1. Starters:

If those of you who skipped the beginners' introduction, (which included details of connecting up, switching on and keyboard familiarisa tion) find the terminology used here confusing then go back to the introduction section for the Foundation Course.

Subjects covered in this chapter:

* Conventions used in this user guide
* Switching on
* Keyboard familiarisation

No matter how familiar you are with programming and computers, please ensure that you follow these few setting up instructions. If you have just opened the box and cannot wait to get started, then this chapter will tell you all you need to know to satisfy your initial curiosity and so get into the application of the BASIC. This section is intended for users who have some familiarity with computers. Beginners should start at the Introductory chapter.

IMPORTANT - you MUST read this:

Terminology.

In order to clarify the references in the text to keys on the keyboard, and text that forms part of the program listing, the following conventions are used from here onwards in this user guide:

[ENTER] : Keys that do not have a corresponding printed character on the screen are shown in this form, surrounded by square brackets [ ].

QWERTYUIOP : Keys that have a corresponding printed character are shown in this form, without square brackets.

10 FOR N ▫ 1 t o 1000 : Text that either appears at the screen, or is to be typed in at the keyboard is shown in this form. Note difference between the zero 0 and the capital letter 0

It is assumed that you will end each program or direct command line by hitting **[ENTER],** and this will not be repeated in the listings given throughout the remainder of the guide.

Furthermore, it is assumed that you will type in run after each program is entered. BASIC converts all keywords entered in lower case letters into UPPER CASE when a program is L I S Ted. Examples shown from here on use UPPER CASE, since this is how the program will appear when L I S Ted. If you enter using lower case, you will be able to spot typing errors more readily since the mistyped keywords will still be displayed in lower case when L I S Ted.

1.1.1 OPEN THE BOX !

The Colour Monitor

IMPORTANT

Please refer to the Setting **Up** instructions detailed at the start of this User Guide describing the wiring up of the Mains Plug to the Mains Lead of your equipment.

With the computer correctly connected as shown in *Foundation 1*, switch on the monitor, and then the computer, using the slide switch on the right hand end. After about 30 seconds of warm up time, the monitor will display:

```
Amstrad 64X Microcomputer (v1)
©1984 Amstrad Consumer Electronics plc
      and Locomotive Software Ltd.
BASIC 1.8
Ready
```

This is known variously as the 'Reset', 'Early Morning' or Wake Up' message, that indicates the computer has been completely reset to its initial state - a condition which occurs on power up, and after a full computer reset via the keyboard, when the correct sequence of three keys have been pressed simultaneously.

(**[CTRL][SHIFT]** and **[ESC]**ape pressed in sequence and held down simultaneously - try it now before you enter anything else.)

Adjust the **BRIGHTNESS** control on the right hand side of the monitor. The colour is preset at the factory, so if you want to change the colour of the display (gold lettering on a blue background), this must done by program instructions and you will have to skip to the graphics primer and BASIC keywords in Chapter 8. If you don't mind not jumping ahead a little, here's a short program that will give you one of the best and most readable combinations of **colours** for text entry, using the high resolution **80** column display mode.

Type in:

```
10 MODE 2
20 INK 1,0
30 INK 0,13
40 BORDER 13
```

...or you could enter the above lines as single statements in the 'direct mode'.

If the above program means nothing to you, or you cannot seem to type it correctly, you will need to go back to the BEGINNERS FOUNDATION COURSE which you will find at the start of this guide. You'll find that the 80 column text display is easily the most useful display mode for developing programs - you might like to save the above short program (when you've read through chapter 2) on the start of a blank cassette, to save typing it in every time you switch on.

BEWARE! The high brightness of the colour monitor means that you must be careful to avoid eyestrain through sitting nearby, or using brightness levels that are greater than required by ambient lighting conditions. The moment you feel the onset of eyestrain, switch off and do something else, but to avoid it in the first place:

1. Always work with enough other light to enable you to read the print of this manual with ease. If you are reading this manual by the glare from the screen; this is definitely not to be recommended!
2. Use the minimum brightness required to see what you are doing with ease.
3. Sit as far away from the screen as you can.

