GOSUB 200
    60  PRINT INCHES;"inches is equal to";MILLIMETRES;"millimetres"
    65  INPUT "Again";ANSWER$
    67  IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
    70  END
    100 INPUT "How many feet";FEET
    110 INPUT "How many inches";INCHES
    120 INCHES = INCHES + FEET * 12
    130 RETURN
    200 FEET = INT(INCHES / 12)
    210 INCHES = INCHES MOD 12
    220 FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
    230 RETURN
Ok
```

6.7 Deleting lines

We need a new line 60 now to display the final message. This time delete the old line 60 first. Delete a line simply by typing its number, followed by a <CR>.

```
60<CR>
```

Now put in the new line 60

```
60 PRINT FEET.AND.INCHES$;" is equal to";MILLIMETRES;"millimetres"
```

and run the program thus:

```
This program converts feet and inches to millimetres

How many feet? 3
How many inches? 6

 3 feet 6 inches is equal to 1066.8 millimetres
Again?y

How many feet? 8
How many inches? 4

 8 feet 4 inches is equal to 2540 millimetres
Again?n
Ok
```

This is a good point to stop and SAVE your program.

If you save it as **CONVERT** you will lose your first version of the program. When you are developing a program it is always a good idea to keep one or two old versions to fall back on if disaster strikes.

Accidents do happen, and they are not always the fault of the computer. A general power failure, for example, will lose all your work that was in memory at the time.

A good rule to keep to is to SAVE whenever you have typed in as much as you would be prepared to retype if necessary.

Save this program as CONVERT2

```
SAVE CONVERT2<CR>
```

BASIC CAN PUT YOUR RESULTS ON PAPER

7.1 LLIST and LPRINT

You can use BASIC to put your work in permanent form, on paper. This can help you in two ways. Firstly you can produce a permanent record of your BASIC programs and secondly you can produce permanent records of information output by your programs. The key instructions here are LLIST and LPRINT. As you can see, these are the LIST and PRINT instructions which should be familiar to you by now. To divert output from the screen to the printer, an L is added to the beginning of the instruction.

Type in

```
LLIST
```

Provided your printer is connected and switched on, BASIC will now send the listing of your program to be printed on paper.

Now change line 60 of your program to

```
60 LPRINT FEET.AND.INCHES;" is equal to";MILLIMETRES;" millimetres"
```

When you run the program now, the result is printed out by the pnnter.

BASIC STORES INFORMATION FOR YOU

8.1 Developing a filing system

BASIC can store away information on your floppy disk for future reference. As an example, we are going to develop a program which records the name, age and height of various people, and shows you how we put this information on disk, and how to recall it. We will build up a permanent record of the information entered in a **file.**

We will write the program so that it asks for the person's height in feet and inches, and stores it in millimetres. We will use the conversion program you have just created as a subroutine in the new program.

Firstly, though, we will have to make some changes to the program. LIST it again.

```
   10  PRINT "This program converts feet and inches to millimetres"
   20  PRINT
   30  GOSUB 100
   40  MILLIMETRES = INCHES * 25.4
   50  PRINT
   55  GOSUB 200
   60 LPRINT FEET.AND.INCHES;" is equal to";MILLIMETRES;" millimetres"
   65  INPUT "Again";ANSWER$
   67  IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
   70  END
   100 INPUT "How many feet";FEET
   110 INPUT "How many inches";INCHES
   120 INCHES = INCHES + FEET * 12
   130 RETURN
   200 FEET = INT(INCHES / 12)
   210 INCHES = INCHES MOD 12
   220 FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
   230 RETURN
Ok
```

As a subroutine, this is simply going to ask for a measurement in feet and inches, and return the metric equivalent to the main body of the program. We don't need to print any results out, and we don't need to ask if you want to repeat at this stage. So first the irrelevant lines must be removed

Remember that to delete a line, we simply type the line number followed by <CR>.

The lines that need removing are:

```
    10
    20
    50
    60
    65
    67
```

LIST the program again

```
   30  GOSUB 100
   40  MILLIMETRES = INCHES * 25.4
   55  GOSUB 200
   70  END
   100 INPUT "How many feet";FEET
   110 INPUT "How many inches";INCHES
   120 INCHES = INCHES + FEET * 12
   130 RETURN
   200 FEET = INT(INCHES / 12)
   210 INCHES = INCHES MOD 12
   220 FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
   230 RETURN
Ok
```

We won't want to END when the routine is done with, we want to go back to the body of the program. So write a new line 70

```
    70 RETURN
```

8.2 The RENUM command

We want to put this routine at a point lower down the program. We'll start it at line 1000. You don't need to rewrite all the line numbers though, this is a job that BASIC can do for you. Just type

```
    RENUM 1000
```

This tells BASIC to change all the line numbers to start at line 1000. It will put the new numbers in in steps of 10. LIST the program again.

```
      1000 GOSUB 1040
      1010 MILLIMETRES = INCHES * 25.4
      1020 GOSUB 1080
      1030 RETURN
      1040 INPUT "How many feet";FEET
      1050 INPUT "How many inches";INCHES
      1060 INCHES = INCHES + FEET * 12
      1070 RETURN
      1080 FEET = INT(INCHES / 12)
      1090 INCHES = INCHES MOD 12
      1100 FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
      1110 RETURN
Ok
```

An interesting point to notice is that BASIC not only changes the line numbers, it also changes the numbers in the GOSUB instructions so that they point to the right place.

8.3 The OPEN instruction

Now we can set about building the program which asks for and stores the information. Follow each step as we go:

```
10 OPEN "O",1,"HEIGHTS"
```

This is quite a complicated one. **OPEN** means **make a disk file available**. The **O** tells BASIC that the disk file is to be used for output - that is you are going to write information onto the disk. The **1** is the file number. You will use this number within the program to refer to the file. **HEIGHTS** is the name that the file will have in the disk directory.

```
20 INPUT "Name";IN.NAME$
```

BASIC will ask for a name to be entered. You can't use NAME$ for the variable, by the way, because NAME is one of the words used by BASIC for its own purposes. We therefore use IN.NAME$.

```
30 PRINT
```

A blank line.

```
40 PRINT "How old is ";IN.NAME$,
```

Notice that a semicolon appears after IN.NAME$. When BASIC encounters a semicolon at the end of a PRINT instruction, it won't start a new line on the screen, but will print the next item at the end of this one.

```
50 INPUT AGE
```

No prompt string this time. BASIC simply supplies the question mark to the last question, and waits for you to type in a number.

