LINK 480Z CASSETTE SYSTEM USERS GUIDE
PN 11684

Preface

The purpose of this manual is to enable you to install and start using the Research Machines 480Z Cassette System. It makes particular reference to the use of cassette storage, BASIC in ROM, and ROM packs. (If your 480Z is equipped with disc storage and you wish to use disc-based software you should refer to the manuals delivered with your 480Z disc drive, in particular the "480Z Disc Systems Users Guide".)

Chapter 1 of this manual is a general introduction to microcomputers and the terminology that you will need to become familiar with. Chapter 2 is a description of the features and facilities of the 480Z cassette system. Chapter 3 explains how to install the 480Z cassette system and how to connect external devices, such as a cassette recorder and a visual display unit.

When your system is correctly installed you can begin using the BASIC language interpreter, as described in Chapter 4. Chapter 5 describes the 480Z Resident Operating System (ROS) facilities and how to select the external devices, including a printer, for input and output operations.

Appendix A explains hexadecimal notation, which you may need to become familiar with as you read through this and the accompanying documentation.

Appendix B is for more advanced users: it gives details of 'escape sequences' that can be used in BASIC to make use of certain 480Z features for controlling the screen display. A list of useful escape sequences is provided to assist you in writing your BASIC programs.

Appendix C contains the 480Z character sets. The first is the normal set of characters available, and the second is the graphics character set obtained when using the 480Z in graphics mode.

Appendix D provides a detailed "map" of the 480Z cassette system internal memory.

Appendix E is a glossary of some of the terms used when discussing microcomputers.

Related Publications

This manual refers to the following publications, which are available from Research Machines:

LINK 480Z BASIC IN ROM REFERENCE MANUAL, PN 11682

— This manual is essential for any user wishing to write BASIC programs for execution under the control of the 480Z BASIC in ROM interpreter.

LINK 480Z INFORMATION FILE, PN 10939

– This manual contains detailed information about the LINK 480Z hardware, especially the physical characteristics of the input/output ports, the high resolution graphics option, and the printed circuit boards.

CONTENTS

Chapter 1
Introduction to Microcomputers

This chapter introduces the basic concepts of microcomputer systems to users with little or no previous knowledge of computer systems.

What is a Microcomputer?

A microcomputer is an electronic machine which processes information rapidly. A microcomputer cannot think for itself; it can only do exactly what it is instructed to do by computer programs.

The term *microcomputer system* generally refers to a number of separate elements which work together to achieve a required result. These elements can be thought of in two main groups; the electronic machines, called *hardware,* and the computer programs, called *software.*

Hardware

A typical microcomputer system consists of:

- A keyboard
- A visual display unit
- A processor
- An internal memory
- An external storage device
- A printer

The *keyboard* lets you communicate with the processor.

The *visual display unit* (VDU) lets you see what you have told the system to do, and what messages the system has sent to you. These messages may include any other information previously stored in the computer system.

The *processor,* as the name suggests, processes your information as instructed by a computer program. The processor includes a microprocessor that executes programs by performing arithmetical and logical operations on information stored in the computer's memory.

This *memory* is mainly 'volatile': volatile memory, usually called random access memory (RAM), can be modified by a program during processing, but when the system is turned off the contents of this memory are lost. Each time the system is turned on the program and the information must be reloaded from a 'non-volatile' storage device.

The internal memory also includes read-only memory (ROM) which is non- volatile; the contents of ROM are not lost when the system is turned off. This memory has special uses within the system.

The size of the memory available, whether volatile or non-volatile, is normally expressed in terms of *Kilobytes*, where one byte comprises eight binary digits (bits) of memory and Kilo (K) represents the value 1024. One byte of memory can be used to represent one character of information. For example, the 480Z is generally equipped with 64K bytes of RAM, this means that the 480Z has 64 x 1024 (= 65,536) bytes or memory locations for storing programs and information.

The *external storage media* are usually a form of magnetic storage. Either magnetic tape (cassettes) or magnetic discs are commonly used as storage media for microcomputers. These media hold the programs and items of data (files) that you will be using or creating. The processes involved are similar to those of a tape recorder, as described later.

Finally, you will often need a printed copy of the results of the work performed by the computer. This is provided by the *printer.* The copy is called a *listing.*

External input/output devices, such as VDUs, printers, and cassette recorders, are usually referred to as *peripheral* devices.

Software

Software really means computer programs. A *program* is a sequence of instructions. There are two types of software:

- System software
- Applications software

System software is a set of programs supplied by the computer system manufacturer. Without these programs, it is an extremely laborious process to write programs for the system. The system software helps to make the computer easier to program and use. The *operating system* is an example of system software. Another example is the system software routines that are held permanently in the computer's non-volatile (Read-Only) memory. This type of software is frequently referred to as *firmware.*

Applications Software is a set of programs which you may use to solve your specific information and data processing problems. These programs are designed to take care of the most commonly-used applications. This saves a lot of time for many users. For example, this manual has been prepared using a word processing application program, called WordStar*, that is available for use on many different microcomputer systems.

* WordStar is a registered trademark of MicroPro. Inc.

Programming Languages

Both system and applications software obviously need to be written in some computer language. Many languages are now available but the most commonly used on the 480Z is BASIC.

BASIC is called a *high-level* language because it is easily written and understood by computer users, especially those with only a little knowledge of computers. Therefore, when you begin to write your own applications program you will probably use BASIC. Many applications programs provided by Research Machines for your 480Z are written in BASIC.

High-level languages need to be translated into the machine code (binary digits) that the computer can read and understand. A language *compiler* or *interpreter* is provided for this purpose, and in the 480Z the BASIC interpreter is built into the machine, being situated in ROM as firmware (hence, BASIC in ROM).

Assembly languages are called *low-level* languages because they are written in a coded form (known as *mnemonic* code) which is very like the internal machine code used by the computer. Their advantage over high-level languages is that they are faster to run and more efficient in their use of the processor and memory. However, the task of programming in an assembly language is much greater than in a high level language, and consequently assembly languages are used normally only where considerations of execution speed and memory space override those of programmer-productivity and ease of program maintenance. Much system software and some applications software is written in an assembly language. The 'interpreter' for an assembly language is known as an *assembler.*

Why 'Micro'?

Computers vary according to their size, complexity, and cost. *Microcomputer* is the name give to the smallest of the general purpose computers that are widely used in industry, education, administration, and, increasingly, the home. Microcomputers possess many of the features of the larger, more complex, and more expensive computers. However, microcomputers are assembled in small cabinets that are easily transported. They fit easily into the existing office, laboratory, classroom, and home, use a small amount of power, and cost substantially less than larger systems.

The most significant differences between a microcomputer and its larger cousins are its much smaller capacity for storing information and, usually, a lower speed for processing this information. In most other respects they provide the same facilities.

Chapter 2 Introducing the LINK 480Z Cassette System

In this chapter we shall look at the 'make-up' of the 480Z, beginning with the hardware and going on to consider the operating software and finally how the internal memory is organized.

Before reading this chapter, remove the equipment from its packing. You are shown how to install the equipment in the next chapter.

The Hardware

The Research Machines 480Z is built into a moulded structural foam case which houses the printed circuit boards, power supply, and the keyboard. A VDU (video monitor or a domestic television set) and a cassette recorder are connected to the 480Z to make up a basic system, as shown in Figure 2.1. Other 'peripheral' devices, such as a printer or a ROM pack, can be connected as and when required.

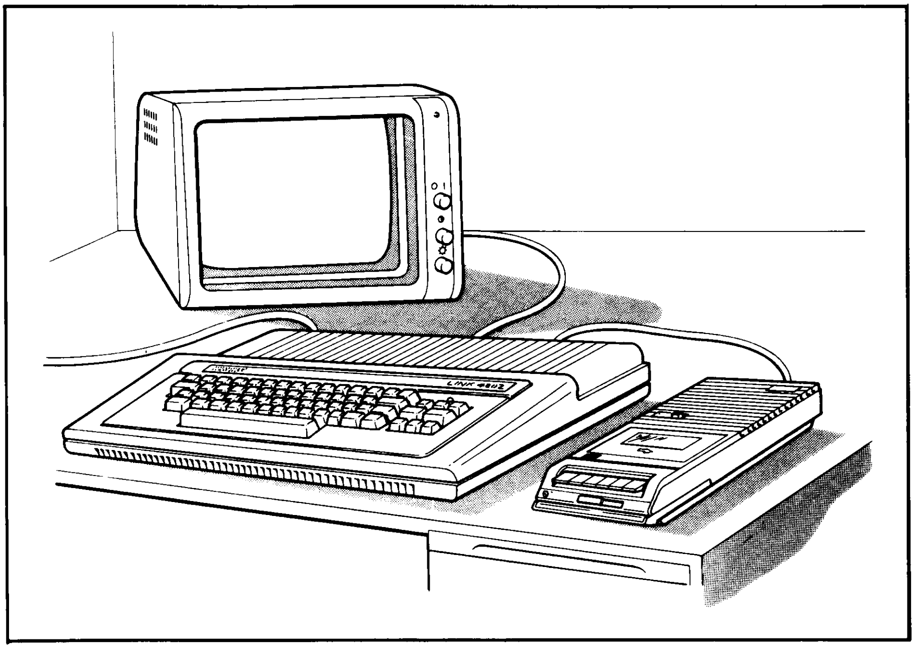


Figure 2.1 Basic 480Z System

Early 480Z machines were housed in a black metal case. If you encounter this version, don't worry; it functions identically to the later machine housed in the moulded structural foam case. However, there are some important differences that you should be aware of, and these are:

- Memory: A number of early models were supplied with 32K bytes of random access memory
- BASIC in ROM: A number of early models were not fitted with BASIC in ROM
- 80-Character Option: All machines are now fitted with the 80-character option allowing the monitor text display as either 40-character lines or as 80-character lines. Some earlier machines will only permit a 40-character line display
- HRG: If your 480Z is fitted with the High Resolution Graphics (HRG) feature, it will be possible to produce high resolution graphics output on the monitor. Otherwise you will be limited to low resolution graphics output. You can tell if your 480Z is equipped with this feature by checking the messages displayed when you first start using BASIC in ROM (described in chapter 4)

  If your 480Z has the HRG feature, you will also be able to produce colour graphics output provided that you connect a colour monitor or colour TV able to process TTL/RGB signals.