A desk lamp positioned alongside the monitor will help to reduce eyestrain - as long as it is positioned behind the front of the screen to avoid reflections.

1.1.2 The Green Screen Monitor

The GT64 monitor has three controls beneath the display front. These are to provide the user with adjustment of the **BRIGHTNESS,** the CONTRAST (the difference between the brightest and dullest parts of the display), and the Vertical **HOLD** adjustment, that enables the user the lock the display, and prevent unwanted vertical 'rolling' of the display.

The Vertical **HOLD** will not require frequent adjustment - and once initially set, may be forgotton. The **BRIGHTNESS** and **CONTRAST** may be adjusted from time to time to suit the lighting conditions in the room where the CPC464 is in use.

Using a monochrome monitor (a single colour display that varies the intensity of the characters to provide contrast between the different elements of the display), the switch on message will be the same as with the colour monitor (see the previous page) - except that the text will be displayed in bright green on a duller green background.

Although too much use of your computer and GT64 monitor can lead to eyestrain, the generally 'softer' display from the mono monitor will be much easier to work with over an extended period. In particular, you will be able to make the most of the full width '80 column' text display mode (80 letters or numbers on a single line across the screen) since the resolution definition (the ability to display a number of small elements of the screen display close by one another without 'blurring') of a

monochrome screen is inherently superior to any but the most expensive colour displays.

Adjust the **BRIGHTNESS** control to provide an adequately lit display without the 'dots' that make up the lines in the characters of the display becoming excessively blurred.

To setup in the 80 column mode, type the following brief program listing into the CPC464. You get a choice of display:

```
10 REM set display format
20 FOR n=0 TO 26
30 MODE 2
40 INK 1,n
50 INK 0,(26-n)
55 BORDER n
60 LOCATE 15,12: PRINT "Hit any printing key
   to change the display format"
70 a$=INKEY$
80 IF a$="" GOTO 70
90 NEXT
100 GOTO 20
```

The above illustrates a further important point concerning the representation of style in this manual. Some program listing lines will wrap (over run the line end) and it is important to note that when we print the listings, the additional spaces occuring after a line break are not required in the program itself - they are inserted just to tidy up the listing.

This doesn't illustrate all the possible combinations of shades of grey (colours), but if will give an idea of what's available. When you find a combination you like, stop the program using the **[ESC]** key twice (a * B r e a k * message will occur).

From here on, the guide will be making many references that are specific to the colour monitor option. Programs that display interesting colour and graphics effects may appear nearly invisible on the monochrome monitor, although great care has been taken to produce a range of colours with a progression of corresponding grey levels (described more fully in chapter 5).

The advantage of the monochrome monitor is the crisper and less fatiguing display when doing program development - but if and when you find that you are bitten by the computing bug, then there's little hope of you avoiding the inevitability of getting the colour option as well!

1.1.3 The MP1 TV Modulator/Power supply

The MP1 is an additional item that you may wish to purchase if you are currently using your CPC464 computer with the GT64 green tube monitor. The MP1 enables you to use the computer with your domestic colour TV and thereby enjoy the full colour facilities of your CPC464 computer.

IMPORTANT

Please refer to the **Setting Up** instructions detailed at the start of this User Guide describing the wiring up of the Mains Plug and Mains Lead of your equipment..

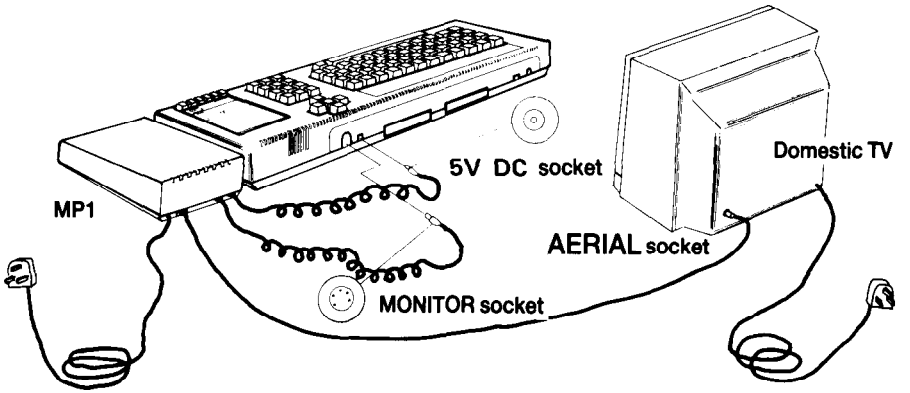


Figure 2: Connections for the MP1, computer and the aerial input of your TV

The modulator/power supply (MP1) should be positioned to the right of the computer on a suitable table close to the TV set and the Mains Supply socket. As shown in figure 2, above.