```
60 PRINT
70 PRINT "How tall is ";IN.NAME$;"?"
80 GOSUB 1000
```

At this point BASIC goes off to our subroutine to ask "How many feet", "How many inches", and converts the height to millimetres.

8.4 Remarks – the REM instruction

```
90 REM Ask for height in feet and inches, and convert
```

REM is a slightly odd BASIC instruction in that it tells BASIC to **ignore everything on this line**. REM instructions are valuable though, because they allow you to write comments into the program listing to help you to keep track. REM is short for REMark.

8.5 Writing to a disk file – the WRITE# instruction

WRITE# tells BASIC to put the variables that follow into a disk file. The 1 is the file number to write to – BASIC refers to disk files by their numbers and **not** their names.

```
100 WRITE# 1,IN.NAME$,AGE,MILLIMETRES
```

Follow the file number with a comma, and put commas between the items of information, or fields in the record.

```
110 PRINT
120 INPUT "Another entry";ANSWER$
130 IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
```

This is the old routine to ask if you want to continue entering information or to stop.

8.6 The CLOSE instruction

When you have finished, BASIC closes down the now full file and completes the process of writing it to the disk.

```
140 CLOSE 1
150 END
```

You really do need the END instruction here. If it isn't there, BASIC will look for the next higher line number, and find the subroutine. It will ask you again for a height and convert it. Then BASIC will protest that it has been asked to RETURN without being told to GOSUB anywhere!

LIST the program – or LLIST it if you want a paper copy.

```
      10    OPEN "O",1,"HEIGHTS"
      20    INPUT "Name";IN.NAME$
      30    PRINT
      40    PRINT "How old is ";IN.NAME$;
      50    INPUT AGE
      60    PRINT
      70    PRINT "How tall is ";IN.NAME$;"?"
      80    GOSUB 1000
      90    REM Ask for height in feet and inches, and convert
      100   WRITE# 1,IN.NAME$,AGE,MILLIMETRES
      110   PRINT
      120   INPUT "Another entry";ANSWER$
      130   IF ANSWER$="Y" OR ANSWER$="y" THEN GOTO 20
      140   CLOSE 1
      150   END
      1000  GOSUB 1040
      1010  MILLIMETRES = INCHES * 25.4
      1020  GOSUB 1080
      1030  RETURN
      1040  INPUT "How many feet";FEET
      1050  INPUT "How many inches";INCHES
      1060  INCHES = INCHES + FEET * 12
      1070  RETURN
      1080  FEET = INT(INCHES / 12)
      1090  INCHES = INCHES MOD 12
      1100  FEET.AND.INCHES$=STR$(FEET)+" feet"+STR$(INCHES)+" inches"
      1110  RETURN
Ok
```

SAVE this program under a name such as **HEIGHTFL** and then RUN it. Enter the details as shown in the following example, or provide similar information of your own. You will use it in a few moments.

```
Name? Jim Morrison

How old is Jim Morrison? 34

How tall is Jim Morrison?
How many feet? 5
How many inches? 11

Another entry? Y
Name? Felicity Smith

How old is Felicity Smith? 23

How tall is Felicity Smith?
How many feet? 5
How many inches? 4

Another entry? Y
Name? Otis P Finklestein Jr
```

```
How old is Otis P Finklestein Jr? 56

How tall is Otis P Finklestein Jr?
How many feet? 6
How many inches? 7

Another entry? N
Ok
```

8.7 Reading a disk file

The information on these people is now safely stored away on disk. How do you get at the information? As a last exercise we will build up a new program that reads the file **HEIGHTS** and prints out the contents on paper.

The first thing you need to do is to remove the existing program from the computer's memory. Do this by typing

```
NEW
```

Now follow the steps again.

```
10 OPEN "I",1,"HEIGHTS"
```

Once again we tell BASIC to make a file available to us, but this time we tell it to OPEN "I". This means for input only. You are only going to read from the disk, not write anything on it.

8.8 Arrays or tables – the DIM statement

```
20 DIM IN.NAME$(20),AGE(20),HEIGHT(20)
```

This is your first introduction to the idea of an array. What this means is that BASIC can set up tables in memory. The figure in brackets after the variable name is the maximum number of entries in the table. In this case we have set the maximum at 20, but we could have up to 255 entries if we wanted. Each table can be thought of as 20 different variables numbered 1 to 20.

DIM, by the way, is short for **dimension**.

```
30 COUNT = 0
```

We are going to ask BASIC to count the number of entries in the file, but first we set the counter variable to zero.

```
40 LPRINT TAB(5);"NAME";TAB(40);"AGE";TAB(50);"HEIGHT (metres)"
50 LPRINT TAB(5);"====";TAB(40);"===";TAB(50);"==============="
60 LPRINT
```

Lines 40 to 60 tell BASIC to print a heading on the printer. TAB(40) is a special BASIC function which means **skip to column 40 on the line**. The headings are underlined with = signs, and a blank line is printed.

8.9 Controlled loops – WHILE.....WEND

```
70 WHILE NOT EOF(1)
```

WHILE introduces a kind of controlled loop. We want BASIC to go round the loop reading one file entry at a time until it reaches the end of the file. We don't necessarily know how many entries there are in the file, so we tell BASIC to keep going until it reaches the end.

EOF(1) is a special function used by BASIC to recognise the end of a disk file. It is short for **End Of File number 1**.

If we know how many times we want BASIC to go round a loop, we use FOR and NEXT instructions. These will be explained a little later.

```
80 COUNT = COUNT + 1
```

Increase the value of the counter by one each time BASIC passes through the loop.

```
90 INPUT# 1,IN.NAME$(COUNT),AGE(COUNT),HEIGHT(COUNT)
```

INPUT# 1 tells BASIC to read the values from disk file number 1 into the appropriate table entries.

```
100 HEIGHT(COUNT) = HEIGHT(COUNT) / 1000
```

Change the height in millimetres to a height in metres.

```
110 WEND
```

WEND brings to an end the loop started by WHILE. If there are no more entries in the file, then the loop will be skipped and we have the total number of records in the variable COUNT.

```
120 CLOSE 1
```

Close the disk file; we don't need it any more.

8.10 Controlled loops - FOR....NEXT

```
130 FOR I = 1 TO COUNT
```

We are asking BASIC to go into a loop again, to print out the values in the tables. This time though, we know how many times to go round, so we use a different kind of controlled loop.

The variable **I** starts at 1 the first time through, and increases by 1 each time until it becomes larger than the number of records in COUNT. When that happens, BASIC exits from the loop.