The 480Z Keyboard

Figure 2.2 shows the keyboard in closer detail.

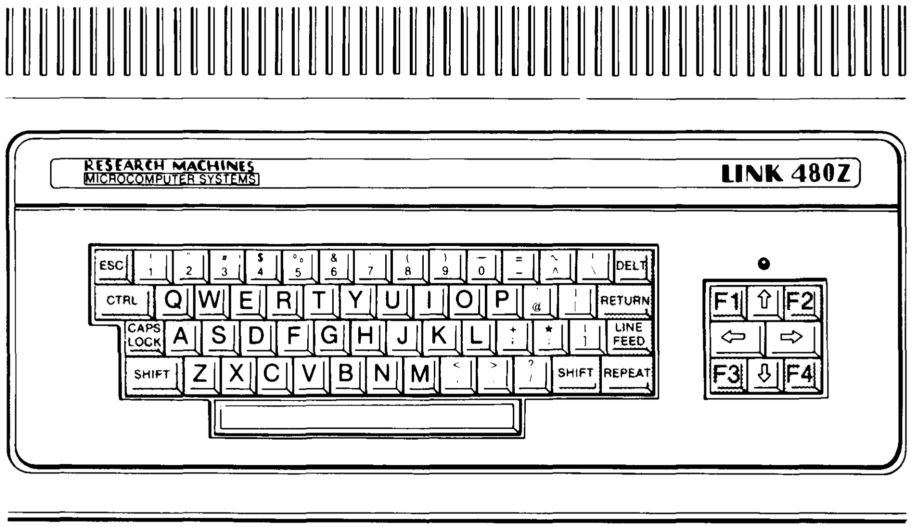


Figure 2.2   480Z Keyboard

The keyboard holds two separate 'blocks' of keys (or keypads), one large and one small.

The large keypad consists of alphabetic, numeric, and symbol characters. Some keys have only one character printed on them but others have two, and this is explained when we look at the *control* keys in the next section.

The long key at the bottom of the keypad is called the *space bar* and is used to move the cursor across the VDU screen by entering 'space' characters The cursor is the square that marks the position on the screen where any character you enter will appear.

The small keypad contains eight keys (F1, F2, F3, F4 and 'arrow' keys) which are usually referred to as *function* keys. This is because they can be programmed by you to perform particular functions, such as screen editing. The use of the function keys depends on the application program you happen to be running. Their functions are different, for example, when programming in BASIC to when using the text editing facility, TXED. You can use 'Escape Sequences' to define what a particular key should do, and these are introduced in Chapter 4.

Keypad overlays are used to indicate the functions that each key will perform; these are plastic overlays which slip easily over the eight keys. You should read the appropriate manuals for the software for which keypad overlays are available (BASIC, TXED, etc.) to find out more about their use.

Control Keys

The use of the keyboard is demonstrated in chapter 4, but here we shall look at the important *control* keys and the way they operate. These keys are darker in colour than the other keys to make them easily identifiable. When you have finished this chapter, and have set up your system as shown in chapter 3, you should come back to this section and practise using these keys:

$\boxed{\text{CTRL}}$

The CONTROL key is used together with other keys to give commands. Hold this key down while pressing the key you require. For example, pressing CTRL and F (you will see this written as CTRL/F) puts the 480Z Front Panel onto the VDU screen. The Front Panel is described in chapter 5 (ROS facilities). You will be introduced to other control commands as you work through the manual. Full descriptions of all possible control commands are given in the 480Z BASIC in ROM Reference Manual.

$\boxed{\begin{array}{c}\text{CAPS}\\\text{LOCK}\end{array}}$

When pressed down and 'locked' this causes all alphabetic characters to be displayed in upper case; otherwise they are displayed in lower case.

DELT	This is the DELETE key and when pressed causes the character immediately to the left of the cursor to be deleted. The number of characters deleted is equal to the number of times the key is pressed.
RETURN	You will see this key mentioned many times throughout this manual. It is used when you enter commands and when you are writing and running programs. It is the way of telling the computer that you have entered something on the keyboard and you now want it to obey a command or store some information. When you see **RETURN** written after a command it means simply 'press the RETURN key'.
REPEAT	Press a key then press the REPEAT key. You will see that the character is repeated until you release the REPEAT key. If you hold down the space bar and press **REPEAT** you will move the cursor across the screen.
SHIFT	Hold down this key and press another key: if it is a letter key, the letter will be printed in upper case (this is useful if you don't want to keep the CAPS LOCK key down). If it is a double symbol key, the **top** symbol will be displayed. It can also be used, in combination with other keys, for giving commands and this will be explained in the relevant part of the user guide.
LINE FEED	Pressing this key moves the cursor vertically down the screen to the next line. It is usually only used when running programs and will be explained in the relevant documentation for the programs concerned.
ESC	Like the line feed function, the ESCAPE key usually serves a special function within programs and will be explained in the relevant documentation for the programs concerned.

We shall now look at the software that controls all the activities of the 480Z.

The Software

ROS, the 480Z Operating System

The programs that make up the Resident Operating System, ROS, are permanently stored in read-only memory and are therefore termed 'firmware'. These (system) programs are not affected by any other (applications) programs you may run.

ROS controls the internal operations of the computer and was designed to:

- Perform certain functions you request, from commands you enter at the keyboard
- Control the operation of the 'peripheral' devices connected to the 480Z, as well as the keyboard
- Look after the way the 64K bytes of 480Z internal memory are used by programs so that you do not have to be concerned with what is happening inside the machine
- Allow you to write and check low-level (machine code) programs
- Activate the BASIC in ROM interpreter (as described later in the manual) when you want to run BASIC
- Detect the presence of a ROM pack plugged into the back of the 480Z and take the appropriate action, if required

ROS starts up automatically when the 480Z is switched on or when the reset button on the rear panel is pressed. The facilities that ROS provides are fully described in chapter 5.

Internal Memory of the 480Z Cassette Systems

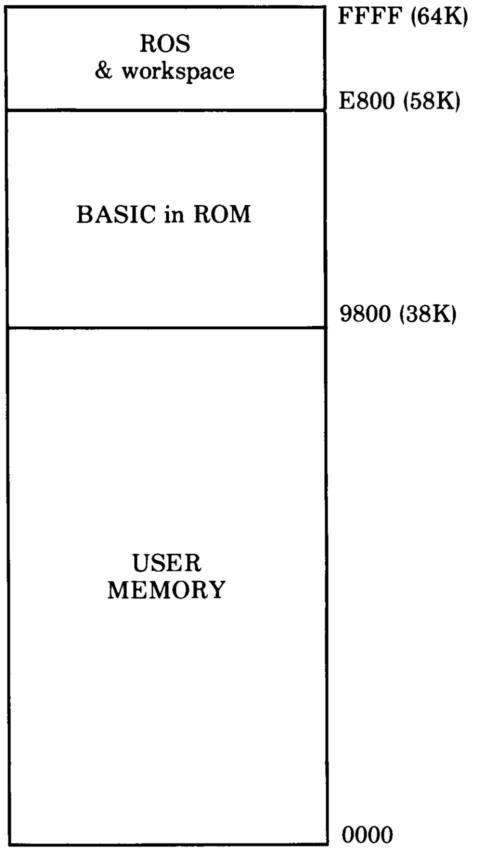


Figure 2.3 480Z Cassette System Memory Usage

The 480Z is generally provided with 64K bytes of internal memory. Figure 2.3 shows, in a simplified form, how this memory is shared between the various system software components such as ROS and BASIC in ROM, as well as the user programs.

User memory, as the name suggests, is the random access memory (RAM) available for you to put data and programs into. With the 480Z this is 58K; but if you are running BASIC in ROM, the BASIC in ROM interpreter prevents the use of some of this memory and leaves you with up to 36K for your BASIC source programs and data.

The remaining 6K bytes of memory are reserved for ROS and includes some area to be used as 'workspace'.

If you want more details of the memory map see Appendix C, or refer to the 480Z Information File.

Chapter 3
Installing the 480Z Cassette System

This chapter explains how to install the 480Z and how to connect the VDU and the cassette recorder.

Checking the Equipment

When you unpack the 480Z, check that you have received:

- The 480Z microcomputer
- A cassette containing some demonstration programs
- A cassette containing the TXED (Text Editor) program, if ordered
- A connecting lead for a UHF television

Other equipment and their accessories may also be included according to your requirements. However, to operate your 480Z you will need at least the following:

- A VDU for black and white (monochrome) display and which can be either:
  - — a UHF television (either colour or monochrome)
  - — a monochrome character display monitor
- A cassette recorder (and its mains lead) and a connecting lead
- A 3-way or 4-way mains adaptor
- 3-pin plugs to connect the above items to the mains supply.

If your 480Z is fitted with the High Resolution Graphics (HRG) feature, it is also capable of generating colour displays through the socket marked TTL/RGB. For this you will need a colour monitor (or a colour television) with the TTL/RGB capability.