Now reduce the volume control on your TV set to a minimum - the CPC464 has its own internal loudspeaker, so the hiss from the TV with the volume turned up and the computer switched on is quite normal, Switch on your TV, and then switch on the computer using the slide switch marked POWER on the right hand end.

The red ON lamp at the top centre of the computer keyboard unit should be illuminated, and you must now tune in your TV set to receive the signal from the computer.

If you have a TV with push-button channel selection, press a channel button to select a spare or unused channel. Adjust the corresponding tuning control in accordance with the TV set manufacturer's instructions (the signal will be approximately at channel 36 if your TV has a marked tuning scale), until you receive a picture similiar to that shown on the next page:

```
Amstrad 64K Microcomputer (v1)
©1984 Amstrad Consumer Electronics plc
        and Locomotive Software Ltd.

BASIC 1.8

Ready
```

Tune in the TV set accurately until the clearest picture is seen. The writing will be gold/yellow on a deep blue background, although it will vary according to individual TV alignment conditions.

If your TV has a rotary programme selector knob, turn the tuning control until the above picture appears and remains perfectly steady. (Again, at approximately channel 36).

As the signal passes through all the various stages of the process of first being modulated, and then being demodulated, a certain amount of deterioration of the video signal will occur. The results cannot be as good as those obtainable by using a direct video interface monitor, and depending on the quality of your TV, you may find that the 80 column text mode (mode 2) produces results that are not perfectly clear, in which case you should use mode 1 for text entry whenever convenient.

1.2 First steps

You are now 'plugged in': the power supply should be connected, the video lead hooked up to either of the monitors or the modulator option, and you have switched on. Your computer is now waiting for an input:

The 'switch on' screen display, ie. the 'Wake Up' message, is the only 'built-in' text that you can display without first entering additional instructions through the keyboard.

If you are familiar with the BASIC programming language then there's a good chance that you will already have entered a brief program to 'get aquainted'. AMSTRAD BASIC will be familiar in many respects, and just to get you going, we will show you a brief program that you can enter that will display all the built-in characters that are available from the computer. This is the character set, which is the term used to describe the complete range of numbers, figures and other printable elements of the display that can be called up by typing at the keyboard.

Some of the characters that you will see are not directly accessible by pressing the keys on the keyboard, but only available for display using the ***PRINT CHR$(‹number›)*** statement described later on in this guide.

This is because each element stored in the computer is stored in the unit of data known as the 'byte' - and as you will see if you work through Appendix II, a byte has 256 different possible combinations of value. But as the computer has to use at least one whole byte per character stored (whether we want it to or not, it's the smallest denomination that the CPC464 appreciates), we might as well use all 256 possible combinations, rather than simply be satisfied with the 96 or so standard characters that are printed on most typewriters - and throw away the spare 160 possibilities.

The 'standard' range of characters is known as a 'subset'. It is classified throughout the computer world as the 'ASCII' display system, a term derived from:

American
Standard
Code for
Information
Interchange

...it's primarily a system that ensures the data sent from one computer to another is in a recognisable form. It's about the only aspect of computing that is truly universal, so we strongly advise that you become familiar with all aspects of ASCII. Appendix III lists the ASCII display range, together with the additional characters available on the CPC464 and their numeric codes.

Some of the other 'unprintable' characters can be displayed by using a combination of the CONTROL (marked on the keyboard as **[CTRL])** key and the other keys on the keyboard - but don't worry about that just yet, since until you understand the control key function, you can do more harm than good by testing it at random.

