AMSTRAD

PERIPHERAL – TECHNICAL MANUAL

EDITION 2

dk'tronics®
Limited

CONTENTS ***PAGE***

CONTENTS

CP/M is a registered trademark of Digital Research Inc.

Introduction

This manual contains all the current deatils of peripherals for the Amstrad 464, 664 and 6128 computers that are available from DK'tronics.

All peripherals are supplied with comprehensive software in cassette, disc or ROM format and are usually controlled through BASIC via Resident System Extensions, (RSX's). As such the new commands are preceeded by the (BAR) | symbol which is obtained by pressing the <SHIFT> key together with the @ key.

Throughout this manual all command key entries are shown enclosed in the symbols < >, ie <ENTER>.

The command keys are given as follows :-

<ENTER>
<CLR>
<CTRL>
<SHIFT>
<COPY>
<ESC>
<TAB>

All text, numeric entries are inclosed within the symbols ' ', ie type in '|lightpen'.

If the user is required to enter values for the RSX commands then these are enclosed in the brackets [ ].

When ever the user is required to enter data for the various RSX's then the syntax is as given in the example below:-

```
|savew,[ window number ],[ bank ]
```

Therefore the user would actually type in the following:-

press <SHIFT> together with the @ key ,to obtain the (bar) symbol,

then type in 'savew,5,2' then press <ENTER>

where window number = 5 and bank = 2

GRAPHICS LIGHT PEN

The DK'tronics graphics lightpen is compatible with the GT64 * CTM640 * MP1 and DDI 1 interfaces.

It is supplied with a sophisticated graphics package in both cassette and ROM versions.

Features include:-

- colour palette
- 'nudge' control for 1 pixel accuracy
- brush choice
- text handling
- user defined sprites
- magnify
- shrink
- circles
- rectangles
- lines
- curves
- colour fill
- tape and disc facilities
- picture storage and retrieval
- pen calibration utility
- printer dump

GRAPHICS LIGHTPEN MANUAL

WARNING

This unit must be used in accordance with these instructions. Never plug in or remove the interface without first disconnecting the power from the computer. Failure to follow these instructions may result in damage to the interface or the computer.

1.1 INSTALLATION

To set up your Graphics Lightpen, TURN OFF THE POWER TO YOUR AMSTRAD, then plug the interface into the disc port on the CPC 464 or the expansion port on the CPC 664/6128. If you have a disc drive (464) then fit the disc drive interface onto the back of your lightpen via the through connector. Attach the lightpen itself to the jack socket on the left side of the interface.

The DK'Tronics Graphics Lightpen is compatible with the Amstrad disc drive (DDI-1) and the DK'Tronics speech unit, and any combination of the three can be used.

Now turn on your computer. The computer should initialise normally. If it does not, switch off the power to the computer by the switch on the monitor and check that all the connections have been made correctly. Now switch on again. Due to the construction of the monochrome monitor, you may have to wait a short while before switching on while the monitor resets power to the computer, this is quite normal.

1.2.1 SOFTWARE LOADING DETAILS (cassette)

Your Graphics Lightpen can be used directly from BASIC or machine-code programs of your own, but the graphics package supplied illustrates what can be achieved with the lightpen and serves as a useful demonstration of its powers. It is also a powerful program in its own right as shown by the artwork on the packaging, which was drawn using the Graphics Lightpen.

To load, simply place the cassette in the tape drive, press <PLAY>, and type 'RUN ""' and press <ENTER>. The program will now load into the computer.

If you have an Amstrad disc drive attached to your computer then you need to type '|TAPE' and press <ENTER>. Now type 'RUN ""' and press <ENTER>. The procedure for loading on the Amstrad 664 is similar except you need to attach an external cassette unit before using the above instructions. Full instructions on loading tapes may be found in your Amstrad manual.

The tape is recorded on both sides using SPEED WRITE 0 and is NOT protected, so that users with disc drives may copy the software to disc. After four and a half minutes the Graphics Package should have loaded.

1.2.2 SOFTWARE LOADING DETAILS (ROM)

All that is required to run the ROM software is to type in "|lightpen" then press <ENTER>.

1.3 RUNNING THE PACKAGE - BASICS

Once the program is run, the screen will clear to a white background with a blue border and a menu will appear. A menu is a list of tasks which the computer can carry out. Instead of you having to tell the computer what to do by using symbols and words (like in BASIC), it enables you to select from a certain number of functions. Compare this to the menu you might find in a restaurant. The menu saves you having to ask for meals which are not available, and reminds you of the kinds of dishes that you can have. Similarly when you want the computer to draw a circle or edit some screen, you simply select an item from the menu. To make the things easier to remember, all the choices are in the form of pictures. These express what the function is without using lots of text!

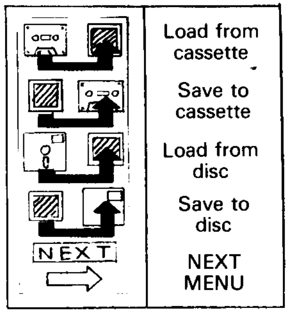
This is the first of several menus which hold up to five choices represented as pictures or 'ikons'. Pointing the lightpen to each ikon will make a set of white brackets appear around the ikon. This indicates which choice you require. When the brackets are around the ikon you require, press either <ENTER> key to make your choice. Remember to leave the lightpen pointing at the ikon while you press <ENTER>.

If you wish, the selection can also be made by pressing the number keys '1' through to '5' and then <ENTER>. This is useful for making choices before the pen has been calibrated. You can change your mind once you have pressed a number key, but you cannot use the lightpen to select an ikon again until the next menu appears. Once <ENTER> has been pressed your choice is made, however pressing <ESC> will get you back to where you were.

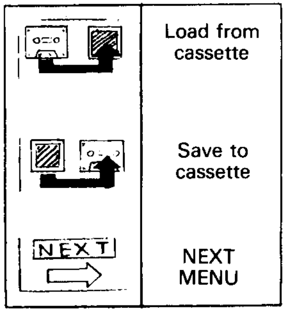
Practice pointing the lightpen at the ikons to make choices and see where you get to! Remember <ESC> will always get you back to the last menu.

When you are familiar with choosing ikons, press <ESC> until the very first menu appears on the screen. This is the LOADING and SAVING menu and allows you to load and save pictures you have drawn to tape or disc. The menu will look different depending on whether you have a disc drive attached:-

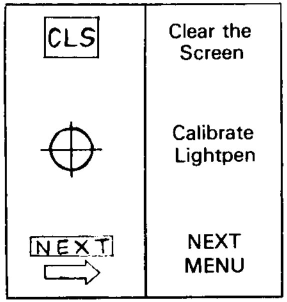
Disc Connected



Tape only



Point to the last ikon 'NEXT' or press '5'. Then press <ENTER>. A new menu will appear with just three ikons:-

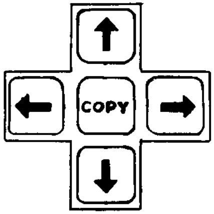


The first ikon will clear the screen's contents and should be carefully used! The second allows you to enter the lightpen calibration routine. Point the lightpen at the calibrate ikon and press <ENTER> to confirm your choice.

A target like the ikon itself will appear on the screen. Now hold the lightpen up against the screen in the centre. If all is well the target will hover directly under the top of the pen. Increase or decrease the brightness and/or contrast on your monitor or TV until the target follows the pen to all corners of the screen. For best response hold the pen perpendicular to the screen, touching the glass and keep your arm well supported to prevent your hand shaking.