More than one type of VDU can be connected simultaneously to a 480Z, i.e. UHF TV, monochrome monitor, and TTL/RGB monitor. All will produce the same display.

The mains lead on the monitor supplied by Research Machines may already have a continental plug moulded to it. Please cut this plug off and attach a standard plug, taking all safety precautions.

Please keep all the boxes and packing, if possible. In the event of the equipment being transported for any reason, this packing will help prevent damage to it.

Installation

When you install your 480Z, you should place it on a convenient work surface close to a mains power outlet. A suggested layout is to place the keyboard in front of the display unit. Other items of equipment, such as cassette recorder or a printer, should be placed as close as possible to the 480Z to enable you (or the operator) to be able to reach them easily. Do not forget to provide enough work surface for papers and manuals.

The 480Z and its accessories should not be exposed to direct sunlight. The ventilation slits on the rear panel should not be obstructed; also, the sockets on the rear panel of the 480Z should remain accessible if you intend to connect and disconnect external devices from time to time without disturbance.

Connecting External Devices (Peripherals) to the 480Z

In a basic system (sometimes called a basic configuration) the peripherals include a VDU and cassette recorder, as mentioned earlier. A printer may also be added when needed. Before connecting them to your 480Z, note these important points:

- All the equipment should be connected to the same mains adaptor or extension lead
- The mains supply to the adaptor or extension lead **must be switched off**
- You should refer to the diagrams of the 480Z rear panel and monitor for guidance. These diagrams are shown in figures 3.1 and 3.2

Visual Display Unit

This can be a monochrome or colour monitor, or a domestic television set. The connections to the 480Z are different for each.

Instructions for installing the Panasonic monochrome monitor are provided in the Installation Notes for the Panasonic Monitor (PN 11293), which are included with the monitor. A separate cable is also provided.

- Connect one end of the cable to the VIDEO IN socket at the rear of the monitor

- Connect the 6-pin DIN plug at the other end of the cable to the MONITOR socket on the rear panel of the 480Z

- Set the Hi-Z/75Ω switch on the monitor to the 75Ω position

The procedure for other monochrome monitors will be similar.

If you are using a Microvitec colour monitor, you will have been supplied with a separate TTL/RGB cable:

- Connect the 6-pin DIN plug to the INPUT socket on the rear of the monitor

- Connect the 8-pin DIN plug to the TTL/RGB socket on the rear panel of the 480Z

If you are using a television set (either black and white or colour), you need the standard coaxial cable with a plug at each end, as supplied.

- Connect one end to the aerial socket of the TV set

- Connect the other end to the TV socket on the rear of the 480Z.

If you are using a colour television with TTL/RGB facilities, it should have been supplied with a cable with a plug appropriate for connecting it to the 480Z.

A suggested 'switching on' procedure is given later in this chapter.

Cassette Recorder

The cassette recorder supplied with your 480Z is a SANYO DR101. Future releases of 480Zs may include other makes of recorder, depending on the availability of supplies at that time. If so, this will be reflected in later versions of the user guide.

The recorder is used, to begin with, for reading (loading) programs from cassette tapes; later on it can be used for storing programs you have written (or modified) so that you can run them again. Chapters 4 and 5 explain how to use the demonstration cassette supplied with your 480Z, and chapter 4 also describes how to store your programs.

To set up the recorder, use the cable supplied with it (marked 'cassette cable'). Note that the cable has a 7-pin DIN plug at one end, and two 3.5mm jack plugs at the other. The jack plugs may be one of two types:

1. Both plugs are coloured grey, one labelled M and the other labelled E.

2. One plug is black (labelled M), the other is white (labelled E).

The use of both types is identical:

- Connect the jack plug labelled M (or coloured black) to the MIC socket on the recorder and the plug labelled E (or coloured white) to the EAR socket.
- Connect the 7-pin DIN plug at the other end to the socket marked 'cass' on the rear of the 480Z.
- Set both the mode switch and the phase reversal switch to their NOR (normal) positions. (The phase reversal switch should **not** be pushed in)
- Turn the volume control to a position more than half way to its maximum value.

Note that a muting plug is not required.

If you need a detailed description of the hardware interface to a cassette recorder, refer to chapter 3 of the manual: '**LINK 480Z Information File**'

Switching On the System

When the system has been set up, as described above, you can now switch it on. Before you do so, make sure that the power (ON/OFF) switches on the 480Z and the VDU are turned off. The steps to follow are:

- Plug in the mains extension lead or adaptor to the mains and switch on the mains supply
- Switch on the VDU and allow it to warm up for a few seconds
- Switch on the 480Z using the mains ON/OFF switch on the rear panel.

  The red LED (Light Emitting Diode) above the right-hand keypad on the keyboard will light up.

  (If you are using a television you will also have to select and tune one of its channels to the 480Z's TV output signal and set the volume control to a minimum.)

  The 480Z ROS start-up message will appear like this on the screen:

```
RML 80 Character LINK 480Z            V1.2x
Z-NET Firmware Vers 1.1x           Address 00

To start BASIC in ROM type the command R

Please give a command or type H for help

}
```

- Adjust the brightness and contrast controls on the VDU until the picture is clear and sharp.

- The message **'Z-NET firmware'** shows that your 480Z has a network transceiver board fitted to it and can be used as a network station in a Research Machines network. (More information on networks is available in the network guides, supplied by Research Machines to network users.)

The use of BASIC in ROM is described in Chapter 4.

The command H displays a list of the commands that ROS will accept. These are described in Chapter 5.

The symbol '}' is the ROS monitor program prompt to tell you that the system is waiting for you to enter a command. It is followed by a bright square, the 'cursor', which moves across the screen as you key in information. Your 480Z is now ready for use.

You should now go back to chapter 2 and practise using the keyboard and the control keys. Then you can go to chapter 4 and start using the BASIC interpreter.

Connection Information

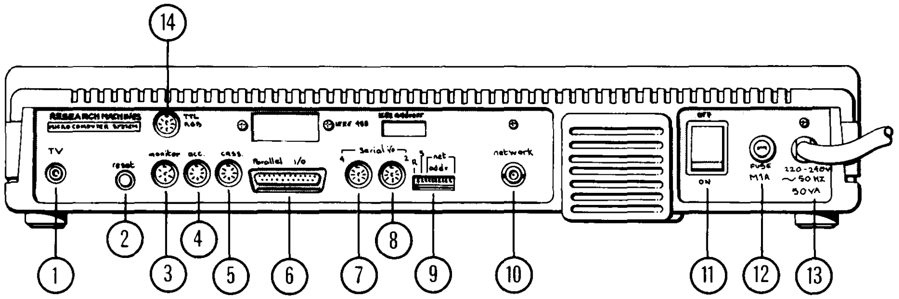


Figure 3.1 480Z Rear Panel

1. Modulator UHF (TV)
2. RESET Button
3. Video (Monitor)
4. Accessories
5. Cassette
6. Parallel I/O
7. Serial I/O 4
8. Serial I/O 2
9. Network Address, Speaker, and RESET Disable Switches
10. Network Interface Coaxial
11. Mains ON/OFF
12. Fuse
13. Mains Cable
14. TTL/RGB Display

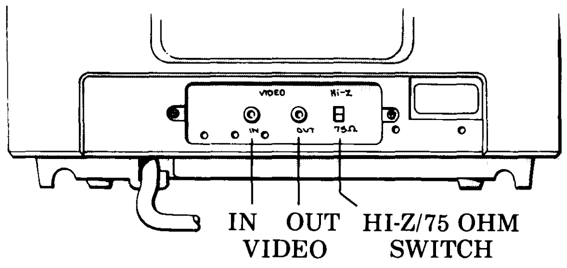


Figure 3.2 480Z Monitor

Chapter 4
Using BASIC on the 480Z Cassette System

This chapter discusses the BASIC in ROM interpreter, how to run the programs on the demonstration cassette, the use of ROM packs, and, finally, the use of 'Escape Sequences' from BASIC. It also includes a section on how to recover from errors occurring while using BASIC.

The version of the BASIC language supported by BASIC in ROM is described in the manual: **"LINK 480Z BASIC in ROM Reference Manual"**. This manual is supplied with each 480Z cassette system. Although it is not the intention of this manual to teach BASIC programming, as this is adequately covered in a number of books written for that purpose,* it both introduces BASIC to newcomers and provides reference information to help experienced BASIC programmers use BASIC in ROM. This manual is also supplied with a quick reference card that summarizes the main features of this version of BASIC.

*Some useful BASIC programming texts are:

"Illustrating BASIC", by Donald Alcock (Cambridge University Press),
"Computing Using BASIC", by Tonia Cope (Pub. Ellis Horwood Ltd.)

BASIC in ROM Interpreter

BASIC in ROM, the BASIC interpreter, is provided in read-only memory to simplify and speed-up the loading of BASIC on the 480Z.

Before proceeding with this chapter, make sure that you have worked through chapter 3 and set up your system correctly. You should now have your system switched on and ready.

To start with, the monitor screen should display the ROS start-up message (described in chapter 3).

To activate BASIC in ROM press **R**. As soon as BASIC in ROM is activated, the screen displays a message similar to:

```
RML Extended BASIC in ROM        V5.4x
Copyright (c) 1983 by Research Machines
High Resolution Graphics Level 2  V1.1x
Ready:
```

[Note, however, that high resolution graphics (HRG) is an option on the 480Z. If your machine does not have the HRG option fitted, the message "No HRG option installed" is displayed before the "**Ready:**" prompt.]