1.2.2

To see exactly what these characters look like on the screen for yourself, type in the following program, and we'll exercise your curiosity and the CPC464. This program will also help establish your confidence in both the simplicity of programming, and the fact that as long as you can get to the 'wake up' message on switching on, then you are not likely to encounter any further 'hardware' problems or misunderstanding, and the CPC464 is simply waiting to be programmed with the right information to process.

(If you make an error when you are typing this program, skip to 1.2.7 to see if you can correct the error without needing to restart entering the program from scratch.)

When typing the program on the next page into the computer, it doesn't matter if you use the lower case letters (a b c), or the **[SHIFT]**'ed uppercase letters (A B C), the computer will sort out the following program either way. You MUST delimit the words using spaces or other delimiters (commas, colons etc. as appropriate) at the positions shown, since AMSTRAD BASIC permits the use of the reserved words (fully listed in Appendix VIII) within variable names.

The 'physical' keyboard is shown in figure 4: it is referred to as the 'physical keyboard' here since many of the keys are available for user re-definition with a series of expansion tokens described in subsequent sections.

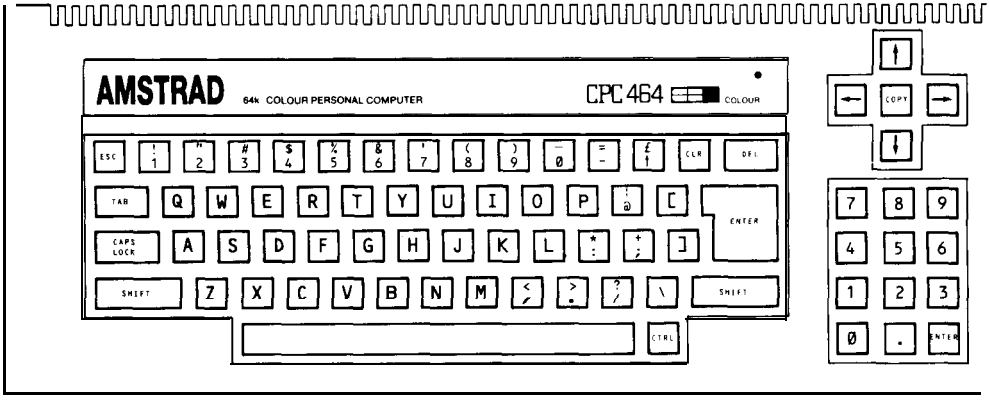


Figure 4: The keyboard of the CPC464

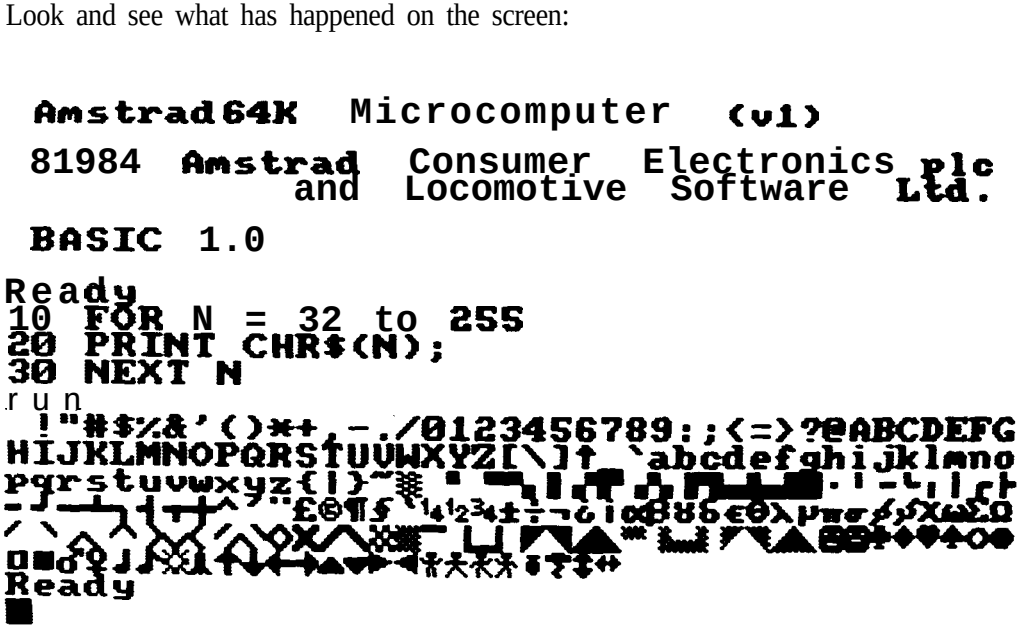
Pressing [ENTER] has the effect of **[ENTER]**ing the command or program line you have just typed into the computer, and then asking the computer to process the instruction contained on that line - or if the line began with a number, then to store it away as part of your program.