```
140 LPRINT IN.NAMES$(I);TAB(40);AGE(I);TAB(50);HEIGHT(I)
```

We ask BASIC to print the entry, using TAB to put each item under its proper heading on the paper.

```
150 NEXT I
```

As WEND marked the end of the WHILE loop, NEXT marks the end of the FOR loop. The index I is increased by 1 and BASIC jumps back to the FOR instruction to compare it with the upper value specified. If I is larger than **COUNT**, BASIC will skip the loop and carry out the next instruction, which is:

```
160 END
```

SAVE this program under a new name, and then RUN it. On your printer you should get:

```
  NAME                              AGE      HEIGHT (metres)
  ====                              ===      ===============

Jim Morrison                         34        1.8034
Felicity Smith                       23        1.6256
Otis P Finklestein Jr                56        2.0066
```

BASIC IS MUCH MORE BESIDES

In this primer we have only been able to give you a taste of BASIC. All the same, we have brought you a long way, and you have learned enough to make BASIC work for you, on a simple level.

If you are interested enough to carry on working with BASIC, your supplier should be able to provide you with the full language manual. Using the information in there, you will be able to develop sophisticated software of your own.

BASIC SUMMARY

These last few pages aim to set out, in a concise form, what you have learned about BASIC through this primer. Some additional facilities are also included which you may wish to use or experiment with, (see also under Appendices). A comprehensive BASIC manual is available from your supplier.

10.1 Data types

Numeric	6.8 NUMBER	The first is a numeric constant, the second a numeric variable.
String	"A string constant" VARIABLE$	Strings contain any character. The first is a constant, the second a variable.

10.2 Commands

LIST	LIST	Lists the program lines on the screen —
LLIST	LLIST	Lists the program lines on the printer
OLD	OLD CONVERT	Loads the program in the disk file "CONVERT.BAS" into memory
NEW	NEW	Clears the current program from memory
RENUM	RENUM 1000	Renumbers the program in memory starting at line 1000
RUN	RUN	Executes the program in memory
SAVE	SAVE CONVERT	Saves the program in memory and puts it in the file CONVERT.BAS

Note that the above instructions are not used in programs but are commands from the operator to BASIC to manipulate programs.

The above list is not exhaustive – for more information contact your supplier

10.3 Program instructions

CLOSE	CLOSE 1	Closes disk file number 1
DIM	DIM HEIGHT(20)	Sets the size of an array variable
END	END	Ends the execution of the program
FOR	FOR I=1 TO 10	Repeat all the lines until NEXT, adding 1 to I each time until I is greater than 10
GOSUB	GOSUB 1000	Jump to the subroutine starting at line 1000
GOTO	GOTO 20	Unconditional branch to line 20
IF..THEN	IF X=1 THEN GOTO 20	If the condition is true then execute the instruction that follows THEN
INPUT	INPUT "Another";A$	Prints the string, waits for keyboard input and stores it in the variable A$
INPUT#	INPUT# 1,AGE	Reads the next item in a disk file into the variable AGE
LET	LET PI=3.14159 or PI=3.14159	Assigns the value on the right to the variable on the left
LPRINT	LPRINT 3 * 4.5	Prints the string or number (that follows) on the printer
NEXT	NEXT I	Marks the end of a FOR loop
OPEN "O"	OPEN "O",1,"DATA"	Open the disk file called DATA for output. Call it file number 1
OPEN "I"	OPEN "I",2,"DATA"	Open the disk file called DATA for input. Call it file number 2
PRINT	PRINT "Hello"	Prints the string or number (that follows) on the screen
REM	REM Only a remark!	Ignore what follows on the line. Used to make comments in the program.
RETURN	RETURN	Return from a subroutine and continue with the instruction after the GOSUB
WEND	WEND	Marks thr end of a WHILE loop
WHILE	WHILE X=2	Repeat all the lines until WEND as long as the condition remains true
WRITE#	WRITE# 1,IN.NAME$	Writes the variable that follows to disk file number 1

The above list is not exhaustive – for more information contact your supplier

10.4 Functions

CHR$	CHR$(XYZ)	Returns the ASCII character corresponding to the value of the expression
EOF	EOF(1)	A special function which is true when the next disk file item to be read is the end of the file
INT	INT(INCHES/12)	Calculates the expression in brackets and returns the next smaller integer
LEFT$	LEFT$(A$,3)	Returns the number of characters specified starting at the left hand end of the string
RIGHT$	RIGHT$(A$,3)	Returns the number of characters specified starting at the right hand end of the string
SQR	SQR(A)	The square root of the expression
STR$	STR$(13.2)	Returns the numeric value of the argument as a string
TAB	TAB(50)	When printing, skips to the column number specified in the argument

Note that all functions require an argument in brackets

The above list is not exhaustive – for more information contact your supplier

10.5 Operators

+	A$ + " inches"	String operator used to concatenate strings
+	COUNT + 1	Arithmetic addition
-	HEIGHT - 6	Arithmetic subtraction
*	8 * 7	Arithmetic multiplication
/	INCHES / 12	Arithmetic division
↑	10 ↑ 2	Raise to the power of
MOD	63 MOD 9	Modulo - gives the remainder after 63 is divided by 9
=	X = 2	Equals
<>	A <> B	Does not equal
<	A < B	Is less than
>	A > B	Is greater than
<=	A <= B	Is less than or equal to
>=	A >= B	Is greater than or equal to
NOT	NOT EOF(1)	Condition not true
AND	IF A=1 AND B=2	True if both conditions are true
OR	IF A=1 OR B=2	True if either condition is true
XOR	IF A=1 XOR B=2	True if either condition is true *but not if both are true*

LSI OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

HARDWARE OPTIONS

Your system has been designed with expansion in mind. Ask your supplier for details of the following options:

- **Additional Monochrome Monitor**
  12" High Quality with P34 green phosphor

- **Additional Monochrome Monitor**
  As above but with tilt and swivel mounting

- **Additional Colour Monitor**
  12" eight colour

- **Additional Colour Monitor**
  14" sixteen colour

- **Additional Random Access Memory** (RAM) board
  256K RAM
  or
  512K RAM per board

- **8087 Maths Co-processor**

- **Communications board** providing
  2 RS232 Ports (Async only)
  2 RS232 Ports (Sync/Async with DMA)
  1 RS422 with DMA

- **Graphics board (monochrome)** providing
  720 x 325 Pixels

- **Graphics board (colour)** providing
  720 x 325 Pixels x 4 Planes mapped to 16 colours

HARDWARE OPTIONS

- **Network Interface board**

- **Vertical Mounting Kit**

- **Carrying Case**

- **UHF Modulator**

- **Mouse**

- **Plug-in Additional Disk Storage** in the following packages:
  2 x 1.2Mbyte eight inch Floppy Disk Drives
  5 Mbyte Winchester Disk Drive
  10 Mbyte Winchester Disk Drive
  20 Mbyte Winchester Disk Drive
  Cartridge Tape Back-up Unit
  5 Mbyte Winchester Disk Drive complete with Cartridge Tape Back-up
  10 Mbyte Winchester Disk Drive complete with Cartridge Tape Back-up
  20 Mbyte Winchester Disk Drive complete with Cartridge Tape Back-up

- **Esprit II Terminal** for multi-user operation

- **Exec 10 Terminal** for multi-user operation

- **Various Printers** – ask your supplier

Contact your supplier for full details of any of the above options or for any other requirements not listed above.

SOFTWARE OPTIONS

Your system is one of the most powerful tools that you will ever purchase and can be used for a wide variety of tasks limited only by the availability of suitable software and your own ingenuity. The following list of items is far from complete. Ask your supplier for details of other programs which may be suitable for your application.

- **CP/M 86 Plus Manual** – for advanced users (includes disk containing additional Digital Research programs not covered in this System Guide but described in the manual).

- **Personal BASIC Manual** – official Digital Research manual and tutorial for advanced BASIC programmers.

- **Octosoft** – special offer package consisting of a word processor, financial planner and a database package; ask your supplier for details.

- **Alternative Operating Systems** – ask your supplier

- **Other Programming Languages** – ask your supplier

- **Other Application Packages** – ask your supplier