The 480Z is now ready for you to begin using BASIC. As an introduction to loading and using BASIC programs you should use the demonstration cassette supplied with your

480Z. The cassette contains three programs: DEMON, WRITER, and TUNE. WRITER and TUNE are both machine-code programs and are described in chapter 5 (ROS facilities)

DEMON is the only BASIC program of the three. To load and run the program, follow the instructions given in the next section for loading BASIC programs from cassette.

Loading BASIC Programs From Cassette (LOAD Command)

The demonstration program DEMON is used as an example here. Remember that where you see the name DEMON you should replace it with the name of the program you are using if it is not DEMON.

Remove your demonstration cassette from its plastic holder, and follow these steps carefully:

1. Make sure that you have set up the cassette recorder as shown in chapter 3.

2. Place the cassette in the recorder with Side A facing **upwards.** Press the REVIEW/REWIND key to ensure that the tape is correctly wound to its start position.

3. At the keyboard enter:

   LOAD "DEMON" $\boxed{\text{RETURN}}$

   It is important to remember the pair of quotation (" ") marks around the program name.

4. Press the PLAY key on the recorder.

The 480Z now begins to search for the start of the program DEMON and displays the message:

Searching

on the screen.

When the start of the program "DEMON" is found, the 480Z displays:

Searching DEMON 0000

The program is recorded on the tape as a series of numbered blocks; these blocks are loaded in sequence starting at the first block that the computer finds.

The number 0000 indicates that the beginning block of the program has been identified and is being loaded into the 480Z user memory area.

When the whole block has been loaded the display changes to:

```
Loaded    DEMON 0000
Searching DEMON 0001
```

The block numbers change continuously as each block is found and loaded so that, for example, the next display shows:

```
Loaded    DEMON 0001
Searching DEMON 0002
```

and so on, until the whole program has been loaded.

When the whole of the program has been loaded into memory, the screen shows the prompt:

Ready:

5. Press the **STOP** key on the recorder.

   (It is also possible to interrupt loading of the program as explained below.)

   You are now ready to run the program using the **RUN** command.
6. Enter **RUN** RETURN.

To exit from the program, enter **CTRL/C**.

Interrupting Program Loading

You may wish, for some reason, to halt the loading of your program from a cassette tape. To do this, enter **CTRL/Z**; a message will then appear on the screen to tell you that loading has been interrupted.

If you wish to begin loading the same program again you must rewind the tape to the required position. For example, if you interrupt loading the program DEMON you must rewind the tape to the physical start of the tape. Then begin the loading procedure from Step 3, as described previously.

Errors in Program Loading

It is possible for an error to occur while you are reading a program from the cassette. BASIC in ROM indicates a read error with the message:

Read error — — A=Accept, B=Abort, R=Retry

In the majority of cases you will want to try again to read the block of program where the error occurred and the way to do this is to:

- Press the STOP key on the recorder.
- Rewind the tape sufficiently to position it before the start of the block in which the read error occurred.

  BASIC in ROM remembers which block was being read when the error occurred and will begin searching for that block, ignoring any blocks that it has already read successfully.
- Enter R (for Retry) and press the PLAY key on the recorder.

  BASIC in ROM will attempt to read the block again.

Note that entering **A** will cause the faulty block to be loaded, and the display will indicate this for you by placing an 'e' in front of the program name. For example, if block 0002 of our demonstration program DEMON was in error the display would show:

Loaded eDEMON 0002

If you enter **B**, this simply stops further loading of the program.

Saving Programs On Cassette (SAVE Command)

As well as loading programs from cassette tapes, you will also want to create your own programs and store them on cassette tapes for future use. Having written your program, called TEST1 for example, you follow these steps:

1. Insert a cassette tape into the recorder. If it is a blank tape or a tape that contains information that can be overwritten, make sure that it is wound past the 'leader' tape.

   If the tape already contains information that must not be overwritten, such as programs or data files, make sure that the tape is wound past the last of these programs or files.
2. Turn the volume control on the recorder to more than half way towards the maximum value.
3. Enter: **SAVE "TEST1"**
   (Do not press **"RETURN"** yet!)
4. Press the RECORD button on the recorder.
5. Press $\boxed{\textbf{RETURN}}$

Important: Remember *always* to press the RECORD button before pressing the **RETURN** key as you may otherwise lose the start of the program.

When the whole program has been saved on the cassette tape, the message:

"**Ready:**"

appears on the screen.

6. Press the STOP button on the recorder.

When you next want to load the saved program, such as TEST1, you simply wind or rewind the tape to the correct starting position and use the LOAD command as described previously.

The LOAD and SAVE commands are also described in the manual: "**LINK 480Z BASIC in ROM Reference Manual.**"

ROM Packs

ROM packs are devices available for connection to a 480Z to speed up the process of loading programs because of the long time delays found with cassette tapes.

This feature is implemented in 480Z systems equipped with BASIC in ROM, version 5.4, or a later version. If your 480Z has an earlier version of BASIC in ROM, you can obtain the required version from Research Machines.

A ROM pack can contain a number of files. These files can be one of the following:

- A single machine-code program
- One or more BASIC programs (and any machine-code routines which are to be added to the BASIC in ROM interpreter)
- One or more data files to be used by a BASIC program.

Use of ROM packs with both BASIC and machine-code programs is explained later.

Many applications programs supplied in ROM packs have a BASIC "menu" program as the first file. This program tells you about the other programs in the ROM pack. It may also help you to access the programs more easily as well as showing you the commands you must enter to use them. To find out what is available you should follow the manual supplied with the ROM pack concerned.

A ROM pack is shown in figure 4.1. It consists of a plastic case about the size of a tape cassette case and is fitted with a 25-pin plug at one end; inside the case is a printed circuit board on which are mounted a number of components and one or more EPROM (Erasable Programmable Read-Only Memory) integrated circuits.

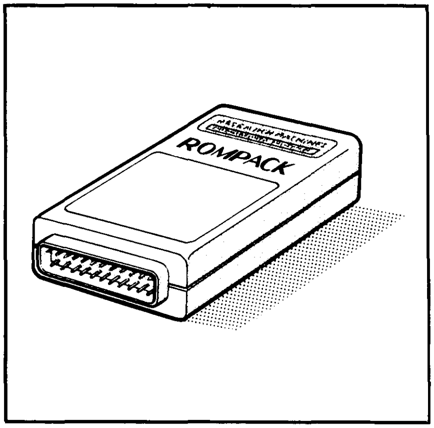


Figure 4.1 A ROM Pack

You cannot alter the programs held in the ROM, nor can you store other data in it. ROM packs are simply external storage devices that cannot be written to but which hold frequently-used programs. The only way to get a new set of programs and files is by removing the EPROM integrated circuits and replacing them with different ones.

Research Machines provides documentation giving more details on the preparation and use of ROM packs. This describes how your own programs and files can be inserted into a ROM pack which has no software in it.

Plugging in the ROM Pack

The ROM pack is designed to plug into the parallel I/O socket on the rear of the 480Z (see figure 3.1).

Using The ROM Pack for Machine-Code Programs

If the ROM pack contains a machine-code program, it is used under the control of ROS, as follows:

- When the 480Z is switched on or when the RESET button is pressed, ROS checks for the presence of a ROM pack with a machine-code program stored in it. If one is present, ROS automatically loads it and starts executing it.

Using the ROM Pack for BASIC Programs

If the ROM pack contains one or more BASIC programs, it is used under the control of BASIC in ROM in one of two ways, as described below.

1. When the 480Z is switched on, BASIC in ROM checks to see whether a ROM pack is already connected. If a ROM pack is connected and it contains one or more BASIC programs, then BASIC in ROM will load and run the first (or only) program automatically (this is the same as using the LOADGO command in BASIC).

The same process occurs if you press the RESET button while a ROM pack is present in the parallel I/O socket.

2. If no ROM pack is connected when you switch on, or press the RESET button, and you wish to load a program from a ROM pack, the procedure is:

   i. Connect the ROM pack to the parallel I/O socket.

   ii. Press **R** to activate BASIC in ROM.

   BASIC in ROM again attempts to load and run the program.

If you have loaded a BASIC program which is going to read a file from a ROM pack that is not yet connected, simply connect the required ROM pack. Do not switch off or reset the 480Z.

When you want to load a BASIC program other than the first one in the ROM, you can do so even if you have a program already loaded:

- Enter: **CTRL/Z** to stop the BASIC program being executed.
- Check that the required ROM pack is connected. Enter the LOADGO command together with a channel number (#8) and the program name, for example:

```
LOADGO #8, "PROG"
```

where #8 is the input/output channel number assigned in BASIC in ROM to the port for the ROM pack.

These procedures are fully explained in the "**LINK 480Z BASIC in ROM Reference Manual**". This manual also explains how to access data files as well as large BASIC programs held on two or more ROM packs.

Restarting BASIC after an Error

It is important that if something goes wrong while you are running a BASIC program you should be able to 'recover' your program without losing whatever data is in the memory. An example is when the system does not respond to input from the keyboard. In cases like this, you should proceed as follows:

1. Hold down the CTRL and SHIFT keys and enter 8. This displays the ROS prompt '}'.

2. Enter: **C**

   This is the ROS command to continue a system program (in this case BASIC in ROM). The screen displays the message:

```
Recovered
Ready:
```

which shows that BASIC in ROM has been reactivated and that the program previously running is ready to run again.

3. Enter: **RUN** RETURN

The contents of memory have not been affected and the program is executed from the beginning.

If this procedure does not work, try this:

1. Press the RESET button on the rear of the 480Z.

   This takes you back to the original ROS prompt, '}'.

2. Enter: **J**

   J will be displayed on the screen followed by the symbol >.