[ENTER] is also sometimes referred to as the 'carriage return' or simply 'return': which is a reference to the early forms of computer terminals which were based on mechanical typewriter principles. The term has remained, and is enshrined forever in the ASCII character set, where the code for [ENTER] is annotated by the letters 'CR'. Type in:

```
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
RUN
```

1.2.3

Look and see what has happened on the screen:



The computer has been told to display its full character set using the brief program you have just written. If it hasn't, then you've made an error in typing in the program that you haven't noticed - skip to 1.2.7 and see how to resolve the problem.

Assuming that all is well, and you have the required result, we'll examine what lies behind the displayed characters - it will help you understand exactly how your CPC464 communicates through its range of characters.

The first point is to notice that the computer is not instructed *to PR I NT "abcdefghijklmn..... etc"*, it is asked to:

PRINT CHR$(N)

N just happens to be a convenient shorthand note for a variable, the choice of the letter is arbitrary, it just happens to be the mathematicians' favourite in such applications, a variable is an item of computer information that 'varies' according to the instructions given in the program.

A number like 5 is fixed, it occurs between the numbers 4 and 6 - thus it is not a variable, a character N is also fixed - it's a letter from the alphabet.

So how did the computer know the difference ? If the letter N had been declared to be the alphabetical character, we would have typed N in quotation marks:

"N"

- and the computer would have responded with the message **S** *yn* **tax** *e r r o r* -because it does not understand the command sequence **F** 0 R "N".

Simply by using N in this way, we have told the computer that N is a variable. The definition of the F 0 R statement in BASIC requires that it should be followed by a variable - so the computer assumes that whatever follows F 0 R is just that.

We have also told the computer that N = 3 2 t o 2 5 5. Thus we have declared the range of the variable, it is in effect a sequence starting at 32, and finishing at 255.

Having declared this variable, we should then instruct the computer what it should do with it - the next line does just this:

```
20 PRINT CHR$(N);
```

It tells the computer that it should convert this number value that has been assigned to N into the character of the corresponding number, the C H R$(N) function is the BASIC instruction that performs this task. And having looked into its memory to see what character corresponds to the value of N, the computer prints it on the screen.

The semicolon at the line end instructs the computer to prevent the carriage return and line feed (set the next character to print back to the left hand column, and drop down to a new line) that otherwise would automatically occur, causing the character set to list down the leftmost column of the screen, rather than one character after another in rows.

The next line tells the computer that when it has performed the task with the first number in the sequence (32), it should return to the line where the F 0 R is located, and do the same again with the N E X T value it assigns (allocates) to the variable N. This process is known as looping, and this is one of the most vital and fundamental aspects of computer programming and operation.

This F 0 R loop is one of the most fundamental features of computing, it occurs in all programming lanuages in one shape or form. It saves typing in long repetitious sequences manually, and you will quickly come to use it in your own programming.

When this F 0 R loop reaches the limit of its declared range (255), the opeiation ceases and the computer then looks for the next line after line 30 - but there isn't one, so it simply stops and returns to the command prompt by displaying R e a d y. This tells you that the computer is ready to accept further instructions - or you can RUN again and repeat the execution of the program. The program is safely stored away in the memory and will remain there until you tell the computer otherwise -or if you turn the power off - when all data (programs, variables etc) will be lost unless you save it using the cassette.