- **AXIS Accessories and Additional Features** – ask your supplier or see your AXIS Modular Accounting Package manual for further details.

- **Other AXIS Programs** – ask your supplier or see your AXIS Modular Accounting Package manual for further details.

Contact your supplier for full details of any of the above items or for any other requirements not listed above.

WORD PROCESSING

One of the most useful aspects of a microcomputer is that it allows you to take advantage of Word Processing facilities.

A Word Processor offers advantages that an ordinary typewriter cannot give you:

You can personalise a letter which is being sent to a number of different people. It is possible to print the letter several times, each time having a different name and address at the top. You can even include personal details in the text, and the letter appears to have been typed individually.

You can build a letter or report from previously written segments of text. Blocks of text you use a lot can be stored away on disk and recalled when you want them by only a few keystrokes.

You can print out your document at any time which suits you. When you have finished writing on the screen, the document is stored on a disk, and is available there whenever you want to alter it or print it.

Once the text has been printed, it is still stored on the disk. So, if you ever want another copy, you can print one straight away.

Typing onto the screen instead of directly to paper means that mistakes can be corrected very easily, and without fuss.

The text is presented on the screen just as it will appear on the paper. You can change the appearance of the page with just a few simple keystrokes. You can change the line width, justify the right-hand margin or centre text as you wish.

SPREADSHEETS

A Spreadsheet is a program package which takes the sweat out of calculations and financial planning. It replaces paper, pencil, eraser and pocket calculator all in the same unit. It is also more powerful.

A Spreadsheet can be regarded as a very large sheet of paper with a grid marked on it. The computer's screen shows you a portion of the sheet, and you can move around easily to see other portions.

You can enter text into the Spreadsheet, to allow you to lay out reports attractively and clearly.

You can write a figure into any cell of the Spreadsheet, and use it for calculations. You will have a number of choices for the way in which numbers are presented - as whole numbers, for instance, or in pounds and pence format.

You can enter a formula into a cell, which the program will then calculate for your automatically. The formula can refer to other cells, or to a range of cells. So you can, for example, add up a column of figures.

By altering a figure in a cell, formulae in other cells which are affected by the change will be automatically re-calculated.

You can change the parameters in a report, and see the effect. In this way you can change one figure in a complicated report, and you will see the changes ripple through the sheet as all the formulae affected are re-calculated. This allows you to **think** on the computer.

When you are satisfied with your report, you can store it away on a disk for future reference. Or, you can print out any portion of the sheet for inspection.

DATA BASE PACKAGES

A Data Base package stores information for you, and retrieves it when you need it. Data Bases vary widely in power, so you should choose one that suits your purpose. Most Data Bases allow you to control the way that information is presented on the screen.

At its simplest, a Data Base can store a file of names and addresses for you, together with a certain amount of other detail. You will be able to search for, or select, information from the file on a key or keys. So, for example, you can pick out a particular surname, or part number, and find the details relating to it.

More elaborate Data Bases will let you search for any attribute, or combination of attributes. So you could select from a customer list those records where the customer lives in Surrey, drives a red Range Rover and breeds Burmese cats – if that is the kind of information you need to have at your fingertips.

The most powerful packages can hold and use several different files at the same time. Many can also perform calculations of varying complexity. You can use these to keep and automatically update stock records for example. Or you can generate invoices, keying in a quantity and a part number, and allow the Data Base to read in the details of the invoice items from another file and calculate the totals.

As well as presenting the information on the screen, most Data Bases will let you produce printed reports on selected records. In some cases the print format is fixed for you, but others allow you complete freedom in designing what goes on your page. In this way you can not only generate your invoice, but set it out on pre-printed forms as well.

Finally, you can use a Data Base in conjunction with a Word Processor for selective mail shots.

LSI

OCTOPUS

SYSTEM GUIDE

INTRODUCTION HARDWARE OPERATING SYSTEM BASIC PRIMER EXPANSION OPTIONS APPENDICES

APPENDICES

APPENDICES

CONTENTS

THE LSI KEYBOARD

The Octopus keyboard is totally soft in that all keys return up/down codes so that any key can in principle be used as a control, shift or lock key. However, some dedication of keys is necessary to provide compatibility with CP/M and applications expecting ASCII keycodes. This dedication is as follows:

The CONTROL, SHIFT LOCK, CAPS LOCK and the two SHIFT keys are dedicated to their usual functions except that the combination of CONTROL, SHIFT and DEL will reset the machine.

The QWERTY pad returns the usual ASCII codes, modified in the usual way by CONTROL, CAPS LOCK, SHIFT LOCK and SHIFT keys.

The four arrow keys and the HOME key may each be programmed according to either method 1 or 2 below.

Each of the function keys F1 thru F32 may be used according to either method 1 or 2 below.

1 The key is a function key which expands into a string of 7-bit ASCII characters (including control characters if required).

2 The key is a SUPERSHIFT key. Held down on its own, it has no effect; held down while a non-function key is pressed, it returns its keycode followed by the normal ASCII code for the non-function key with the usual modification by any of CONTROL, CAPS LOCK, SHIFT LOCK and SHIFT keys.

If multiple supershift keys are held down, multiple lead-ins are returned. If a supershift key is held down while a function key is pressed, the expansion of the function key does **not** include supershift lead-ins.

Keycodes for function keys are as follows:

F1 - F24	80H - 97H respectively
F25 - F28	B0H - B3H respectively
Up arrow	B4H
F29	B5H
Left arrow	B6H
Home	B7H
RigHt arrow	B8H
F30	B9H
Down arrow	BAH
F32 - F32	BBH - BCH respectively

See also under KEYGEN.

MONITOR CONTROL CHARACTERS

The following control codes are supported:

07H	Beep.
08H	Backspace - no effect if cursor in leftmost column.
0AH	Linefeed.
0DH	Carriage return.
1BH	Escape - introduces graphic character if following character is less then 20H or escape sequence if following character is greater than 20H.
1CH	Advance cursor by one character position (no effect if cursor in rightmost column)

Note - all other characters below 20H are ignored.

MONITOR CONTROL CHARACTERS

The following control sequences are available on your system:

Monitor Function	Decimal	Hex	Char
CURSOR UP	27,65	1B,41	ESC,A
CURSOR DOWN	27,66	1B,42	ESC,B
CURSOR RIGHT	27,67	1B,43	ESC,C
CURSOR LEFT	27,68	1B,44	ESC,D
ERASE SCREEN	27,69	1B,45	ESC,E
GRAPHICS MODE ON	27,70	1B,46	ESC,F
GRAPHICS MODE OFF	27,71	1B,47	ESC,G
CURSOR HOME	27,72	1B,48	ESC,H
CURSOR UP (with scroll)	27,73	1B,49	ESC,I
ERASE END OF SCREEN	27,74	1B,4A	ESC,J
ERASE END OF LINE	27,75	1B,4B	ESC,K
CURSOR ON	27,50	1B,80	ESC,P
CURSOR OFF	27,51	1B,81	ESC,Q
CURSOR ADDRESSING	27,89,<y>,<x>	1B,59,<y>,<x>	ESC,Y,<y>,<x>
SET ATTRIBUTE	27,97,<f>,<b>	1B,61,<f>,<b>	ESC,a,<f>,<b>
SET CURSOR HEIGHT	27,99,<h>,<s>	1B,63,<h>,<s>	ESC,c,<h>,<s>
SET SCREEN MODE	27,109,<m>	1B,6D,<m>	ESC,m,<m>
SET SCROLL SPEED	27,115,<r>	1B,73,<r>	ESC,s,<r>
INVERSE VIDEO ON	27,112	1B,70	ESC,p
INVERSE VIDEO OFF	27,113	1B,71	ESC,q

Note on cursor addressing

In the sequence shown above <y> and <x> are the row and column numbers, plus 32 decimal, respectively.

For example:

```
Row  0 is  0+32=32
Row 23 is 23+32=55
Col 69 is 69+32=101
```

In the above table the commas are not part of the sequence.

Note on attribute setting

In the control sequence shown <f> and <b> are the foreground and background attributes respectively in accordance with the following table:

Monochrome:

<f> = @	Default
<f> = A	Blank
<f> = B	Grey background
<f> = D	High intensity foreground
<f> = F	B + D
<f> = H	Underlined
<f> = J	B + H
<f> = L	D + H
<f> = N	B + D + H
<b> = @	Default
<b> = B	Inverse video
<b> = H	Blinking
<b> = J	B + H

Colour:

<f> = @	Black foreground
<f> = A	Blue foreground
<f> = B	Green foreground
<f> = C	Cyan foreground
<f> = D	Red foreground
<f> = E	Mauve foreground
<f> = F	Yellow foreground
<f> = G	White foreground
<f> = H to O	as for @ to G respectively, underlined
<b> = @ to G	as for <f> = @ to G but defines background colour
<b> = H to O	as for <f> = @ to G but defines background colour, blinking

Note on cursor height setting

In the control sequence shown <h> and <s> are the cursor height and style respectively in accordance with the following table:

<h> = @ to L	For cursor height 1 to 13 respectively
<s> = @	No cursor
<s> = A	Static cursor
<s> = B	Fast blinking cursor
<s> = C	Slow blinking cursor

Note on screen mode setting

In the control sequence shown <m> is the screen mode in accordance with the following table:

<m> = @	24 x 80 plus status line
<m> = A	24 x 40 plus status line
<m> = B	28 x 132 plus status line

Note on scroll speed setting

In the control sequence shown <r> is the scroll speed in accordance with the following table:

<r> = A	Slow scroll
<r> = B	Medium speed scroll
<r> = C	Fast smooth scroll
<r> = Z	Jump scroll

Other notes

Your monitor supports an 8-bit character set.

The effect of cursor movements at the edges of the screen. are not programmable. Auto new line occurs when an attempt is made to exceed the line length but not before, e.g. in 80 column mode, auto newline occurs when the 81st displayable character is received, **not** after the 80th character is displayed.

The cursor can appear one position to the right of the rightmost column. This only occurs after writing a character to the rightmost column; it may not be placed there by any cursor positioning commands.

The monitor has some compatibility with the Zenith Z19 and Heath H19 terminals, including the inverse video sequences.

GRAPHICS CHARACTER SET

Business Graphics Character Set

The sequence ESC F will set the monitor into graphics mode. Subsequent characters in the range 60H to 7FH will be mapped to the graphics characters 00H to 1FH until ESC G is received. (Note that graphics characters may also be generated by preceding the character in the range 00H to 1FH by ESC).

The following **business graphics** characters are available in **graphics mode**:

\	◆
a	—
b	▬
c	▬
d	■
e	■
f	▤
g	█
h	▓
i	█
j	┘
k	┐
l	┌
m	└
n	+
o	↑

p	↓
q	_
r	→
s	←
t	├
u	┤
v	┴
w	┬
x	│
y	≤
z	≥
{	π
|	≠
}	#
~	•
(DEL) ▒	÷

ASCII CHARACTER SET

CONTROL SHIFT	ASCII CHAR	HEX	DEC
@	NUL	0	0
A	SOH	1	1
B	STX	2	2
C	ETX	3	3
D	EOT	4	4
E	ENQ	5	5
F	ACK	6	6
G	BEL	7	7
H	BS	8	8
I	HT	9	9
J	LF	A	10
K	VT	B	11
L	FF	C	12
M	CR	D	13
N	SO	E	14
O	SI	F	15
P	DLE	10	16
Q	DC1	11	17
R	DC2	12	18
S	DC3	13	19
T	DC4	14	20
U	NAK	15	21
V	SYN	16	22
W	ETB	17	23
X	CAN	18	24
Y	EM	19	25
Z	SUB	1A	26
[	ESC	1B	27
\	FS	1C	28
]	GS	1D	29
↑	RS	1E	30
_	US	1F	31

ASCII CHARACTER SET

ASCII CHAR	HEX	DEC
SP	20	32
!	21	33
"	22	34
#	23	35
$	24	36
%	25	37
&	26	38
'	27	39
(	28	40
)	29	41
*	2A	42
+	2B	43
,	2C	44
-	2D	45
.	2E	46
/	2F	47
0	30	48
1	31	49
2	32	50
3	33	51
4	34	52
5	35	53
6	36	54
7	37	55
8	38	56
9	39	57
:	3A	58
;	3B	59
<	3C	60
=	3D	61
>	3E	62
?	3F	63

ASCII CHARACTER SET

ASCII CHAR	HEX	DEC
@	40	64
A	41	65
B	42	66
C	43	67
D	44	68
E	45	69
F	46	70
G	47	71
H	48	72
I	49	73
J	4A	74
K	4B	75
L	4C	76
M	4D	77
N	4E	78
O	4F	79
P	50	80
Q	51	81
R	52	82
S	53	83
T	54	84
U	55	85
V	56	86
W	57	87
X	58	88
Y	59	89
Z	5A	90
[	5B	91
\	5C	92
]	5D	93
↑	5E	94
_	5F	95

ASCII CHARACTER SET

ASCII CHAR	HEX	DEC
`	60	96
a	61	97
b	62	98
c	63	99
d	64	100
e	65	101
f	66	102
g	67	103
h	68	104
i	69	105
j	6A	106
k	6B	107
l	6C	108
m	6D	109
n	6E	110
o	6F	111
p	70	112
q	71	113
r	72	114
s	73	115
t	74	116
u	75	117
v	76	118
w	77	119
x	78	120
y	79	121
z	7A	122
{	7B	123
|	7C	124
}	7D	125
~	7E	126
DEL	7F	127

THE PBASIC LINE EDITOR

To allow you to change the lines of your PBASIC programs easily, you can use the built in line editor.

To edit a particular line, say line 100, type in