Once you have got the target to follow the pen, reduce the brightness until you get best results with minimum brightness. The target may 'jitter' to a greater or lesser degree but this does not matter as you will see when using the 'nudge' feature.

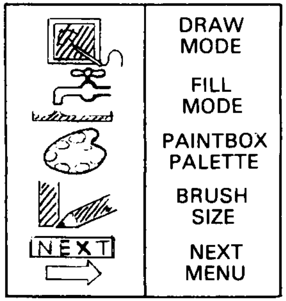
The target may seem to be a short distance away from the tip of the lightpen. If so, you can adjust the centre of the pen by using the Amstrad's cursor keys:-



Hold the pen very steady and press the respective directions to move the cross under the tip of the pen. Press <ENTER> to record this new setting or <ESC> to leave the setting unchanged. You can recalibrate at any time during using the program.

Your lightpen is now fully functional and calibrated. Do not worry if it does not seem accurate enough for fine line drawing - using the DK'Tronics 'nudge' feature the pen can draw as accurately as you wish:-

The last menu will reappear. Select the 'NEXT' ikon and press <ENTER> to confirm your choice. A new menu will appear:-



You are now in the third menu. Pressing <ESC> will make the previous menu appear. Press <ESC> again and the first menu will appear. Pressing <ESC> again will not take you to any further menus. Select 'NEXT' to get into the second menu, then 'NEXT' again to get back to the third menu.

*** *REMEMBER* ***

Pressing <ESC> whilst a menu is on the screen will take you BACK one menu.

Pressing <ESC> whilst you are performing some action on the screen (eg Calibrating, Drawing) will simply make the current menu reappear.

In all cases, selecting 'NEXT' selects and displays the next menu.

There are five main menus and several sub-menus. You now know enough to flick through the five menus and back again but do not select any function as yet.

Go back to the third menu and select the third ikon, a paint palette. A paint box will be displayed (different tones on the monochrome monitor). Use the lightpen or number keys ('1' to '0') to select a colour/tone from the box. The paint box is surrounded by a yellow boarder, you can use this to select the darker colours by pointing the pen to the border next to the required colour.

*** *REMEMBER* ***

The lightpen needs light to work - it will not pick up on dark colours or tones.

Select a bright colour/tone and press <ENTER> to confirm your choice. If you press <ESC> then the colour will remain the same and you will return to the last menu.

Now select the fourth ikon, the pencil ikon. A menu of pencil sizes will appear and you can select any size you want! The bottom ikon is a spray can! this produces a splatter effect like an airbrush. Press <ENTER> to select your pencil.

When you return to the third menu, point the lightpen to the first ikon and press <ENTER> to confirm your choice. You are now in draw mode!

There should now be a clear screen with a single small cross under the lightpen. This cross represents where the computer 'sees' the lightpen. Pressing <ENTER> will draw a blob using the pencil and colour/tone you selected. Move the pen around the screen and press <ENTER> again. More blobs will appear! If you keep the Large <ENTER> key pressed down, then repeated blobs will appear and in this way you can shade in areas of screen.

Now press the space bar. A continuous line will be drawn from the last point to the tip of the lightpen. If you are using a thin pencil then you can keep the space bar pressed and draw sequence of lines quite quickly. If you are using a thick pencil then you may have to move the lightpen more slowly.

When you are fed up, try pressing <ESC> and changing the colours and pencil sizes as outlined beforehand. Also try clearing the screen by pressing <ESC> twice and going back to the second menu.

When you have finished making patterns on the screen using the drawing mode, clear the screen then enter draw mode again. Now move the pen to the top right hand corner of the screen without pressing any key. Now press one of the cursor keys and notice what happens.

The cross has now halted in one position and can only be moved by using the four cursor keys. This is the 'nudge' feature which means you can move the cross to a single pixel. Pressing <ENTER> or the SPACEBAR willstill have the same effect as before. If you press <COPY> then the cross will once more move with the lightpen. Try to draw some shapes by moving the cross, pressing <ENTER> then moving the cross to another point, use the 'nudge' feature to move the cross to an exact position for say a square or a triangle. Now press the SPACEBAR to draw a line to that point. Finally, use the 'nudge' to link the final edge of your shape to the starting point.

1.4 FILLING IN SHAPES

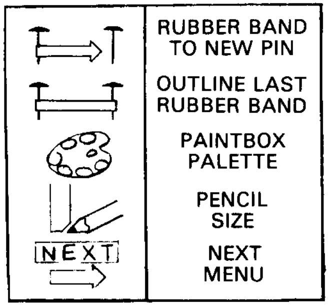
When you have your shape on the screen you may wish to fill in the centre with a colour/tone. Press <ESC> to re-enter the third menu. Then select the second ikon, the tap! After you press <ENTER>, the menu will disappear and the cross will reappear. Move the cross so that it is inside the shape and press <ENTER>. If you want to fill in small shapes where it is difficult to move the tip of the light pen into the centre of the shape, then move the cross close to the shape, then use the 'nudge' feature to move the cross into the exact area to be filled. Then press <ENTER> as before.

Watch out for shapes which have 'a leak'! Any shape which is not fully enclosed will let the ink spill out until it fills the whole screen. You can still press <ESC> to stop a fill whilst it is in action. It is worth remembering that whatever colour exists at the point where you start filling is taken as background colour. Hence you can remove objects by filling in a shape with white colour.

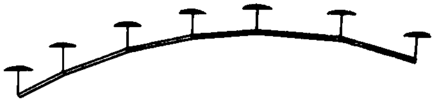
The best way to learn how to use fill is to experiment with lots of different shapes drawn in draw mode, then fill them in. Remember to change the paint colour and pencil size. You may even like to try and fill the outline of a shape. If you use a thick line, then you can change the colour of the outline in one fill command.

1.5 DRAWING CURVES AND COMPLEX SHAPES

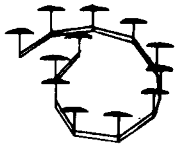
Select the fourth menu:-



For drawing complex shapes or curves, the software makes use of 'rubber bands' stretched between 'drawing pins'. For instance, a curve can be considered as a series of short lines:-



Or even:-



The Graphics package allows you to define a shape on the screen using one rubber band and up to twenty drawing pins. This feature is most useful because just as real drawing pins can be removed, so can the imaginary computer drawing pins.

Once you are happy with the shape you have drawn, the computer removes the rubber band and pins then outlines the shape in the current pencil and colour/tone.

Go back to the second menu and clear the screen. Then return to this menu and select the first ikon. Press <ENTER> to confirm your choice. A cross like the one used in draw and fill mode will appear in the bottom left of the screen. The first effect that you may notice is that the cross does not follow the lightpen. This is because the cross is initially locked in position and will only move to the lightpen when <COPY> is pressed. Try keeping <COPY> pressed and waving the lightpen on the screen. The cross represents where the first drawing pin (or the start of a shape) will go. Let the <COPY> key go and the cross will lock into position. It can now be finely adjusted by using the cursor keys.