This program neatly illustrates a fundamental point about computing - that is everything the computer does is related to numbers. The computer has displayed the alphabet - and a whole range of other characters - using a number as its reference to the character required. When you type the key marked A, you don't ask the computer to type an A on the screen, but you tell the computer to look into the part of its memory that contains the numeric information to display a letter A on the screen. The actual location of this data is defined by the numeric code that is activated by the action of typing at the keyboard.

Each character has a corresponding number, and these are listed in Appendix III of this manual.

Similarly the displayed character has nothing to do with 'writing' the letter on the screen, it's all about numbers.

1.2.4

Please don't worry if you do not understand all the technicalities or jargon used ***on*** this page. It is important to ***fully*** explain how the computer deals with your instructions and comes up with the required results, but it is also likely that only the technically minded will fully appreciate the explanation, *If you* find this section heavy going, skip to section 1.2.5.

For example, the code for the letter A is 97. The computer doesn't understand 97 either, and this number has to be translated from the human decimal code into a code that the computer can relate to - it's generally referred to as machine code, and the principles underlying this aspect of the machine are covered in APPENDIX II.

At first, the translation from the decimal number notation we are used to in every day life to the hexadecimal notation of the computer will seem heavy going. Thinking of numbers that are based on the ten unit is so natural that to do otherwise is like trying to eat with your knife and fork in the opposite hands.

A similar degree of mental dexterity must be aquired to understand hex notation (as HEXadecimal is abbreviated), but once you do, many things about computing will fall into place and the elegant structure of the numbering system will become apparent.

Once the computer has translated the striking of the A key into the type of number it understands, it looks into that part of the memory indicated, and the result is another series of numbers that define the character. That is to say the character you see displayed on your screen is built up from a block of data, stored in memory as a numeric matrix:

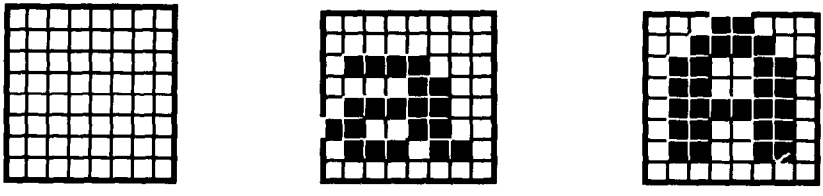


Figure 5: A blank character grid Lower case a Upper case A

The elements of the matrix are rows and columns of dots. The character is displayed by turning the required sequence of dots on or off - each dot is determined by data stored in the computer's memory. There are 8 rows and 8 columns in each character cell on the CPC464 display and if you don't find a character you want in the set of 256 that are provided, then you can redefine your own characters using the instructions that are given using the keywords S Y M B 0 L and S Y M BO L A F T E R in Chapter 8.

User defined characters can be made up using any combination of 0 to 64 dots, so the complete character set that uses all possible combinations of this matrix would comprise many more different elements (characters). Add to this the fact that you can group blocks of elements together to form larger block characters, and possibilities for user-defined characters are only limited by your time and ingenuity.

1.2.5 Back to the program !

The result of the first program you have typed looks rather untidy. There's still the remains of the 'wake up' message at the top of the screen. It looks tidier if we wipe the screen clean before we started the program running. We'll add one line to the program to fix this.

Type the following in on the line where the cursor rests (the cursor is the solid block immediately underneath and to the left of the Ready prompt message, and if you didn't know that, what are you doing reading this before the foundation section??):

```
5 CLS
RUN
```

See how the screen clears completely this time before writing the character set starting at the top left.

This also demonstrates one of the most understanding aspects of the BASIC programming language, namely that it does not matter in which order you enter the program line numbers - and you don't actually need to have the program displayed, to add to it once it's been entered into the memory.

The computer always sorts the line numbers into strict numerical sequence before it starts to execute the program. Check by using the L I S T instruction.