```
    EDIT 100
```

PBASIC replies by printing the line, then leaving the cursor underneath the first character. Like this:

```
    100   PRUNT "This line is wrong and needs editing"
  Ed
```

The **Ed** prompt shows that PBASIC is in **edit mode.** It will stay there until you type a carriage return to save the changes you have made. Once in **edit mode** you can:

Move the cursor left or right in the line
Insert characters
Delete characters
Search for characters
Replace characters
Leave and re-enter Edit Mode

You do these with the following command keys:

`<CR>`	Save all the changes made and return control to Command Mode.
`<SPACE>`	Move the cursor one character to the right without altering the line.
`<BSPACE>`	Move the cursor back one character to the left without altering the line. CONTROL-H does the same job.
`A<CR>`	Ignore all the changes made so far and put the cursor back to the beginning of the line for re-editing.

nC	Delete n characters starting with the cursor position, and replace with the n characters that you type in immediately afterwards. So *2CINP<CR>* replaces the two characters after the cursor with the three characters INP.
nD<CR>	Deletes the n characters above and to the right of the cursor position.
E<CR>	Saves all the changes made and returns to PBASIC's command level. The same as a carriage return.
<ESC>	Ends the insertion of characters started by an I command without ending the command line.
H<CR>	Deletes all characters up to the end of the line from the cursor position.
I	Inserts characters into the line starting at the cursor position. All characters typed are inserted until you type <ESC> to end insert mode or <CR> to end both insert mode and the current command line.
K	Deletes all characters from the cursor position to the first occurrence of the character typed in following the K.
L<CR>	Returns the cursor to the first character in the line.
<LF>	A line feed, when in insert mode, inserts a line feed and allows characters to be inserted on a new line.
Q<CR>	Ignore all changes to the line, and return to the command mode.
R<CR>	Moves the cursor one character to the right of the end of the line.
S	Searches for the character typed immediately afterwards, and positions the cursor beneath it.
X<CR>	Positions the cursor at the end of the line, and allows characters to be inserted.

PBASIC LANGUAGE DEFINITION

Program lines

Every PBASIC program line has a number. Line numbers indicate the order in which the lines are stored in memory, and are used as a reference in editing.

A line can be up to 255 characters long. More than one instruction can be placed on a line, but each instruction must be separated by a colon (:).

Line numbers must be in the range 0 - 65535

Constants

Constants can be of four types.

1 **String**
A string constant is a set of characters in quotes.
e.g. "This is a string"

2 **Integer**
Whole numbers between -32768 and +32767.
e.g. -39

3 **Real numbers**
Numbers which may contain a decimal point.
e.g. 453.67

4 **Floating point numbers**
A number in scientific notation – a real number times a power of ten.
e.g. 2.3 X 10 ↑ 8 is written 2.3E8

PBASIC LANGUAGE DEFINITION

Variables

There are four types of variable.

1 **String variables**
The name is distinguished by a $ at the end.
e.g. ADDRESS$

2 **Integer variables**
The variable name has a % on the end.
e.g. AGE%

3 **Single-precision numeric variables**
Real numbers with up to 6 significant digits. Names are distinguished by a !, but if no suffix is added, then ! is assumed.
e.g. SALES!, PROFIT

4 **Double-precision variables**
Real numbers with up to 16 significant digits. Names distiguished with a #.
e.g. SIGMA#

Arithmetic operators

↑	Exponentiation	PI↑2
-	Negation	-3
*	Multiplication	2.3 * 8
/	Division	67 / 1.5
MOD	Modulo	167 MOD 12
+	Addition	5 + 3
-	Subtraction	64 - 23

Operations are carried out in the order of precedence above. Expressions in (brackets) are resolved first.

PBASIC LANGUAGE DEFINITION

Relational operators

`=`	Equality	`X = 1`
`<>`	Inequality	`Y <> 3`
`<`	Less than	`LENGTH < 6`
`>`	Greater than	`TOTAL < 3.4`
`<=`	Less than or equal to	`SUM <= 29`
`>=`	Greater than or equal to	`HEIGHT >= 5.0`

Relational operators are used to compare two values. If the comparison is true, -1 is returned. If false, zero is returned. The result may be used to make a decision.

Logical operators

Logical operators perform tests on relations. The operation returns -1 if the expression is true, 0 if false.

`NOT`	True if relation is false	`NOT (HEIGHT = 5.0)`
`AND`	True if both of two relations are true	`(L%=2) AND (N<>0)`
`OR`	True if one or both of two relations is true	`(ANS$="Y") OR (ANS$="y")`
`XOR`	True if one relation is true, but not both	`(LEN = 2) XOR (WID = 3)`

PBASIC INSTRUCTIONS

CALL	CALL ASMPROG	Calls an assembler subroutine written by yourself. The vapiable ASMPROG holds a hexadecimal start address.
CHAIN	CHAIN "NEWPROG.BAS"	Reads the program NEWPROG from the diskette and runs it.
CLEAR	CLEAR	Sets the value of all numeric variables to zero.
COMMON	COMMON A,B,C,SUM	Passes the variables in the list to a CHAINED program.
DATA	DATA 1,2,"Hello"	Stores numeric and string constants within memory, to be read by a READ instruction.
DEF FN	DEF FNADDS=A+B+C	Defines and names your own functions.
DEFINT *DEFSNG* *DEFDBL* *DEFSTR*	DEFINT A-M	Declares variable types as integer, single precision, double precision or string.