When you have the cross just where you want it, press <ENTER>. The cross now becomes a pin, and one end of the rubber band is fixed to the screen. If you now press <COPY> again while moving the lightpen on the screen, you will see a line from the first pin to the cross at the tip of the lightpen. The line will not affect anything already on the screen. Press <ENTER> after the line is correct. This will insert the second pin into place and the rubber band is held between the two pins. If you press <COPY> again a second line will come from the last drawing pin. As you continue putting pins down, the shape grows. Up to twenty pins can form the rubber band into many complex shapes.

For very small shapes you may not even want to use the lightpen for positioning of the small cross.

If you make a mistake in the shape you have drawn, press <DEL> to remove the drawing pins in the reverse order to which you placed them.

Once your shape is finished, press <ESC> to enter the fourth menu. You may now wish to select a colour and a pencil to use, then select ikon number two, outline rubber band. On pressing <ENTER> the menu will briefly disappear, the computer will remove the rubber band, and then outline the path of the rubber band with the pencil and colour you have just chosen.

The rubber band, although invisible, is still active, and can be reused. Try choosing a different pencil, then select the second ikon and outline the rubber band in a different pencil. You can repeat this as many times as you like, until you use the rubber band mode again by choosing ikon number one.

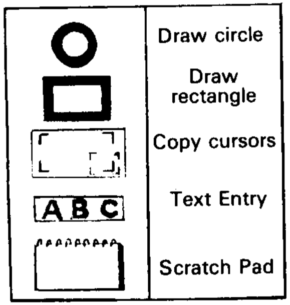
Try using the rubber band to draw some shapes on the screen and see if you can fill them in different colours.

If you run out of rubber bands, then return to the menu and outline that section of the shape, then continue the rest of the line by entering ikon one again.

NOTE: If you define a rubber band and do not use it, when you select the rubber band ikon again, the last rubber band will still be in use and you may have to press <DEL> to remove it.

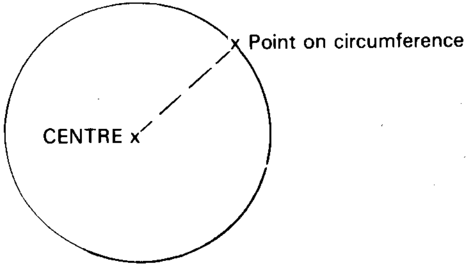
1.6 CIRCLES AND RECTANGLES

Select the fifth menu:-

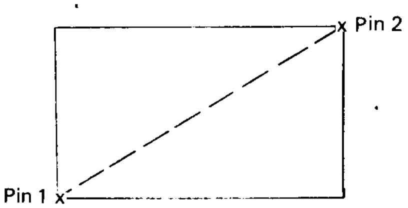


Circles and rectangles can quickly be drawn by using the rubber band to specify details about these shapes as follows:-

CIRCLE:-



RECTANGLE (or SQUARE):-



1.6.1 CIRCLE DRAWING

Enter rubber band mode (ikon 1, menu 4). Move the cross to where you want the centre of the circle to be. (By pressing <COPY> or using the nudge keys). Then fix that centre point by pressing <ENTER>. Now stretch out a rubber band to become the desired radius. (That is, the cross is a point on the circumference of the circle). Now fix a second pin using <ENTER>, then press <ESC>.

Once the radius is defined, select the 'NEXT' ikon to enter the fifth menu. Now select the circle ikon on the top of the menu. The computer will now draw a circle around the defined radius in the current pencil and colour.

Note that the computer will refuse to draw a circle if it goes off the screen. If the computer does refuse to draw a circle, try shortening the radius, or moving the centre.

1.6.2 RECTANGLE DRAWING

The procedure is almost the same as for the circle above, except that the two points defined are the ends of one of the diagonal lines through the rectangle. When you enter the fifth menu, select the second ikon (the rectangle) and press <ENTER>. The computer will now draw a rectangle around the diagonal you have just defined, in the current pencil and colour. It does not matter which direction your diagonal goes, as long as it is not a horizontal or vertical line.

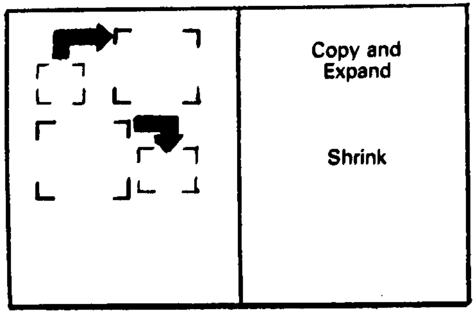
MENU REPETITION:-

You may have noticed that the pencil and palette ikons appear on more than one menu. This is just to make selecting of colour and pencil easier.

1.7 COPYING, EXPANDING AND SHRINKING

Often when you have drawn one complicated shape on the screen, you want to make copies of it. Copies can be easily made, or even copies which are bigger or smaller than the original.

To demonstrate this facility, first clear the screen, then draw a circle in the centre of the screen (see section 1.6.1). Now enter menu five and select the third ikon. Press <ENTER> to confirm your choice. A sub-menu will appear containing only two ikons:-



The first ikon represents the copy with possible magnify. The second ikon is the copy with possible shrink.

Select the first ikon, and press <ENTER>. The menu will disappear and your circle will be on the screen. Somewhere on the screen will be a set of green (lighter on monochrome) flashing corner brackets.

Press <COPY> and move the lightpen on the screen. The brackets can be positioned in the same way as the rubber band cross. Use the cursor keys to fix the final position of the brackets.

The brackets surround the block which you want to copy. Try to get your circle into the centre of the brackets. If the circle is too big (which it probably will be!), then try the following:-

1. **Press down the shift key with one finger, then press the cursor keys.**

2. **Press down the <CTRL> key with one finger, then press the cursor keys.**

You should have noticed that the size of the brackets actually changed, and in this way try to get the cicle to fit into the brackets.

Even after you have changed the size of the brackets, you may move them by pressing the cursor keys unshifted, or moving the lightpen while pressing <COPY>. Notice how the brackets will not go off the edge of the screen.

Now that you have the area of screen you want to copy defined, you need to tell the computer where to copy it to.
Press <DEL>. There are now two sets of brackets on the screen (although when you first press <DEL> they are on top of each other). Keep <COPY> pressed and move the lightpen on the screen. Notice now that the second cursor is blue (or dark on monochrome) and flashing and that the first set of brackets is no longer flashing. Hence it is easy to tell which bracket is under lightpen control by which set is flashing. Pressing <DEL> again will swap back and forth between the two cursors. Now when you alter the size of brackets, you will effect the size of both cursors.

Move the second set of brackets to a different position, using the 'nudge' feature for final setting. Now press <ENTER>. The computer will copy the area of screen in the first set of brackets into the second set of brackets.

When the copy is complete (the speed of copy depends on how large the brackets are), the original green/light set of brackets will reappear and you can do another copy. Press <ESC> to exit when you have finished. The green/light brackets stay in the same place at the same size, so for multiple copies, simply press <DEL> and move the blue/dark 'TO' cursor to a new position and press <ENTER>.

Press <ESC> to return to the fifth menu as normal when you have finished with copying.

1.7.1 HOW TO ENLARGE AN OBJECT

Clear the screen and draw a very small rectangle in the bottom left of the screen (about six pixels high by eight wide) (see section 1.6.2). When you are in menu five, select the third ikon (the one with two copy cursor brackets). Now select the top ikon in the next menu, (the one showing a small green/light copy cursor and an arrow pointing to a larger blue/dark copy cursor). As before, move the green/light set of brackets so that they just surround the rectangle you have drawn. (Do not make the cursor too big yet).