1.2.6 L I S Ting

You can easily check to see what the computer has stored in its program memory by asking it to list. Type:

```
LIST
```

and the result on the screen is:

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
```

This program will stay in CPC464's memory until you either:

*Switch off

*RESET - by pressing **[CTRL][SHIFT][ESCAPE]** in sequence and holding each down after it has been pressed until the reset occurs.

* L 0 A D or R U N a program from the cassette unit

*TYPE N E W [ENTER] which resets all variables and clears the program memory without resetting things like the display mode and the colours.

Now set up one of the function keys to perform the task of typing **[ENTER]**C *LS:LIST[ENTER]* - a feature that speeds up program entry and development enormously. To do this, enter:

```
KEY 138, CHR$(13)+"CLS:LIST"+CHR$(13)
```

Now press the decimal point key on the numeric keypad. Up to 32 keys can be pre-programmed in this way and you can choose any of the keys on the keyboard to redefine in case you want to use the keypad for its original purpose, see the description of the K E Y command in chapter 8.

If yours is a long program, then define the key as:

```
KEY 138,CHR$(13)+"CLS:LIST"
```

then the key will allow you to enter the line number range you require - or hitting it twice in succession will perform a full listing.

When you are experimenting with colour, it's possible to get lost in a combination of colours where it's not possible to read the display because the background and the writing have been set to the same values, so if you set:

```
KEY 139,CHR$(13)+"mode2:ink1,0:ink0,9"+chr$(13)
```

. ..you only have to press the smaller of the two **[ENTER]** keys (the one on the number keypad), and you can get back to base with a combination of colours that are visible. (You will not lose the program in memory.)

User defined key codes are reset with the rest of the machine since they have to be made available to program instructions, so once you have worked out your favourites, write them into a program, and save it onto tape for easy recall.

1.2.7 Editing primer

You will inevitably make mistakes when typing in the program. Welcome to this section all those of you who skipped here from 1.2.2 !

The CPC464 has tried to make correcting these errors as simple as possible, at the same time avoiding the problems of accidentally overwriting characters that you don't mean to change.

The clustered keys that control the motion of the cursor (the solid block that defines where you are going to type the text) provide the means of steering the computer's attention to the part of the display you wish to alter.

When entering an error in a numbered line eg:

```
10 FOR N = 332 TO 255
```

you have several options:

1. You may hit **[ENTER]** and retype the whole line. The incorrect line will be erased from memory and then be overwritten by the next line to be typed that starts with the same line number.

2. You can press the **[←]** key and move the cursor block back to the incorrect entry:

```
10 FOR N = 332 TO 255
```

Note that the character underneath the cursor is displayed in reverse video. In other words, the character which is normally the colour of the cursor has become the same colour as the background (the 'paper') so it will show through the cursor block that is presently placed over the top of it.

Now press the key marked **[CLR]** (short for CLEAR) and the character within the cursor will disappear - and the line closes up to fill the gap:

```
10 FOR N = 32 TO 255
```

Press **[ENTER]** again, and this corrected line will be the one that the computer remembers. The cursor doesn't have to be at the end of the line - the computer enters the whole line - irrespective of the cursor position.

3. You can also take the cursor back to the character immediately to the right of the one that you wish to delete:

```
10 FOR N = 332 TO 255
```

Now press **[DEL]** (short for DELETE), the character to the left of the cursor is deleted, and the cursor pulls the line along behind it without affecting the character within the cursor.

Press **[ENTER]** and the line will be stored as before.

```
10 FOR N = 32 TO 255
```

1.2.8 Afterthoughts

The preceding methods are fine if you spot the error before you reach the line end and type **[ENTER].** Most errors, however, occur inadvertently and only come to light when you try to run the program - and the computer responds with an error message (APPENDIX VIII).

A number of errors will cause the computer to display the faulty line, with the editing cursor placed over the first column (left end). If this is the case, then you can use the procedures above as if you had spotted the error before entering the program line.

If the error does not prompt you with the faulty line, you will have to LIST the program, find the cause of the problem, and fix it.

1.2.9 Copy Cursor Editing

First list the program using L I S T. (We'll continue to assume that you're working on the short trial program that fits onto a single screen)

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
```

The error is in line 20 - there's an S instead of a string signifier **$**(**$** tells the computer to consider the characters immediately following to be text, and not numeric data). You can either retype 20 from the beginning, and then re-enter, or you can use the screen editor as follows:

Hold down **[SHIFT]** (either the one on the left or right of the keyboard will do) and then press the up cursor key [↑].