DIM	DIM TABLE(25)	Sets the maximum number of entries in an array.
END	END	Ends the program.
ERASE	ERASE TABLE	Removes an array or arrays from memory.
ERROR	ERROR 210	Defines a new error code in addition to those already in PBASIC.
FIELD	FIELD 1,20 AS A$	Reserves space for variables in a random disk file buffer.
FOR..NEXT	FOR I=1 TO 20 NEXT I	Allows a series of instructions to be repeated a given number of times.

PBASIC INSTRUCTIONS

GET	GET 1,I	Reads a record from a random disk file into a buffer variable.
GOSUB... RETURN	GOSUB 1000	Branches to and returns from a subroutine.
GOTO	GOTO 180	Branches out of the normal sequence to a line number.
IF... THEN... ELSE...	IF X=0 THEN GOTO 10 ELSE GOTO 50	Obeys the instruction following THEN, if the condition is true, but the instruction after ELSE, if the condition is false.
INPUT	INPUT "Answer";ANS$	Allows data to be entered from the terminal after the string is printed.
INPUT#	INPUT# 1,A$,B%	Reads data items from a sequential disk file.
KILL	KILL "*.BAK"	Delete files from disk. (see CP/M command ERA)
LET	LET X%=3 A$="Hello"	Assigns a value to a variable. The word LET may be left out.
LINE INPUT	LINE INPUT C$	Allows a line of up to 254 characters to be input from the keyboard, and assigns the string to a string variable.
LINE INPUT#	LINE INPUT# 1,D$	Reads an entire line of characters from a sequential disk file.
LPRINT	LPRINT "Here I am!"	Prints the string that follows on the printer.
LSET RSET	LSET A$=DESC$	Moves data from memory to a random file buffer. LSET left justifies, RSET right justifies.
MERGE	MERGE "NEWPROG"	Merges the named program file into the current program.

ON ERROR GOTO	ON ERROR GOTO 100	Allows the program to handle errors itself, instead of stopping and returning to the system. When an error occurs, program control branches to the specified line number.
ON..GOSUB *ON..GOTO*	ON COUNT GOTO 20,30,40	Branches to one of several line numbers depending on the value of the variable. If the variable = 1, go to the first line number, if the variable = 2 go to the second and so on.
OPEN "I" *OPEN "O"* *OPEN "R"*	OPEN "O",1,"DISKFILE"	Open a disk file. "I" for a sequential input file, "O" for a sequential output file and "R" for a random input/output file.
OUT	OUT 6,27	Sends a character to the machine output port specified by the first number. The second number is the ASCII value of the character.
POKE	POKE 3056,45	Put a character into memory at location specified by the first number. The second number is the ASCII code of the character.
PRINT	PRINT A$;B$,C%	Prints the values that follow onto the screen.
PRINT#	PRINT# 1,A$,B$,C%	Print the values that follow to sequential disk file number 1
PUT	PUT 2,RECNO%	Writes a random disk file record from a file buffer to disk.
RANDOMIZE	RANDOMIZE 623	Sets the random number generator. The number that follows is a "seed" in the range 0 - 65529.
READ	READ WEIGHT%,HEIGHT%	Reads values from a DATA statement into the named variables.

REM	REM This is a remark	Allows comments to be inserted into a program.
RESTORE	RESTORE RESTORE 300	Sets a pointer in the DATA statements. If no line number is given, the pointer is set to the start of the first DATA statement, otherwise to the start of the specified line.
RESUME	RESUME RESUME NEXT RESUME 120	Continues the program after an error handling routine called by ON ERROR GOTO. RESUME alone returns to the line which caused the error, RESUME NEXT to the line after, and RESUME line number to the specified line number.
STOP	STOP	Stops the execution of the program. Unlike END, the program can be restarted with the command CONT.
SWAP	SWAP A$,B$	Exchanges the values of the two variables.
WAIT	WAIT 6,32	Suspends program execution until the machine port specified by the first number returns the character whose ASCII code is specified in the second number.
WHILE... *WEND*	WHILE A%=1 WEND	Repeats the enclosed instructions as long as the condition remains true.
WIDTH	WIDTH 75 WIDTH LPRINT 80	Sets the width in characters of the screen display. WIDTH LPRINT sets the width of the printer.
WRITE	WRITE A$,B$,C%	Acts in the same way as PRINT.
WRITE#	WRITE# 1,A$,B$,C%	Writes the values that follow to a sequential disk file.

PBASIC COMMANDS

AUTO	AUTO 100 AUTO 200,5	Sets automatic generation of line numbers starting at the first ndmber and increasing by the second number each time. If either number is left out, it defaults to 10.
DELETE	DELETE 10-50	Deletes the range of line numbers specified.
DIR	DIR B:*.*	Lists the disk directory. The same as the CP/M system command.
EDIT	EDIT 230	Enters edit mode at the named line.
ERA	ERA B:PROGA	Erases files from disk.
LIST	LIST 10-50	Lists the program lines specified on the screen.
LLIST	LLIST 10-50	Lists the program lines specified on the printer.
MERGE	MERGE ADJ	Loads a PBASIC program from disk and merges it with a program already in memory.
NEW	NEW	Clears the current program from memory so that a new program can be entered.
OLD	OLD CONVERT	Loads a PBASIC program into memory from disk.
RUN	RUN RUN CONVERT	Executes the program in memory. Loads the named program from disk and executes it.
SAVE	SAVE CONVERT	Saves a PBASIC program to disk.
SYSTEM	SYSTEM	Leave PBASIC and return to CP/M.

Note – commands cannot be executed from within a program.

PBASIC FUNCTIONS

ABS(X)	ABS(DIFFERENCE)	Returns the absolute value of the argument.