Now split the brackets up by pressing <DEL> and move the blue/dark set of brackets to the centre of the screen. If you were to press <ENTER> now, you would make a copy of your rectangle. However, before pressing <ENTER>, press the '2' key. This stands for double size. The blue/dark set of brackets now doubles in size. Press <ENTER> now and see what happens. The computer should have copied the rectangle into the blue/dark brackets at double it original size.

Try repeating the above instructions, but press '3' instead of '2'. '3' stands for treble size. Try once more using '4' for quadruple size.

The keys '1' to '4' set the magnification factor where '1' is the same size as original.

When you are defining the positions of the two cursors try setting magnification '4', then increasing the physical size of the cursors by pressing SHIFT and the cursor keys. As you increase the size of the green/light 'FROM' cursor, the size of the blue/dark 'TO' cursor grows accordingly. When the blue/dark cursor gets too big it will revert to a lower magnification, eventually the two cursors will be the same size. Because of this you may need to do some copies in two or more stages. However if you then shrink the physical size of the green/light set of brackets, then the blue/dark set will attain the maximum magnification up to that which you have set.

1.7.2 HOW TO REDUCE AN OBJECT

The method used to shrink an object is similar to expanding an object but you need to use the second ikon on the copying sub-menu.

NOTE: In this mode the area of screen is copied from the blue/dark set of brackets to the green/light set. Do not mix up the two or else you may lose your picture by copying a blank space onto your picture!!!!

Draw a large rectangle in the centre of the screen using the thickest pencil (the fourth ikon on the pencil menu) (see section 1.6.2). Now enter the copying sub-menu and select the second ikon, (a large blue/dark cursor with an arrow pointing to a smaller green/light set). The menu will disappear as normal. The green/light cursor is still the smaller of the two cursors. Press a number '1' to '4' for reduction factor. '4' is four times smaller while '1' is the same size. Now press <DEL> to allow you to move the blue/dark cursor around your rectangle, use the <SHIFT> key and the cursor keys to enlarge the size of the brackets.

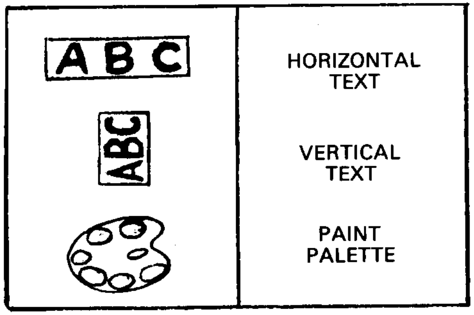
Use the <CTRL> key and cursor keys to reduce the size of the brackets.Now press <DEL> again and move the green/light set of brackets to where you want the area of screen to go. Press <ENTER> and the rectangle will be shrunk into the green/light copy cursor.

It does not matter if the two sets of brackets overlap, however you may lose the original shape if the 'TO' cursor covers it.

The best way to get to know the copying system is by trial and error. You may wipe out a few drawings on the way but this is better than losing a screen you may have spent hours designing. Remember if you are unsure, press <ESC> and start again rather than forgetting whether you are in expand or shrink mode! Once you can use the copy cursors skilfully you will be surprised just how useful and 'simple' they become.

1.8 ENTERING TEXT

If you want to label any of your drawings then use the text feature. When you are in the fifth menu, select the ABC ikon, and p.ess <ENTER> to confirm your choice. A sub-menu containing three ikons appears:-



Select a suitable colour/tone using the third ikon, then select the first ikon for horizontal text. After you press <ENTER> the menu will disappear and a single block cursor (like the AMSTRAD'S BASIC one) will appear this represents where text will appear. Move the cursor by holding <COPY> down, and moving the lightpen around the screen. You can finely place the text by using the cursor keys to nudge the cursor one pixel at a time. Now type what you want to write using the keyboard. All the standard characters are available as normal. The text will appear on the screen but if you want to adjust its position, simply use the same method as outlined above. In this way text can be finely positioned and the amount of required space can be gauged. Once you are happy with the text, press <ENTER> and the computer will ink in the text permanently. If you make a mistake, press <DEL> to remove the letters one by one. Pressing <ESC> will abort and any text not inked in will be lost.

The same applies for vertical text. Try selecting the second ikon on the sub-menu.

Interesting effects can be obtained by writing text twice, offsetting the second piece by one pixel vertically and horizontally using the cursor keys to nudge the test cursor. Try writing in two different colours to give the text a feeling of depth.

To remove text that has been permanently inked in, re-write the same text in the background colour.

You may wish to use characters from the Amstrad character set which are not available from the keyboard. For example characters 160 and 161 are useful for french text. To enter these characters you need to look up the character's code in your Amstrad manual. When you require a special character, while entering normal text press <TAB> and then enter the character's code using THREE numerical digits. For example the character number 160 is entered as:-

```
<TAB>160
```

You do not need to press <ENTER> for the character to appear. Pressing <DEL> will delete the character as normal. NOTE: even if the code of the character is one of two digits, you still need to type out THREE digits, using preceding zeros if necessary. For example the character number 4 is entered as:-

```
<TAB>004
```

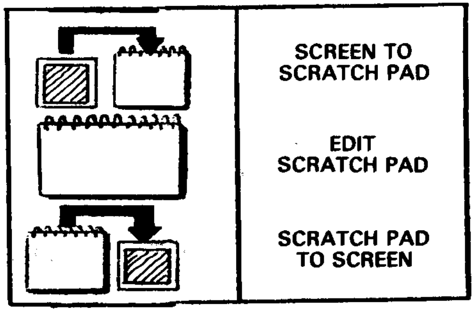
If you make a mistake in entering the digits, then press space and start again by pressing <TAB>.

1.9 FINE DETAIL AND ALTERATIONS

When you are trying to draw very fine detail on the screen, where every pixel may be in a different colour, it would be nice if a section of screen could be magnified to a suitable size.

For this reason a 'scratch pad' has been used to allow very fine detail to be applied to the whole screen.

Draw a small circle on the screen (one with a radius of about 6 pixels). (See section 1.6.1) Now select the last ikon in the fifth menu (the one that looks like a pad). Another sub-menu will appear as follows:-



Use the top ikon and press <ENTER>. (This represents a screen to pad). The menu will disappear and a red or dark bracket cursor will appear similar to the copy cursor mode (see section 1.7). These brackets can be moved using the <COPY> key and the lightpen, or by using the cursor keys to nudge the brackets to a single pixel. Notice that this set of brackets is a single size only. Move the brackets so that they surround the circle. (If your circle is too big, then you may need to re-draw it). When you have the circle in the centre, press <ENTER>. the section of screen is now 'blown' up to a size where fine detail can easily be done.

At the very bottom of the screen is a band containing the ink colour. Press the keys '1' through to '0' on your keyboard to select a colour within the scratch-pad ONLY. If you press the key labelled <CLR> then a special transparent colour will be used (sown as vertical bars). This colour will not be copied from the pad and hence will let the original colours on the screen show through. Now point the lightpen at the screen by pressing <ENTER>, you will be able to plot pixels in the current colour. Use white to blank out any mistakes.

Perhaps easier to use than the lightpen in this mode are the cursor keys. Use these to move the flashing cursor around. Once they are pressed they lock out the lightpen until <COPY> is pressed.