You can either tap it a line at a time, or you can hold it down and wait for the auto repeat to move it for you. The instant you take your finger off the key, the cursor stops, and a little **practise** will soon familiarise you with the effects. If you overshoot the line, then return with the 'down' key **[↓]** again whilst holding down **[SHIFT]**.

This process causes the 'COPY CURSOR' to separate from the main cursor (they both look the same), and the copy cursor is taken to the line you wish to modify. Start with the copy cursor placed over the first character in the line.

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
Ready
■
```

If you were to take the main cursor to the line to modify using the cursor keys without the **[SHIFT]** held down, the computer would not **recognise** this as a valid action since only the characters entered directly after the main cursor will be **recognised** as valid instructions.

If you do try and overwrite the line in this way, you can easily get out of the tangle by pressing the **[ESC]** key ***BEFORE you*** hit any **[ENTER]** key or function key that contains a c h r **$**(13). If you accidentally type (and **[ENTER])** the command word N E W - then your entire program will have been lost forever - so be careful.

Once you have started typing on a line, if you try and move out of it without pressing **[ENTER],** the computer will bleep if you hit an illegal boundary. once you escape from the problem by hitting **[ENTER]** nothing will be lost in the program -unless you have entered a valid line number as the first typed characters, in which case the line whose number has been written will be 'overwritten' and will need to be retyped.

With the COPY cursor correctly in place, keep tapping the **[COPY]** key until the COPY cursor reaches the item on the line that you need to change. (**When** you get used to the speed at which the COPY cursor moves, you will be able to hold down the the **[COPY]** key, and move the cursor more rapidly).

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
Ready
20 PRINT CHR■
```

now release **[COPY]** and type **$** - it will appear under the MAIN cursor, which then steps along one place to the right as usual.

```
20 PRINT CHR$■
```

You must step the COPY cursor past the erroneous S, and to do this hold down the **[SHIFT]** key, and then press the cursor right key **[→]** once. The COPY cursor then rests over the (. Release the **[SHIFT)** and hold down the **[COPY]** key until you reach past the end of the line. Hit **[ENTER]** and the corrected line at the bottom of the screen replaces the incorrect line shown in the listing.

You can combine all these techniques by simply **[COPY]**ing the entire faulty line, and before you hit **[ENTER]** at the end, edit it using the line editing features of the main cursor, using the cursor keys, and the [CLR] and **[DEL]** keys without the **[SHIFT]**. Holding down **[CTRL]** and either the cursor left **[←]**, or the cursor right **[→]** key will take the cursor to either the left or right hand end of the line being edited, in one step.

Practise, you'll find it gets easy in a short while.

Finally, you can edit by typing:

```
EDIT 20
```

The computer responds with:

```
20 PRINT CHRS(N);
```

Simply use the main cursor keys together with the **[CLR]** and **[DEL]** keys, as instructed above, and when you are satisfied with the result, press **[ENTER]** as before. If you get in a tangle, press **[ESC],** and *L* I S T once again. The line where the **[ESC]**ape was pressed will not have been overwritten.

Now enter *L I S T* once again, and you will see the corrected version of the program displayed. If it's not correct, try again!

Thus far we've only begun to explore the groundwork for CPC464. To take a look at the other two modes, type in:

```
MODE 0
RUN
```

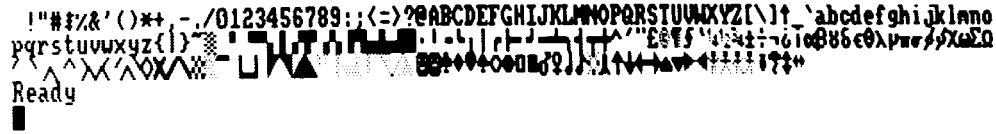
Note that the display first clears, then your program displays the characters 20 wide across the screen:

To return to the original display again, type:

```
MODE 1
RUN
```

And we're back where we started. To look at the 80 column display, type:

```
MODE 2
RUN
```



We've made a start - and you should by now have overcome some of your initial curiosity. Advanced users will already be setting about converting their favourite programs to run under this particular dialect of BASIC, less familiar users should proceed through the primer sections to get an overview of the machine-specific features in the BASIC.