3. Enter: **103** RETURN

   This displays the message:

   Recovered
   Ready:

Escape Sequences

When writing your own BASIC programs you may want to use instructions or commands to make available particular facilities in the 480Z hardware. For example, you may want to define the function of each of the 'special function' keys on the keyboard. You may also, for example, want to switch the screen display from displaying 40 characters per line to 80 characters per line (or vice versa).

An escape sequence is a series of characters sent to ROS by a special form of the BASIC command PUT to do these things for you.

The 'escape' character is sent first and indicates to ROS that the characters that follow it are command characters and are not to be displayed on the screen but are to be interpreted by ROS to perform a specified function.

The internal representation of characters in the computer is in the form of ASCII codes. (A table of ASCII codes is available in Appendix 1 of the "**480Z Information File**", should you require it). The ASCII code for the escape character is 27, so the BASIC statement:

PUT 27, escape sequence

(where "escape sequence" is a list of command characters)

would send the escape sequence to ROS.

Other 480Z features accessed through using escape sequences include:

- activating the beeper and setting the tone or duration of the 'beep'
- redefining the display scrolling window

For details of escape sequences, and a complete list of those available, see Appendix B.

Chapter 5
ROS Facilities

The version of the ROS firmware (1.2) in your 480Z can support the use of floppy discs. However, this is beyond the scope of this manual and you should contact Research Machines for further details.

ROS Commands

ROS has a number of useful features that can be controlled by means of commands entered at the ROS command level.

When the 480Z is switched on, or reset, the ROS start-up message is obtained as described in chapter 2.

The last part of the message requests:

```
Please give a command or type H for help
}
```

Press H to obtain the ROS 'command menu' which lists the commands that are available. The menu looks like this:

```
ROS Commands
        N   Boot network
        B   Boot disc system
        R   Run BASIC in ROM
        W   Switch the character display between 40 and 80 characters
        L   Load a system program from cassette
        D   Dump a system program to cassette
        O   Select the cassette speed and printer options
        C   Continue with the current system program
        J   Jump to an address in memory
        T   Enter terminal mode
   CTRL/F   Enter Front Panel
   CTRL/T   Enter printer test mode
```

ROS requires you to enter one of these commands after the '}' prompt. Some of these are described in greater detail in later parts of this chapter and, if so, are indicated by (*) in the list on the next page:

Command	Action
N	Only used if the 480Z is to act as a 'station' on a Research Machines network.
B	Only used if the 480Z has floppy disc external storage. This is covered in the 480Z disc system guide.
R	Activates BASIC in ROM, as described in chapter 4.
W*	Changes the number of characters per line on the screen from 40 to 80, or from 80 to 40 depending on the current mode. (This has no effect on a 40- character only machine).
L*	Loads a machine-code program from a tape cassette.
D*	Writes a machine-code program onto a tape cassette.
O*	The 'options' instruction allows you to select a printer or to set the cassette speed when using non-standard cassettes.
C	Returns you to executing a system program (such as BASIC in ROM) after an interrupt.
J*	Jumps to a specified memory location and executes its contents . An example of the use of J is given in the section 'Restarting BASIC after an Error' in chapter 4. It is also used when in Front Panel mode, described later.
T	Enables the 480Z to act as a simple terminal when connected to another computer, such as when using a network.
CTRL/F*	Displays the Front Panel on the VDU screen. See '480Z Front Panel', below.
CTRL/T*	Tests the printer connected to the 480Z. See 'Testing a Printer', below.

We shall now look at some of these ROS options in more detail. They are not, however, described in the order given in the list, above.

Loading Machine-Code Programs from Cassette (ROS Command L)

In chapter 4 we discussed the loading of a BASIC program from the demonstration cassette.

Two other programs, WRITER and TUNE, are recorded on this cassette (in machine-code format) and are used to demonstrate how to load machine-code programs as well as showing further uses of the keyboard. These programs follow the BASIC demonstration program DEMON, discussed earlier in chapter 4.

If you have just finished running a BASIC program, reset the 480Z and proceed as below:

Program WRITER

Load WRITER as follows:

- Enter ROS command: **L**

  ROS replies with:

  NAME>

- Enter: **WRITER** $\boxed{\textbf{RETURN}}$ and press the PLAY key on the recorder.

  ROS searches for WRITER and when it finds it replies by displaying the name WRITER again and the number of the block being loaded.

As soon as the program WRITER is loaded, it will start executing automatically.

Remember to press the STOP button on the recorder.

A message like this appears in the middle of the screen:

Press arrow keys to move cursor.

Type CTRL/T to go to top left and clear

Type CTRL/C to exit

Type CTRL/I to switch between insert and overwrite mode

Type CTRL/S to start the program again

- Enter **CTRL/T.**

  The screen goes blank and the cursor is placed in the top left hand corner. You can now key in some text of your choice. This will give you practice at using the control keys.

  The four arrow keys on the right hand square keypad can be used to move the cursor around the screen in the directions indicated by each arrow.

  This program is now in 'insert' mode, which means that you can insert new text into the text you have already keyed in. To do this, proceed as follows:

- Move the cursor, using the arrow keys, to the position where you want the new text to be inserted.

By placing the cursor on a character or space, the new text will be inserted directly to the left of that character or space; existing text will move right to accommodate the new text.

- Enter the new text as required.

You might try to replace a portion of old text with new text instead of simply adding to it. The procedure for this is:

- Move the cursor to the starting position where the text is to be replaced.

  For example, if you had keyed in 'CONPUTER' instead of 'COMPUTER', move the cursor until it overlies the letter N which needs to be replaced.

- Enter **CTRL/I.**

  This takes you into 'overwrite' mode which is indicated by the cursor flashing on and off.

  Enter the text you require.

  In the example you would enter the letter M which would overwrite the incorrect letter N.

Remember that while the cursor is flashing any text you enter will successively overwrite the text originally occupying the cursor's position. It is quite safe, however, to use the arrow keys to move the cursor to any position on the screen while in overwrite mode.

- To return to insert mode, enter **CTRL/I** again.
- If, at any time, you want to start again from the beginning of the program enter **CTRL/S** which gives you the start message again.
- To leave the program and return to the ROS monitor prompt, enter **CTRL/C.**

Program TUNE

TUNE is another machine-code program and is loaded in the same way as WRITER, so the same procedure should be followed, remembering to position the tape at the end of the WRITER program if you haven't already done so.

When loaded, this message appears in the middle of the screen:

RESEARCH MACHINES

LINK 480Z

TUNE a musical keyboard

White notes row A to line feed
Black notes appropriate keys on
row above.

The keyboard, or at least part of it, can now be used as a musical keyboard. The keys on the row between A and the LINE FEED key (inclusive) correspond to white notes. The keys on the row above, Q to RETURN (inclusive) act as the black notes.

Storing Machine-Code Programs on Cassette (ROS Command D)

Programs written in machine code, like systems programs, are recorded onto cassette tape using the following procedure:

- At the '}' prompt, enter: **D**

  ROS responds with:

  NAME>

- Enter the name of the program you want to store (*do not* put it in quotes) and press **RETURN**

  ROS responds with:

  First>

- Enter the number 100 (the value in hexadecimal of the *first byte* in memory of the program) followed by **RETURN**

  ROS responds with:

  Last>

- Enter the value of the *last byte,* (in hexadecimal) in memory of the program that is to be stored. This can be obtained from the relevant documentaton for the program concerned.

  Press **RETURN**

  ROS responds with:

  Start>

- Enter: **100**

  Place a blank tape on the recorder and make sure it is wound past the leader tape. Press the RECORD button.

- Press **RETURN**

  ROS replies with the numbers of the cassette tape block it is storing. When finished the ROS monitor prompt, '}' is displayed.

Selecting a Printer for the 480Z (ROS Command O)

There are four options to choose when using a printer on the 480Z. These options are numbered 0, 2, 3 and 4:

0	is the VDU screen
2	is the serial I/O 2 interface
3	is the parallel I/O interface
4	is the serial I/O 4 interface.

ROS will ask which printer you require, as described later.

This section assumes that you are using one of the printers supplied by Research Machines. It does not describe how to connect the printer to the 480Z, as this is covered in the manuals supplied with the printer, but assumes that it is correctly connected, switched on, and 'on line'.

The printer you use will be either a *serial* printer or a *parallel* printer. A serial printer receives signals from the computer sent one after another along the same wire; a parallel printer receives eight signals, sent down eight separate wires simultaneously. The connections for both types are as follows:

If you are using a serial printer, such as a MICROLINE 80, it must be plugged into the socket on the rear of the 480Z marked 'serial I/O 2' or 'serial I/O 4'.

The options for these are numbers 2 and 4 respectively.

If you are using a parallel printer, such as an ANADEX, it must be plugged into the socket marked 'parallel I/O'.

The option for this is 3.

Setting the printer option can be done from ROS command level before starting your work or from the Front Panel (which is described later) if you are already running a program. At the '}' prompt the steps to follow are:

- Enter ROS command: **O**

  ROS responds with:

  Cassette or lineprinter (C/L):

- Enter: **L**

  The display now asks:

  Printer type (0,2,3,4):

  The type is as described at the start of this section.

- Enter a number as requested.

  If you select type 3 the ROS prompt '}' reappears.

  If you select type 2 or 4 ROS replies with:

  Baud code (0-6):

- Enter a baud code chosen from the following list. The printer manual should indicate what baud rate you must select:

code	Baud rate
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600

You now return to the ROS command prompt.

Testing a Printer (ROS Command CTRL/T)

To test if the printer is working correctly, use the ROS CTRL/T command:

- At the ROS prompt '}' enter: **CTRL/T**

  ROS enters a small test routine that allows anything you enter at the keyboard to be printed both on the screen and on the printer.