ASC(S$)	ASC(A$)	Returns the ASCII value of the first character of the string argument.
ATN(X)	ATN(3.56)	Returns the arctangent of the argument in radians.
CDBL(X)	CDBL(LENGTH)	Converts the argument to a double precision number.
CHR$(X)	CHR$(47)	Returns a one character string whose ASCII value is the value of the argument.
CINT(X)	CINT(HEIGHT)	Returns the value of the argument rounded to an integer.
COS(X)	COS(ANGLE)	Returns the cosine of the argument, which must be in radians.
CSNG(X)	CINT(VALUE#)	Returns the value of the argument as a single precision number.
CVI(X$) *CVS(X$)* *CVD(X$)*	CVI(I$)	Convert string values to numeric values. These functions are used in connection with random disk files, where numbers are stored as 2, 4 and 8 byte strings. CVI converts to integer, CVS to single precision, CVD to double precision.
EOF(file)	EOF(2)	This function is true at the end of a sequential file, otherwise it is false.
EXP(X)	EXP(A%)	Returns the constant 'e' raised to the power of the argument.
FIX(X)	FIX(-58.75)	Returns the truncated integer part of the argument.

PBASIC FUNCTIONS

FRE(X)
FRE(X$)
	FRE(0)
	I=FRE("A")
		If the argument is numeric, then FRE returns the size of free memory. If the argument is a string then FRE returns the size available for strings.

HEX$(X)	S$=HEX$(345)	Returns a string containing the hexadecimal equivalent of the argument.
INP(X)	I=INP(6)	Returns the character read at machine port X.
INPUT$(X)	S$=INPUT$(1)	Returns a string of X characters typed at the keyboard without echoing them to the screen.
INSTR(X$,Y$)	I=INSTR("Happy Day","y")	Returns the position of the first occurence of Y$ in X$.
INT(X)	I=INT(FRACTION)	Returns the largest integer less than or equal to the value of the argument.
LEFT$(X$,I)	S$=LEFT$(T$,3)	Returns the leftmost I characters of X$.
LEN(X$)	I=LEN(B$)	Returns the length of the argument string.
LOC(file)	I=LOC(1)	Returns the next record number in a random file.
LOG(X)	I=LOG(45/7)	Returns the natural logarithm of the argument.
LPOS(X)	I=LPOS(0)	Returns the position of the printer head. The argument is a dummy.
MID$(X$,I,J)	S$=MID$(A$,4,2)	Returns a substring of X$, length J beginning with the Ith character.

PBASIC FUNCTIONS

MKI$(I%) *MKS$(I!)* *MKD$(I#)*	S$=MKS$(VALUE)	Converts integers, single precision numbers and double precision numbers to binary strings for output to random files.
OCT$(I)	S$=OCT$(345)	Returns the octal equivalent of the argument in a string.
PEEK(X)	I=PEEK(25636)	Returns the ASCII value of the character in memory location X.
POS(X)	I=POS(0)	Returns the current cursor position. X is a dummy argument.
RIGHT$(X$,I)	S$=RIGHT$(B$,4)	Returns the rightmost I characters of X$.
RND(X)	I=RND(6)	Returns a random number between 0 and 1
SGN(X)	I=SGN(DIFF%)	Returns -1 if the argument is negative, 0 if it is zero, +1 if it is positive.
SIN(X)	I=SIN(ANGLE)	Returns the sine of the argument, where the argument is in radians.
SPACE$(X)	S$=SPACE$(8)	Returns a string of spaces of length X.
SPC(X)	PRINT SPC(66)	This function can only be used with a PRINT or LPRINT instruction. Prints X spaces on the screen.
SQR(X)	I=SQR(2)	Returns the square root of the argument.
STR$(X)	S$=STR$(HEIGHT%)	Returns the numeric argument in a string.
STRING$(I,X$)	S$=STRING$(12,"*")	Returns a string of length I consisting of the single character in X$.

PBASIC FUNCTIONS

TAB(X)	PRINT TAB(25)	Tabs to position X on the terminal. TAB may only be used with PRINT and LPRINT.
TAN(X)	I=TAN(ANGLE)	Returns the tangent of an angle expressed in radians.
USR(X)	I=USR(1)	Calls the user's own assembly language subroutine.
VAL(X$)	I=VAL(NUMBER$)	Returns the numeric value of the string in the argument.
VARPTR(NAME)	I=VARPTR(ANS$)	Returns the memory address where the variable named in the argument is stored.

Note that all functions require an argument in brackets

PBASIC ERROR MESSAGES

1	Missing FOR statement
2	Unidentified error
3	Missing GOSUB statement
4	READ - out of data
5	Function call not allowed
6	Number too large
7	Program too large
8	Line number not found
9	Subscript too large
10	Duplicate array
11	Illegal divide by zero
12	Illegal in direct mode
13	Mis-matching data types
14	Should not occur
15	String too long
16	String expression too complex
17	Not in Break Mode
18	Function not defined
19	Should not occur
20	Illegal RESUME statement
21	Should not occur
22	Missing operand
23	Program line too long
50	Invalid FIELD statement
51	Should not occur
52	Invalid file number or name
53	File not found
54	Invalid file mode
55	File already open
56	Duplicate file number
57	Disk I/O error
58	Duplicate file name
59	Should not occur
60	Should not occur
61	Disk full
62	End of file
63	Invalid record number in GET or PUT
64	Invalid filename
65	Should not occur
66	Missing line number

PBASIC ERROR MESSAGES

99	Break
101	Program too large
102	ON statement out of range
103	Missing line number
104	Missing variable
105	Should not occur
106	Line number does not exist
107	Integer too large
108	Invalid input data
109	STOP
110	Too many nested subroutines
111	Invalid BLOAD file
202	Invalid command
203	Missing line number
204	Missing NEXT or WEND
205	Missing FOR or WHILE
206	Missing comma
207	Missing bracket
208	Invalid OPTION BASE (must be 0 or 1)
209	Statement too long
210	Too many arguments
211	Should not occur
212	Cannot redefine variable
213	Duplicate function definition
214	Invalid GOTO
221	Unrecoverable system error
222	Program not run
223	Too many FOR loops