Using the lightpen or the cursor keys, alter the circle. Now press <ESC> and the sub-menu will reappear. Now select the third ikon (this represents pad to the screen) and press <ENTER> to confirm your choice. The red brackets reappear and pressing <ENTER> copies the contents of the scratch-pad to the area of screen inside the brackets. If you don't move the brackets, then your circle will be 'updated'. If you move the brackets by holding down <COPY> and using the lightpen or by nudging the brackets, then multiple copies of the circle can be created.

Now press <ESC> and select the second ikon on the sub-menu. This returns you to the scratch-pad. You could now re-edit the circle, or even start again.

If you want to create your own high-resolution object then clear the whole pad to a background colour: Select any colour using the '1' to '0' keys (a good idea would be to use white, key '1'), then press <CTRL> and <TAB> together. The screen will blank to the selected colour. Now select another colour and use the lightpen or cursor keys to draw a graphic object. Once you have finished press <ESC> and select the third ikon to place the contents of the scratch-pad onto the screen. Press <ENTER> and move the brackets around. Press <ENTER> again to place the object anywhere on the screen.

You may already have a complicated background onto which you want to place an object without the background being removed around the object. If this is the case then simply design your high-resolution object on a transparent background (selected by pressing <CLR>). If you have picked up an object from the screen with a set background then whilst in edit mode change this background to transparent <CLR> or select the same colour as the background and press <SHIFT> and '*'. This is to save you the trouble of manually changing all the background to clear.

The use of the scratch-pad, although seeminly complicated helps to make alterations to parts of the screen very quickly and accurately. Practice using the pick-up, edit and put-down sequence to edit your pictures. Also be careful not to forget you are in put-down mode when you try to pick something up, the result is the writing over of what you originally wanted to edit (and much displeasure to say the least!).

1.10 USING TAPE AND DISC DRIVES

The graphics package supports both tape and Amstrad disc drives. If you have no disc drive connected then the software acknowledges the fact and does not allow you to try and use discs.

There are two sections about tape and disc which will be of interest, and those are (1) saving the Graphics Package itself (including putting the software onto disc from the tape) and (2) saving pictures you create.

The first one is considered next.

1.11 BACKING-UP YOUR GRAPHICS PACKAGE (cassette version)

The software is NOT protected because users with disc drives may want to transfer the software to their discs. Secondly the software is saved using SPEED WRITE 0; users may wish to resave a copy in SPEED WRITE 1. Finally keeping a back-up is always recommended, and saves the trouble of ordering a second copy when you leave the tape on a magnet!

1.11.1 TO LOAD WITHOUT RUNNING:- (cassette version)

In all cases, the Graphics Package needs to be loaded into the computer as oultined below.

1. If your disc is attached then type '|Tape' and press <ENTER>.

2. Type 'LOAD "LPEN"' and press <ENTER>.

3. Put the tape in the tape drive and rewind the tape. Press <PLAY> then press any key. Do not release the <PLAY> key unless a TAPE ERROR occurs, in which case refer to the Amstrad manual for loading hints.

4, When the computer has finished type '35 STOP' and press <ENTER>,

5, Now type 'RUN' and press <ENTER>,

6, The tape will start to run again and about four minutes should pass,

7, Finally type '35' and press <ENTER>,

The complete program is now in the computer,

1,11,2 TO SAVE TO TAPE:– (cassette version)

1, Select the speed of tape you want to use:–

```
Type 'SPEED WRITE 0' for slow, safe SAVE,
Type 'SPEED WRITE 1' for fast SAVE,
```

2, Place a blank cassette into the drive, It needs to have at least five minutes of tape on it (or left on it),

3, Type:–

```
SAVE "LPEN": SAVE "GRAPH", B, 6000, 7600: SAVE,
"PROG", B, 32768, 9750 and press <ENTER>,
```

4, Press <PLAY> and <RECORD> then any key, You will need to press any key twice again for each section,

1,11,3 TO SAVE TO DISC:–(cassette version)

1, Type '|DISC' and press <ENTER>,

2, Put a disc in your drive which has at least 19K free on it, Make sure the write protect tab is off,

3, Type:–

```
SAVE "LPEN": SAVE "GRAPH", B, 6000, 7600: SAVE
"PROG", B, 32768, 9750 and press <ENTER>,
```

NOTE: You may wish to use SPEED WRITE 1 when saving pictures (see section 1,12), If so, before saving a copy of the program onto tape or disc, add this line:–

```
35 SPEED WRITE 1
```

Now save the program as outlined above,

1,12 LOADING AND SAVING

Once you have completed a picture, it can be stored on tape or disc for later use (see section 1,14), The very first menu contains the ikons which control saving and loading, Press <ESC> as many times as needed to return to the first menu, Whether you have a disc drive or not, the first two ikons represent tape to computer (Load), and computer to tape (Save), If you have a disc drive attached then you have the option of using tape or disc and there are two more ikons representing disc to computer (Load), and the computer to disc (Save),

To save your picture select the second ikon (tape) or the fourth ikon (disc only), The computer will ask you for a name, to be typed from the keyboard, Only letters and numbers are allowed, If you use disc then a directory containing any other pictures and amount of free memory will be displayed, (If you wanted directory only, press <ESC> to abort at this stage,) Once you type in the name, the computer will save the whole picture, For tape this will take some time, and the computer will inform you haw many blocks it has saved, On disc, the computer will return to the main menu after a short time, unless an error occurs, If a disc error does occur then a message will be displayed, you may either correct it for non-fatal errors eg, read errors etc, or press any key and the first menu will reappear then you may need to change the disc and re-save to picture,

It is wise to save complicated pictures frequently (especially if you have a disc) in case you make a mistake, (Like filling the screen in white!) It takes only two minutes to save a screen using SPEED WRITE 1 (see section 1,11) or three and one half minutes using SPEED WRITE 0,

On the disc drive, pictures may be saved with the same name and the computer will rename the old version using the BAK extension, Consult your manual if you are unfamiliar with the AMSTRAD Disc Operating system,

To Load the picture from tape select ikon one, or ikon three if you are using disc, If you are using tape then the computer will load the first binary file on the cassette onto the screen, This need not have been one saved by this package, If all goes well, eight blocks will load into the computer, The computer will report on its progress as normal, Any errors will be reported and the action you need to take depends upon the error eg, if the disc read error occurs then the message "try Retry or Cancel" is displayed,

Note that the files which are output are standard binary files and can be used in your own BASIC or machine code programs, (See section 1,11 and 1,12,)

1.13 USING PRINTERS

To obtain a hard copy of your pictures (or any others you may have created without this package), it is necessary to load the second program on the tape supplied with your Graphics Lightpen. This program does not support any specific printer but it converts the colour data on the screen into a form which printers accept their data. This data can then be very quickly output from a BASIC program designed to suit YOUR printer. Two example BASIC programs are included for EPSON and AMSTREAD printers. From these and your own printer manual it is hoped that any printer, which can be controlled normally from your computer, can be used to produce a screen copy.

Also those users with expansion parallel printer ports, many of which use eight bits instead of seven, can very easily use these ports instead of the Amstrad one. This is possible because all the printer handling is actually done from BASIC.

1.13.1 EPSON PRINTER EXAMPLE:-

Before you can make any copies you need to have saved your picture on tape or disc. Now reset the computer and follow these instructions:-

1. Connect up your printer and make sure that it is working correctly,also connect the disc drive if you are going to be using discs.