- Enter some characters and press **RETURN**

If you get no response from the printer, check that it is correctly connected and switched on, and that it is on line and is loaded with paper. If everything is correct, reset the 480Z and try from the ROS command again.

Refer to the printer manual should it still refuse to work.

To exit from the test routine you should press the RESET button.

Using a Printer with a 'READY' Line

Some serial interface printers can receive data faster than they can actually print it. For instance, a Qume Sprint 5 printer can receive data at 1200 baud, the equivalent of 120 characters per second. However, the Qume can only print at an average of 45 characters per second, so some way of stopping the computer from sending data is required. The most appropriate way is a signal sent to the computer. This signal comes from the printer when it wants the computer to stop transmission until it is ready to receive more data (therefore it is referred to as a READY line).

The Research Machines interfaces, types SIO-2 and SIO-4, can both handle a READY line. The connection details are given in the LINK 480Z Information File.

Changing Cassette Speed Option (ROS Command O)

When the 480Z system is turned on or reset, the cassette recorder interface is ready to accept data in a *high speed* format at a transfer rate of 1200 bits per second. Most Research Machines software issue and demonstration tapes are now sent out in high speed format so you do not usually have to change the speed option.

However, at some time you may wish to read programs from a tape which was recorded in low speed format (as TXED and early software issue tapes were). To do this:

- Enter the ROS Command **O**

- Enter C in answer to the prompt.

  ROS replies with:

  Tape speed (SS/FF)

- Enter the required speed. The first letter gives the reading speed and the second letter gives the writing speed.

  For example, SF requests SLOW reading and FAST writing.

Note that your options are:

SS : Slow read and slow write
SF : Slow read and fast write
FS : Fast read and slow write
FF : Fast read and fast write (the 480Z is set to this when switched on)

The cassette speed option remains as chosen until you enter another **O** command or the 480Z is reset.

The 480Z Front Panel (ROS Command CTRL/F)

A special feature of the 480Z firmware is the Front Panel display.

Before reading this section, press CTRL/F at the keyboard and watch the VDU screen. The Front Panel display starts something like this:

(the actual numbers may be different)

```
>   PC  E87F  F7  FD  EF  04  F7  FD  F1  C9
    SP  FEFC  7F  E8  00  00  46  06  FB  03
    IY  DDAF  04  04  04  00  04  04  04  04   Page 1
    IX  FFFF  FF  FE  FE  66  EE  F7  00  00
    HL  03F8  FF  FF  FF  FF  FF  FF  FF  FF
```

The Front Panel display shows the contents of the internal registers (memory locations) at the time that CTRL/F was entered. You need not be concerned with the contents at this stage.

This manual does not discuss the use of the Front Panel in detail, as it is used mainly for correcting machine-code programs, but introduces some useful commands that are available when the 480Z is in its Front Panel mode.

As examples of the use of the Front Panel we shall look at how to select a printer for output, and switching the screen width while you are using BASIC in ROM.

Selecting A Printer Using The Front Panel

It is possible to set up a printer for output while running a BASIC program, without losing the contents of the memory locations.

The procedure is as follows:

- Press **CTRL/F.**

  BASIC in ROM replies with the question:

  ↑F — — are you sure (Y/N):

- Enter: **Y**

The Front Panel is displayed, and at the bottom of the screen the message is:

Ready:

!

- Enter: **O**

  The display now asks:

  Cassette or lineprinter (C/L):

The procedure is now the same as that given in the section 'Selecting a Printer for the 480Z' on page 5.6. However, instead of returning to the ROS prompt you return to the Front Panel prompt '!'.

- Enter **K**

  You now return to the point where you stopped execution of your BASIC program.

Switching the Screen Width

Another use of the Front Panel is to switch the width of the screen between its two possible working modes of 40 and 80 characters per line.

When you first switch on the 480Z, or press the RESET button, the screen width is 40 characters per line. If you want to change the width then you enter the ROS command **W** after the ROS prompt.

However, if you are using BASIC in ROM then you must enter the Front Panel using CTRL/F as described above. Then proceed as follows:

- At the '!' prompt enter W

  ROS responds with:

  80 character

- Enter: **K**

  You now return to the BASIC 'Ready:' prompt with the screen display in the new line width.

The procedure is exactly the same for changing back from an 80-character line width to 40-character line width.

Note, however, that when using BASIC in ROM you can also use an escape sequence to do the switching. See Appendix B for the sequence required.

Appendix A
Hexadecimal Notation

Hexadecimal notation is widely used in computer-generated output (e.g. assembler listings and the Front Panel display) to represent addresses and the contents of memory locations. Where the decimal number system uses a base of 10, and the symbols 0-9, the hexadecimal (hex) system uses a base of 16, and the symbols 0-9, A, B, C, D, E, F. The relationship between decimal and hexadecimal is as follows:

DEC	HEX	DEC	HEX	DEC	HEX
One	1	Eight	8	Fifteen	F
Two	2	Nine	9	Sixteen	10
Three	3	Ten	A	Seventeen	11
Four	4	Eleven	B	Eighteen	12
Five	5	Twelve	C	etc	
Six	6	Thirteen	D		
Seven	7	Fourteen	E		

Four-bit Binary Numbers

There are sixteen different 4-bit binary numbers each of which can be used to represent a different hex digit:

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Conversely, a single hex digit is a simple way of defining a 4-bit number.

Eight-bit Binary Numbers

To cope with eight bits, the convention is to divide the eight bits into two groups of four and to write a pair of hex digits, the first corresponding to the left-hand four bits (most significant), and the second to be right-hand four bits (least significant). So a single-byte, eight-bit number, containing 10101111 can be written in hex as AF (1010=A, 1111=F).

Sixteen-bit Binary Numbers

In the same way that an eight-bit number can be represented by two hex digits, a sixteen-bit number can be represented by four hex digits.

Negative numbers within the 480Z are often held in 'twos complement' notation. In this notation a number is negated by complementing each bit and adding one. The advantage of this representation is that two such quantities can be added without special treatment for negative numbers.

	A	E
Write number in binary:	1010	1110
Invert (complement) each bit:	0101	0001
Add 1:	0101	0010
Convert back to hex and insert sign	—52	
Convert to decimal	—82	

So, in two's complement notation, the hexadecimal number AE is equivalent to the decimal number —82.

Alternatively, the same values could have been reached using hexadecimal arithmetic:

Subtract hex value AE from 100 hex:	100—AE	52
Insert sign	—52 hex	
Convert to decimal		—82

Appendix B
Escape Sequences

An escape sequence is a character string sent to the screen and recognized as special because of certain characteristic features. Instead of the characters appearing on the screen, other actions take place.

Escape sequences may be generated within a BASIC program by using PUT statements, and an example of their use would be to switch a screen capable of operating in either 40- or 80-character mode into the mode required for the current program.

Characteristic Features

An escape sequence is distinguishable from any other string of characters by the following characteristics:

- The first character of an escape sequence is the ESCAPE character (ASCII 27)
- The second character of an escape sequence is known as the Sequence Introducer (SI), which must be one of the characters from the list below, and its value determines the number of subsequent characters which are assumed to be part of the escape sequence
- The subsequent characters needed for each SI are also shown below, and may include a Switch (SW) and/or a Control Parameter (CP)
- A Switch may take the value '0' (to switch a feature off) or '1' (to switch it on)
- A Control Parameter may take any value given in the lists below.

Errors in Escape Sequences

Once an ESCAPE character has been sent to the screen to denote the beginning of an escape sequence and if the SI is not recognized then the escape sequence is immediately terminated and any further characters in the sequence are displayed on the screen in the usual way.

Illegal characters after a legal SI will invalidate the sequence but will not affect the number of characters diverted from the screen.

Any escape sequence in which an error is detected is not executed.

Escape Sequences

1. Send a special character to the screen:

   SI = '!' (ASCII 33)

   The sequence after the SI is a single byte to be output directly to the screen, with the characters in the range ASCII 0-31 or ASCII 127 *not* being interpreted as special characters.

Example:	**PUT 27,"!",10**
or, if preferred,	**PUT 27,33,10**

   will send a character with the ASCII value 10 directly to the screen.

2. Control the use of the 40-character line and 80-character line display:

   SI = '=' (ASCII 61)

   The sequence after the SI is a switch (SW) followed by a CP.

CP = 'F'	Switch off or on the CTRL/F action (See ROS Command CTRL/F, Chapter 5)
CP = 'G'	Switch off or on the CTRL/A action (CTRL/A is used to control screen output. See the 480Z BASIC in ROM Reference Manual)
CP = 'J'	Switch off or on the 80-character mode (only available on an 80-character system)

   Note: If you are in 'graphics' mode when the J parameter is used, the screen will return to 'text' mode.

CP = 'L'	Switch off or on the alternate character set
Example:	**PUT 27,"=1J"**

   will ensure that the 480Z is operating in 80-character mode (if it is capable of doing so).

3. Restore the function keypad keys to their original uses:

   SI = '>' (ASCII 62)

   The sequence after the SI is a CP only.

CP = 'D'	Restore all of the function and arrow keys to their original values (i.e. undo the effects of SI = '%' which is described in 6, below.)

4. Define or redefine the display scrolling window:

   SI = '?' (ASCII 63)

   The sequence after the SI is 4 bytes followed by a CP.