2. Switch on and type 'MEMORY 9999' and press <ENTER>.

3. Load the second program on the software cassette. It is found at the end of either side. Fast forward the cassette until you are nearly at the end of the cassette and type 'LOAD "DUMPMC"' and press <ENTER>. If the program fails to load then rewind the tape and try again.

4. Type in the following program making sure you copy it exactly:-

```
 10 MEMORY 9999;MODE 0
 20 S=5;POKE 40003,S;POKE 40004,0
 30 INPUT "NAME OF SCREEN";Y$;LOAD Y$
 40 CALL 40000;REM REMEMBER TO LOAD DUMPMC FIRST! (STAGE 3)
 50 WIDTH 255;PRINT #8, CHR$(27);"A";CHR$(4);
 60 D=25; IF S>2 THEN D=50
 70 L=S*2+2;IF S>2 THEN L=L-6
 80 C=10000;FOR J=1 TO D;LA=C;FOR I=1 TO L
 90 PRINT #8,CHR$(27);"K";CHR$(80);CHR$(0);;FOR K=1 TO 80
100 PRINT #8,CHR$(PEEK(C)/16);;C=C+1;NEXT K,I
110 PRINT #8;C=LA
120 FOR I=1 TO L;PRINT #8,CHR$(27);"K";CHR$(80);CHR$(0);
130 FOR K=1 TO 80;PRINT #8,CHR$(PEEK(C) AND 15);
140 C=C+1; NEXT K,I;PRINT #8;NEXT J
150 END
```

5. Type 'RUN' and press <ENTER>.

6. In reply to the prompt 'NAME OF SCREEN' type the name under which you saved the screen, or just press <ENTER> if you are using tape.

7. The screen will load and after a pause the printer will print out your screen using a shading effect to show the different colours.

8. (You may wish to SAVE the above program yourself to save retyping it every time you need to use it).

1.13.2 AMSTRAD PRINTER EXAMPLE:-

The procedure is almost the same as for the EPSON printer except that the program for step 4 is as follows:-

```
 10 MEMORY 9999;MODE 0
 20 S=5;POKE 40003,S;POKE 40004,1
 30 INPUT"NAME OF SCREEN";Y$LOAD Y$
 40 CALL 40000
 50 WIDTH 255
 60 DIM D(8,5);FOR I=1 TO 8;FOR J=1 TO 5; READ D(I,J); NEXT J,I
 70 DATA 127,1,0,1,1 ,128,128,63,2,0 ,192,64,31,4,0
 80 DATA 224,32,15,8,0 ,240,16,7,16,0 ,248,8,3,32,0
 90 DATA 252,4,1,64,0 ,254,2,0,1,0
100 C=10000;S=1;L=1
110 FOR I=1 TO 6;PRINT #8,CHR$(27);"K";CHR$(0);CHR$(8);
120 FOR J=1 TO 80;PRINT #8,CHR$((PEEK(C) AND D(S,1))/D(S,2)+(PEEK(C+480)
    AND D(S,3))*d(S,4));;C=C+1;NEXT J,I
130 PRINT #8;C=C480*D(S,5);S=S+1;IF S=9 THEN S=1
140 L=L+1;IF L=58 THEN STOP
150 GOTO 110
```

1.13.3 ADVANCED INSTRUCTIONS FOR PRINTOUTS

While the BASIC program in the two above examples looks complicated, most of the hard work of converting the screen into data has already been done by the machine-code program you first loaded. The BASIC program simply outputs the data to the printer after sending the respective printer's control codes. If you have one of the above printers then you need not worry about the next section, however you may have a different printer or simply be interested so let's go through the EPSON printer program step by step to see what is happening:-

After setting MODE 0, the program sets two variables in the machine-code program. The variable S sets the size of printout. Location 40003 is then altered to that value to inform the machine code of the size you want.

On the EPSON there are a possible 480 (in single density mode) pixels across the paper. This means that is the capable of displaying the largest size of output possible from the machine code program. On other printers however there may be less pixels across and hence a smaller size may need to be used. Try different values of S (0 to 5) on your EPSON printer to illustrate the effect.

The screen has dimensions of 160 pixels across the screen and 200 pixels down the screen. The machine code software can convert these to six different sizes on the printer. The possible dimensions are:-

S = 0 160 pixels across and 200 pixels down
S = 1 320 pixels across and 200 pixels down
S = 2 480 pixels across and 200 pixels down
S = 3 160 pixels across and 400 pixels down
S = 4 320 pixels across and 400 pixels down
S = 5 480 pixels across and 400 pixels down

For your printer you will need to look up in your printer manual how many pixels your printer is capable of printing in one row. Then select a value of S that is smaller or equal to that maximum value.

As any printer prints, the head will print a number of dots in a VERTICAL column, so this is how the data from the screen is converted. Starting from location 10000 in memory, data is created where each byte represents one vertical column. The printer then has to be informed that the data is on its way, and the BASIC program needs to simply send the data from memory to the printer.

The location 40004 needs to contain either a 0 or a 1, '0' specifies that the most significant bit of the data is the bottom dot while '1' means that the most significant bit of the data is the top bit. If you are unsure, use 0 and it will be clear whether you need to use a 1 because the printout will be upside down!

Line 30 loads the screen from tape or disc. Note that the MODE 0 command makes sure that the screen has not scrolled. That, is the screen always starts at 49152 in memory.

Line 40 calls the machine code routine and fills the memory of the Amstrad from 10000 onwards, the actual amount of data depends on the size value S. For S=5, 24K of memory is needed.

Line 50 commands the EPSON to move just 4 pixels between every line. You may need to consult your printer manual to find out how to set the distance between rows. In some cases it may not be possible to set a 4 pixel line setting. In this case you may have to set a 7 pixel line setting and output 7 pixels of data at a time, meaning that the bottom row of pixels in each line will be lost.

It is necessary to note that the Amstrad's CENTRONICS parallel port uses only 7 bits. While this is industry standard, the EPSON and many other printers use 8 bits.

With only 7 bits you may realise that is impossible to send 8 pixels of data in one instruction to the printer. Hence on the EPSON (and more than likely on your printer) it is necessary to send the top four bits and then the bottom four bits seperately.

In line 60 a variable D is calculated. D stands for depth and is either 25 lines or 50 lines depending on which size (S) of prinout was selected in line 20. On the next line L is calculated. L is the number of blocks of 80 bytes of data. Hence if you use the largest size of printout (480 pixels) then value of L is 6. The data is output as blocks of 80 bytes and not as a block of 480 bytes for one reason. When the amount of data to be printed is sent to the EPSON, then EPSON requires a low and high byte value.

If 480 were converted to low and high byte values you would get 224 low and 1 high, ie (1*256 + 224 = 480).

Due to the 7 bit CENTRONICS port, it is impossible to send the value 224. Hence it is safer to send blocks of 80.

After all these variables have been created, the program sets C to 10000 to count through the memory from 10000 onwards.

There is a loop (J) going through each row up to 25 (or 50 if double height is used). Then there is a second loop (I) going through each block of 80 bytes.

Finally the printer is told that it is to print out 80 bytes of high-resolution data 'CHR$(27);"K";CHR$(80);CHR$(0);'. You will need to supply this from your own printer manual.

NOTE: Some printers use bit 7 being a 1 to designate high-resolution data. If this is the case with your printer, then your printer will not produce high-resolution with the AMSTRAD at all.