CP = 'A'	The 4 bytes define a scrolling window, in the order X(lower), X(upper), Y(lower), Y(upper), and the values must be such that: 0 <= XL <= XU <= 39 (40-character width screen), or 0 <= XL <= XU <= 79 (80-character width screen), and: 0 <= YL <= YU <= 23.
CP = 'B'	The 4 bytes define a rectangular area to be cleared, with order and value limits as above.
Example:	**PUT 27,"?",0,39,20,23,"A"**

   will define a scrolling window to be the full width of a 40-character screen, using only the bottom 4 lines of the screen.

5. Beeper sound:

   SI = '@' (ASCII 64)

   The sequence after the SI is 2 bytes followed by a CP.

CP = 'A'	The 2 bytes define the frequency and duration respectively of the beeper (which is sounded by CTRL/G).
Example:	**PUT 27,"@",150,75,"A"**

6. Define new uses of the function keypad keys:

   SI = '%' (ASCII 37)

   The sequence after the SI is a CP, followed by a byte (of value n), then n further bytes. The sequence is used to redefine the character string that is generated when a function or arrow key is pressed. The new character string will be the last n bytes of the escape sequence.

CP = 'A'	Redefine the up arrow key
CP = 'B'	Redefine the right arrow key
CP = 'C'	Redefine the down arrow key
CP = 'D'	Redefine the left arrow key
CP = 'E'	Redefine the F1 function key
CP = 'F'	Redefine the F2 function key
CP = 'G'	Redefine the F3 function key
CP = 'H'	Redefine the F4 function key

CP = 'I'	Redefine the SHIFT up arrow key
CP = 'J'	Redefine the SHIFT right arrow key
CP = 'K'	Redefine the SHIFT down arrow key
CP = 'L'	Redefine the SHIFT left arrow key
CP = 'M'	Redefine the SHIFT F1 function key
CP = 'N'	Redefine the SHIFT F2 function key
CP = 'O'	Redefine the SHIFT F3 function key
CP = 'P'	Redefine the SHIFT F4 function key

The character string can be of any length from 0 to 127 bytes, but the total number of characters for all keys is also limited to 127.

If the string is too long the escape sequence will have no effect.

Example: **PUT 27,"%G",4,"RUN",13**

will redefine the F3 function key so that it generates the string:

RUN [RETURN]

Appendix C
480Z Display Character Sets

Row ↓ \ Column →	0	1	2	3	4	5	6	7
0	▢	┬		0	@	P	`	p
1	◙	┐	!	1	A	Q	a	q
2	□	┌	"	2	B	R	b	r
3	°	┴	#	3	C	S	c	s
4	²	┘	$	4	D	T	d	t
5	³	π	%	5	E	U	e	u
6	✓	◆	&	6	F	V	f	v
7	▲	┤	'	7	G	W	g	w
8	√	├	(	8	H	X	h	x
9	▶	│	)	9	I	Y	i	y
A	≣	£	*	:	J	Z	j	z
B	♆	←	+	;	K	[	k	{
C	▼	½	,	<	L	\	l	¦
D	◀	→	-	=	M	]	m	}
E	┼	↓	.	>	N	↑	n	~
F	└	─	/	?	O	_	o	█

Table C.1 480Z Text Display Characters

Table C.1 illustrates the 480Z **text** character set. Note that the codes for each character are represented as **hexadecimal** numbers and you should convert these values to find the equivalent decimal values. For example, the symbol "@" is represented in hexadecimal as 40. The equivalent decimal value is 64.

Also note that the shaded characters, coded hexadecimal 03 to 1F (decimal 3 to 31), are displayed on the screen by a BASIC program either by using the PLOT command or by using the PUT command with a suitable escape sequence (see appendix B). Otherwise, they are interpreted as control characters for controlling the use of the display screen. Their use as control characters is summarized below in table C.3.

Dot Plotting Codes

As an exception to this rule, the three shaded characters 00 to 02 are displayed as shown in table C.1 **only** by means of an escape sequence. When these codes are given in a PLOT command, a single dot is plotted, with a width of one X coordinate unit and a height of 1 Y coordinate unit at **exactly** the screen coordinates specified and with either zero intensity (Code 0), medium intensity (Code 1), or full intensity (Code 2).

The complete range of text characters is coded from hexadecimal 00 to 7F (decimal 0 to 127).

Low Resolution Graphics Characters

Table C.2 contains a complete list of the low resolution graphics characters with their appropriate **decimal** codes.

Note:

- Graphics characters are in the range hexadecimal 80 to FF (decimal 128 to 255) and are the standard teletext graphics characters.

- Characters in the range hexadecimal 80 to BF (decimal 128 to 191) are displayed in **grey**, or low intensity.

- Characters in the range hexadecimal C0 to FF (decimal 192 to 255) are the same as the grey characters but are displayed in **white**, or high intensity.

- These low resolution graphics characters can be printed on a Microline printer but other printers, such as the Epson RX and MX series, cannot print them.

Table C.2 480Z Low Resolution Graphics Characters

Grey	White		Grey	White
128	192		160 🬞	224 🬞
129 🬀	193 🬀		161 🬟	225 🬟
130 🬁	194 🬁		162 🬠	226 🬠
131 🬂	195 🬂		163 🬡	227 🬡
132 🬃	196 🬃		164 🬢	228 🬢
133 🬄	197 🬄		165 🬣	229 🬣
134 🬅	198 🬅		166 🬤	230 🬤
135 🬆	199 🬆		167 🬥	231 🬥
136 🬇	200 🬇		168 🬦	232 🬦
137 🬈	201 🬈		169 🬧	233 🬧
138 🬉	202 🬉		170 ▐	234 ▐
139 🬊	203 🬊		171 🬨	235 🬨
140 🬋	204 🬋		172 🬩	236 🬩
141 🬌	205 🬌		173 🬪	237 🬪
142 🬍	206 🬍		174 🬫	238 🬫
143 🬎	207 🬎		175 🬬	239 🬬
144 🬏	208 🬏		176 🬭	240 🬭
145 🬐	209 🬐		177 🬮	241 🬮
146 🬑	210 🬑		178 🬯	242 🬯
147 🬒	211 🬒		179 🬰	243 🬰
148 🬓	212 🬓		180 🬱	244 🬱
149 ▌	213 ▌		181 🬲	245 🬲
150 🬔	214 🬔		182 🬳	246 🬳
151 🬕	215 🬕		183 🬴	247 🬴
152 🬖	216 🬖		184 🬵	248 🬵
153 🬗	217 🬗		185 🬶	249 🬶
154 🬘	218 🬘		186 🬷	250 🬷
155 🬙	219 🬙		187 🬸	251 🬸
156 🬚	220 🬚		188 🬹	252 🬹
157 🬛	221 🬛		189 🬺	253 🬺
158 🬜	222 🬜		190 🬻	254 🬻
159 🬝	223 🬝		191 █	255 █

Table C.3 480Z Display Control Codes

ASCII CODE		If sent to the screen by a PUT command, the code is normally intercepted and interpreted as a control code with the meaning:-
DEC	HEX	
0	00	—
1	01	—
2	02	—
3	03	—
4	04	Resume output
5	05	—
6	06	—
7	07	Beep
8	08	Cursor left
9	09	Tab
10	0A	Cursor down
11	0B	Cursor up
12	0C	Clear screen, cursor to bottom left
13	0D	Carriage return and line feed
14	0E	Carriage return
15	0F	Suppress output
16	10	—
17	11	Clear auto-paging
18	12	—
19	13	Set auto-paging
20	14	—
21	15	Flashing cursor
22	16	—
23	17	Steady cursor
24	18	Cursor right
25	19	Clear to end of line
26	1A	—
27	1B	Begin an escape sequence
28	1C	—
29	1D	Cursor to top left
30	1E	Clear to end of screen
31	1F	Clear screen, cursor to top left

Display Control Codes

The 480Z display is controlled by programs using a set of control codes. These control codes are in the range hexadecimal 00 to 1F (decimal 0 to 31) and their functions are defined in table C.3. They are used from within a BASIC program, using the PUT command. These control codes can be overridden and the special 480Z display characters with the same range of codes (see table C.1) can be displayed by a BASIC program using either the PUT command with an escape sequence or the PLOT command, as mentioned above.

One of the "escape sequences" described in appendix B allows any of the ASCII codes in table C.3 to be displayed as a special character on the screen by means of a PUT command, instead of being interpreted as one of these display control codes. This results in the same character being displayed as those shown in table C.1 for PLOT.

For example, the BASIC statement:

```
10 PUT "Circumference = 2",27,33,21,"r"
```

will display:

Circumference = 2 π r

on the screen, while the statement:

```
10 PUT "Circumference = 2",21,"r"
```

will display:

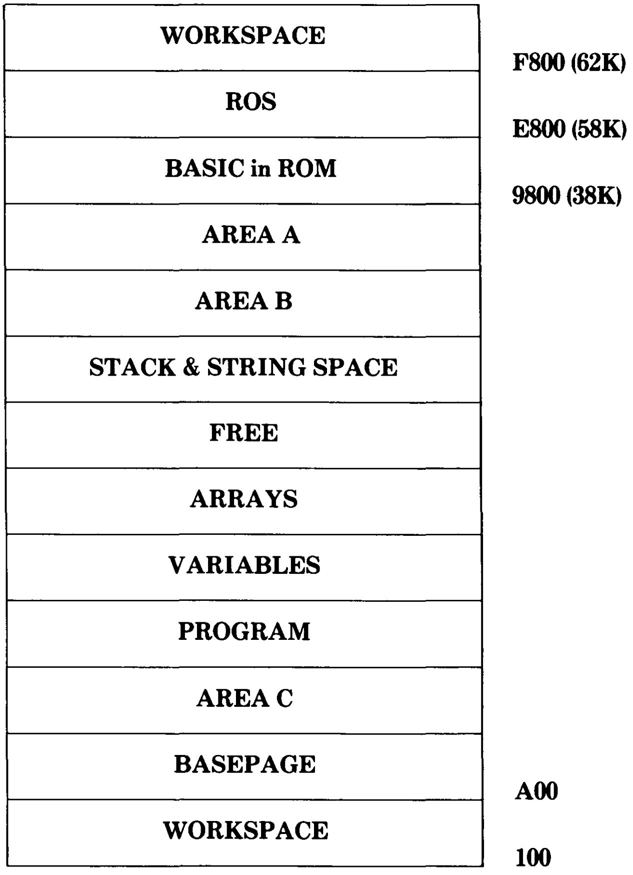
Circumference = 2r

on the screen and set the cursor to its flashing mode.