80 bytes are now read from memory and sent out to the printer. Note how the top 4 bits are selected by dividing by 16. Finally a carriage return and linefeed is sent.

Line 120 onwards repeats the same data but this time the bottom 4 bits are sent by masking out the high 4 bits. (Do not worry if you are unfamiliar with '/16 and 'AND 15', just believe they seperate the top and bottom 4 bits!)

So that's all there is to it!!! If you are confused, try reading your printer manual thoroughly, especially the sections on high-resolution printing. Try using some of the examples in the manual and change the EPSON program to contain sections relevant to your printer. Also instead of keep loading a screen from tape or disc, delete line 30 and replace it with this line:-

```
30 FOR I=1 TO 10: FOR J=1 TO 15:PEN J:PRINT CHR$(64+J);:
   NEXT J:PRINT:NEXT I
```

This will print some different coloured letters on the screen as a test screen.

The authors sincerely hope that presenting the printer routines in this form will allow greater flexibility to use any printer which supports high-resolution graphics.

We hope that it can be appreciated that there are hundreds of different printers which can be used with your Amstrad and a single printer routine could never support all of them. The above information should make it possible to support your printer, with as little effort on your part as possible.

1.14 USING SCREENS IN YOUR OWN PROGRAMS

The screens which are saved from the Graphics Package as standard binary files.

These can be loaded into your programs simply by using a line such as;-

```
40 MODE 0
50 LOAD "NAME"
```

Where the 'NAME' is what you called the screen when you saved it.

The colours used in the Graphics Package are not the same as the colours assigned when the computer is first switched on. Hence you need to re-define the INKS before you see the pictures in the correct colours;-

```
10 BORDER 0;RESTORE 20;FOR I=0 TO 15;READ A;INK I,A;NEXT I
20 DATA 26,18,9,17,8,11,13,22,0,24,6,2,14,19,2,3
```

If you like you can load the screen into other areas of memory by adding a load address on the end of the above statement:-

```
30 MEMORY 19999
40 LOAD "NAME",20000
```

This will load in a screen at 20000 in memory. If you are doing this, remember that your BASIC program has less memory than normal.

You could even write a short machine code program to move the picture to the screen when you need it like the one below:-

```
60 MEMORY 19900;RESTORE 70;FOR I=19950 TO 19961;READ A;POKE I,A;NEXT I
70 DATA 33,32,78,1,0,64,17,0,192,237,176,201
90 CALL 19950
```

This will move the screen which has been loaded at 20000 onto the screen. You may even like to blank all the inks before you make the transfer to make the picture appear immediately.

```
80 FOR I=0 TO 15;INK I,0;NEXT
```

Then switch INKs back on again;-

```
100 RESTORE 20;FOR I=0 TO 15;READ A;INK I,A;NEXT I
```

If you own a disc drive then you could write a program similar to the one below which presents a number of pictures on screen almost like a slide show.

```
 10 MEMORY 16300;POKE 16350,62;POKE 16351,192;POKE 16352,195;
    POKE 16353,&08;POKE 16354,&BC;CALL 16350
 20 MODE 1;INK 0,0;INK 1,22;INK 2,2;INK 3,6;PEN 1;PAPER 0;BORDER 0
 30 PRINT"                        DISPLAY PROGRAM"
 40 PRINT"                       _________________"
 50 PRINT"
 60 S=5;REM NUMBER OF SCREENS TO BE DISPLAYED
 70 DIM N$(S)
 80 N$(1)="HOUSE";REM NAMES OF SCREENS ON DISC
 90 N$(2)="SHIP"
100 N$(3)="SPEECH"
110 N$(4)="CHART"
120 N$(5)="CAR"
130 PEN 2
140 PRINT;PRINT;PRINT;PRINT "DELAY BETWEEN SCREENS (10 - 6000
    SECS)";PRINT;PEN 3;INPUT "
150 IF D<>INT(D) OR D<10 OR D>6000 THEN 140
160 D=D-5
170 MODE 0;RESTORE 170;FOR I=0 TO 15;READ A;INK I,A;NEXT I;
    DATA 26,18,9,17,8,11,13,22,0,24,6,2,14,19,2,3
180 NS=0;FT=1
190 LOCATE 1,13;PRINT " SCREENS FOLLOW!"
200 FOR I=1 TO S
210 SL=16384;IF NS=1 THEN SL=49152
220 T=TIME;LOAD N$(I) ,SL;IF FT=1 THEN FT=0;GOTO 240
230 WHILE(TIME-T)/300<D;WEND
240 POKE 16351,192;IF NS=0 THEN POKE 16351,64
250 CALL 16350;REM SWAP SCREEN BEING DISPLAYED
260 NS=NS+1;IF NS=2 THEN NS=0
270 NEXT I;GOTO 200
```

1.15 USING THE GRAPHICS LIGHTPEN IN YOUR OWN PROGRAM

Your program could be as complicated as this Graphics Package or as simple as a telephone directly program, but both are improved by using a lightpen to save using the keyboard for choices.

Listed below are two programs, one in BASIC and one in machine code to allow you to read the position of the lightpen on the screen.

IN BASIC:-

```
10000 OUT &1C00,17:L=INP(&1F00):OUT &1C00,16:H=INP(&1F00)
10010 L=H*256+L-12292:Y=L/40:X=L-Y*40
10020 RETURN
```

A GOSUB 10000, would return X and Y coordinates of the position of the lightpen. X ranges from 0 to 39 and Y ranges from 0 to 24. This scale is irrespective of which mode you use, although when in MODE 1, each graduation represents one character square.

(The value 12292 may have to be altered by one or two to suit your monitor or television.)

IN MACHINE CODE:-

```
COORDINATES: LD BC,1C00H  ; set register pair to 1C00
             LD A,17
             OUT (C),A
             LD BC,1F00H
             IN L,(C)     ; read low order position
             LD BC,1C00H
             LD A,16
             OUT (C),A
             LD BC,1F00H
             IN H,(C)     ; read high order position
             LD DE,12292
             AND A
             SBC HL,DE    ; remove constant offset
             LD BC,0
             LD DE,40     ; divide by 40
LOOP:        AND A
             SBC HL,DE
             JR C,FINISH
             INC C        ; C is Y position
             JR LOOP
FINISH:      ADD HL,DE
             LD B,L       ; B is remainder =X
             RET
```

This routine returns the X and Y coordinates in the BC register pair, where B=X and C=Y.

The two routines above could simply be included in a program which uses menus or need to select choices from anywhere on the screen.

EXAMPLE:-

The program below is a simple reflex tester. Type it in carefully then RUN it.

After the title disappears, four boxes will be drawn. The word HERE will appear in one of the boxes and you need to point the lightpen to that word as quickly as you can. Ten times the computer will select a box at random. The total time will then be given, the lower the score, the better you have done. Our lowest was 2.85 seconds!