Appendix D
480Z Memory Usage Map

BASIC in ROM is held on ROM page 2 (see the LINK 480Z Information file). In this page there is ROM from addresses 9800 to E7FF hex holding the interpreter and also ROM from addresses E800 to F7FF hex holding ROS. The rest of memory on this page is RAM. On a 64K system, memory from 0 to 97FF is available to the user.

The RAM addresses between 100 and 9FF is used as workspace by the BASIC interpreter and the HRG routines.



For a full explanation of memory allocation refer to the 480Z BASIC in ROM Reference Manual, chapter 13.

Appendix E
Glossary of Terms

Applications software
: Programs designed to carry out a particular task or set of tasks specific to a particular user.

Argument
: A term used to describe a part of a command which can vary and which affects the action of the command.

ASCII
: American Standard Code for Information Interchange: the form in which characters are represented within the 480Z.

Assembler
: A program, usually provided by the computer manufacturer, to translate an assembly language program into machine code. An assembler is an item of systems software, and as such is specific to a particular model of computer.

Assembly Language
: A low-level programming language which is translated into machine code by an assembler. A computer's assembly language is closely related to its machine code (each instruction is translated into one machine-code instruction). Its advantage over machine code is that it is easier to use. Its main advantages over high-level languages are that its programs can occupy much less memory, run much faster, and control more closely the activities of the computer. However, assembly programs take longer to write and require of the programmer a more intimate knowledge of the computer and programming techniques.

BASIC
: Beginners' All-purpose Symbolic Instruction Code: a high-level programming language commonly available on microcomputers. Its advantages are that it is easier to learn its vocabulary and grammar than with most other languages.

Baud rate
: The rate, expressed in bits per second, at which data is transmitted through a serial interface.

Bit
: Binary Digit - one of the digits used in binary notation (i.e. either 0 or 1).

Break
: To halt execution of a program by entering a command at the keyboard or an instruction in the program.

Buffer

A part of computer memory set aside for temporary storage of data. It may readily be filled or emptied without disturbing the contents of the rest of the memory.

Bug

An error in a program.

Byte

A set of 8 bits, often corresponding to a single character (i.e. one byte can be used to represent one character). Microcomputer memory sizes are usually defined in terms of bytes.

Cassette-based (Configuration or microcomputer)

A microcomputer configuration that uses cassette recorders and tapes to save and load programs or data.

Central processing unit (CPU)

The "nerve centre" of the computer, consisting of the Arithmetic Unit, Control Unit and Memory. (Also called the Central processor). In a Microcomputer, the CPU is often on one single printed circuit board.

Character set

A character could be any one of the letters of the alphabet (upper or lower case), a numeric digit, or another associated symbol (e.g. * or ?). A character set is the set of characters that a device is capable of handling. In the context of this manual the character set is the ASCII set. There are small differences between the character set used by the LINK 480Z and between various printers but the alphanumerics are the same.

Command

A sequence of characters entered from the keyboard and executed by a program. Each command consists of the name of the command followed by a list of arguments.

Compatibility

Two computers are compatible if programs can be transferred between them and used without alteration.

Compiler

A program which translates a high-level language program into a computer's machine code before the program is executed. (Each high-level language instruction generally generates several machine-code instructions.)

Configuration

A term used to describe the central processor and the devices linked to it.

Control Character

A character typed while the CTRL key is pressed. The name originates from devices (such as the teletypewriter) which use these characters to perform control functions such as turning on and off the paper-tape reader. These characters are used to perform control operations (e.g. turning a printer on and off) or to issue commands.

Data

Information in a form acceptable to a computer for input, storage, processing, and output.

Default

The value of an argument or part of an argument that is assumed by the system when no argument value is supplied by the user.

Documentation

1. A complete description of a program usually including helpful notes, flowcharts, program listing, test data and sample output.

2. A complete description of a computer and its system software, usually in the form of manuals.

Editor (Text Editor)

A program which allows the user to edit a file. In some microcomputers, editing facilities are built into the operating system and are automatically available through simple commands. Editing facilities are very valuable since there is no need to retype complete files when only minor modifications are required. The standard LINK 480Z Text Editor is a program called TXED.

Enhance

To improve or "upgrade" a computer configuration by the addition of an extra facility or item of equipment.

Error Message

Response on screen to a faulty command or to an error detected by the operating system.

Execute

To obey a program by carrying out its commands (has the same effect as "run".)

File

Used in many computer applications to mean a collection or grouping of related information. A file on a magnetic disc or tape consists of a number of 'records' which have a common characteristic.

Front Panel

A feature that enables the contents of 480Z memory to be examined and modified and machine-code programs to be executed one instruction at a time.

Graphics

The ability of a computer to display diagrams and produce images as opposed to merely displaying or printing text.

Graphics Board
A circuit board inside the microcomputer that enables graphics to be used.

Hardware
The items of equipment making up a computer configuration.

High-level language
A language such as BASIC which is easily written and understood. It needs an interpreter or compiler to make it machine readable. See chapter 1 "Introduction to Microcomputers".

High resolution graphics
Graphics capable of displaying reasonably fine lines (i.e. 250 horizontal picture points or more).

Interface
The necessary hardware and/or software to form a working connection between two systems or parts of a system (for example, the central processor and a printer).

Interpreter
A program which executes another program (written in a high-level language) one instruction at a time.

Kilobyte (K)
Consists of 1024 bytes and is used to indicate the size of the memory of a microcomputer. For example, 8K of memory is 8192 (8 x 1024) memory locations (bytes). When using systems software, e.g. BASIC in ROM, the software indicates how many Kilobytes of memory it will occupy.

Low-level language
A language written in machine readable or mnemonic form. See chapter 1, "Introduction to Microcomputers".

Machine-code
The code used by the computer to represent the instructions which it will recognize and obey.

Monitor
A monitor television screen used to display output from a microcomputer. Same as Visual Display Unit (VDU). Should not be confused with Monitor Program.

Monitor Program
Term often used when referring to an operating system program (such as ROS in the 480Z) which monitors or controls the activities of the system.

Parallel I/O
The transfer of bits of data simultaneously. Each bit is transmitted along a separate channel or wire. See SERIAL I/O.

Patch
: A group of instructions inserted to provide an extra facility or to correct an error in a program.

Ports
: Memory addresses in which the data from external devices are placed on arrival and from which data are output.

Printer
: A machine which produces printed output (print-out) from the computer.

Program
: A complete set of programming language statements to perform a specified task.

Programming language
: A code (with a limited vocabulary of key words and well-defined syntax rules) that allows the user to communicate with a computer and instruct it to perform tasks.

Prompt
: Indication on the screen that the computer is ready for you to key in a command or a response.

Random Access Memory (RAM)
: Memory of a computer which is capable of holding the programs and data. The content of this kind of memory is lost when the computer is switched off. Sometimes called volatile memory.

Read-Only Memory (ROM)
: A memory whose content is fixed and not lost when the machine is switched off. The ROS monitor program is held in ROM so that it is available as soon as the computer is switched on. The contents of ROM cannot be altered by program instructions.

Register
: These are special memory locations reserved for specific functions, and are used to hold data that affects the overall operation of the system. A computer normally contains several registers.

Restart Address
: The memory address to which execution can be transferred to restart a program without losing the data on which the program is operating.

Save
: The transfer of a program from immediate memory to a backing storage medium, such as a cassette tape or a disc.

Serial I/O
The transfer of bits of data in sequence rather than simultaneously. Bits are transmitted along the same channel or wire one after another. See PARALLEL I/O.

Scroll
To display lines of characters on a screen either moving upwards with new lines coming from below or downwards with new lines coming from above.

Software
A term used to refer to programs in general.

Text Editor
A program used to enter, edit, and store text. The text sometimes represents a computer program or it may be an article or a letter. See EDITOR.

Upgrade
Normally used to mean an addition or enhancement to a computer system to improve its capacity, performance, or functional characteristics.

User Memory
The memory available in RAM to the user for storing programs and data; that is, the random access memory in the microcomputer that is not used by the operating system or other essential items of systems software.

Visual Display Unit (VDU)
A display device, generally incorporating a cathode ray tube (as used in a television receiver), on which information is displayed. See MONITOR.

USER'S COMMENTS

To help Research Machines to produce the highest quality microcomputers, supporting software, and technical publications, we like to hear from users about their experiences with our products.

Do share your thoughts with us by jotting them down on the tear-off form on the next page. You can leave out your personal details, if you want to. Fold the form in two, seal it with a piece of adhesive tape, and put it in the post. No stamp is needed if you post it within the United Kingdom.

If you would like to give more information than we have allowed room for on the form, we will be very pleased to receive a separate letter from you. You can even use the form to ask for a post-paid envelope, if you wish.

Additional information will be most useful, if you give us as much detail as possible about your hardware configuration, software version number, or manual title, so that we can relate your comments to the correct products.