Try changing parts of it, if you want, or experiment with writing programs of your own.

```
   10 MODE 1:BORDER 0:INK 0,26:INK 1,0:INK 2,12:INK 3,20
   20 PRINT , ,"    TEST YOUR REFLEXES"
   30 FOR I=1 TO 2000:NEXT:CLS
   40 C=1:FOR I=1 TO 25 STEP 2:C=-C*(C<>3)+1:FOR X=24 TO 344
      STEP 320:FOR Y=16 TO 216 STEP 200:MOVE X+I,Y+I:DRAW
      X+I+250,Y+I,C:DRAW X+I+250,Y+I+150,C:DRAW X+I,Y+I+150,C:
      DRAW X+I,Y+I,C:NEXT Y,X,I
   50 PEN 1;V=5:FOR N=1 TO 10:REM NUMBER OF TESTS
   60 V1=INT(RND*4):IF V=V1 THEN 60
   70 FOR I=1 TO INT(RND*3000)+1500:NEXT I
   80 V=V1:X1=9+(V AND 1)*20:Y1=7+(V AND 2)*6:LOCATE X1,Y1:
      PRINT "HERE!"
   90 T=TIME
  100 LOCATE 18,13:PRINT INT(TIME-T)/3)/100;"   ":GOSUB 10000:IF
      (Y-1<Y1 AND Y+3>Y1) AND (X-5<X1 AND X+1>X1) THEN 120
  110 GOTO 100
  120 LOCATE X1,Y1:PRINT"       ":REM 5 SPACES
  130 TT=TT+(TIME-T):NEXT N
  140 LOCATE8,13:PRINT"YOUR TIME WAS";INT(TT/3)/100;"SECONDS."
  150 for I=1 TO 3000:NEXT:LOCATE 1,13:
      PRINT"                             ":REM 5 SPACES
  160 RUN 50
10000 OUT &1C00,17:L=INP(&1F00):OUT&1C00,16:H=INP(&1F00)
10010 L=H*256+L-12292:Y=L/40:X=L-Y*40
10020 RETURN
```

1,16 REVIEW OF GRAPHICS PACKAGE COMMANDS

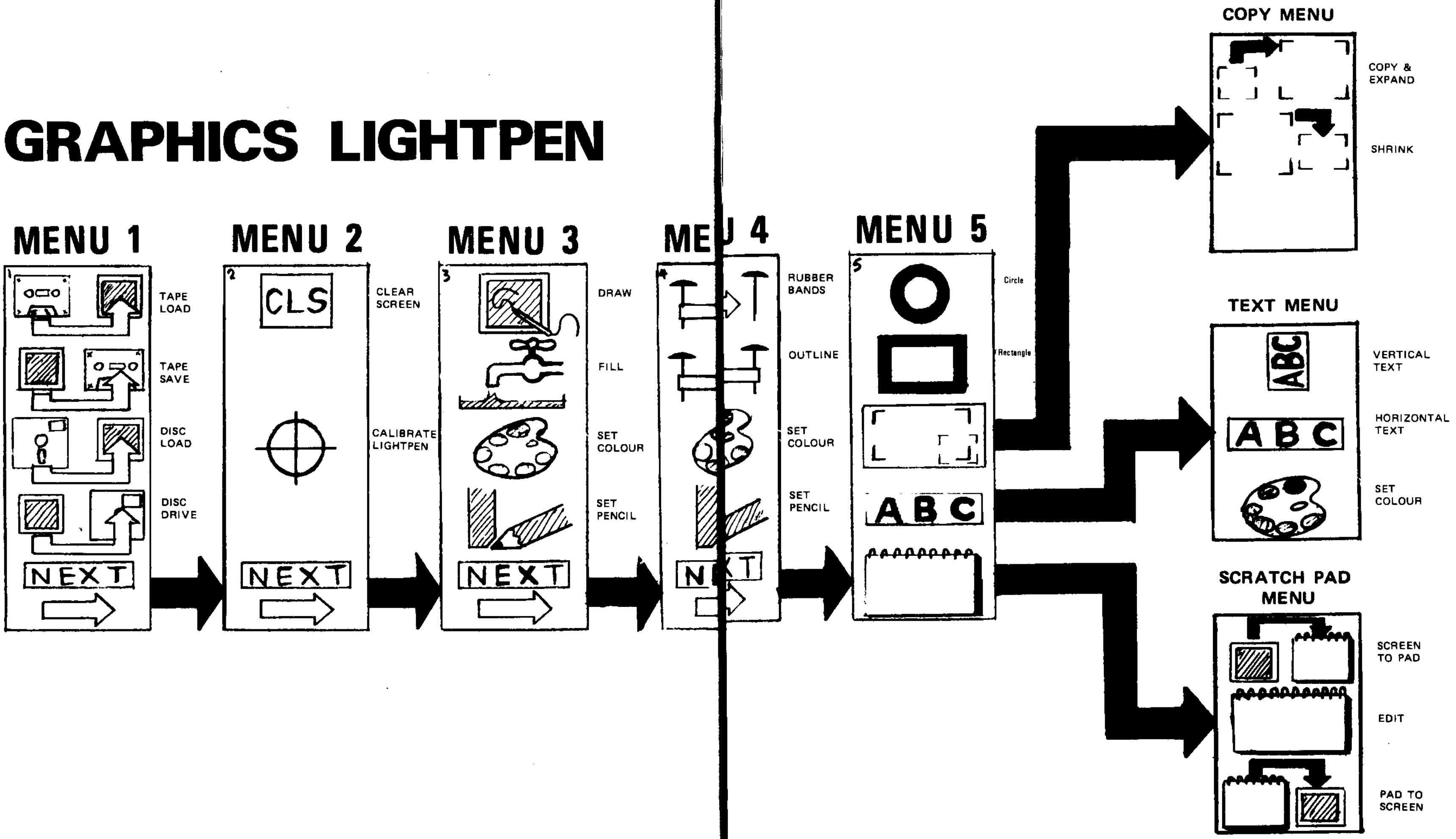
This section is intended as a quick reference once you have gone through the rest of the manual, Use it in conjunction with section 1,17 showing all the menus and the menu numbers,

Menu 1:	Ikon 1	Load Screen from Tape	(Section 1,12)
	Ikon 2	Save Screen to Tape	(Section 1,12)
	Ikon 3	Load Screen from Disc	(Section 1,12)
	Ikon 4	Save Screen to Disc	(Section 1,12)
	Ikon 5	Goto Menu 2 (Ikons 3 and 4 will be missing on tape only systems)	
Menu 2:	Ikon 1	Clear Screen to White Background	(Section 1,3)
	Ikon 2	Calibrate Lightpen	(Section 1,3)
	Ikon 3	Goto Menu 3	
Menu 3:	Ikon 1	Draw Mode (Use <ENTER> and SPACE BAR)	(Section 1,3)
	Ikon 2	Fill Mode (Use <ENTER> to fill area)	(Section 1,4)
	Ikon 3	Select Ink Colour/Tone	(Section 1,3)
	Ikon 4	Select Pencil Texture and Size	(Section 1,3)
	Ikon 5	Goto Menu 4	
Menu 4:	Ikon 1	Rubber Bands (Use <ENTER> for pins <COPY> to move rubber bands out, <ESC> to finish)	(Section 1,5 & 1,6)
	Ikon 2	Outline Rubber bands	(Section 1,5)
	Ikon 3	Select Ink Colour/Tone	(Section 1,3)
	Ikon 4	Select Pencil Texture and Size	(Section 1,3)
	Ikon 5	Goto Menu 5	

Menu 5:	Ikon 1	Draw Circles Use first pin for centre, second pin to set radius)	(Section 1,6,1)
	Ikon 2	Draw Rectangles (Use 2 pins to set diagonal)	(Section 1,6,2)
	Ikon 3	Copy, Shrink and Expand	(Section 1,7)
	Sub Menu:	Ikon 1 Expand or Copy (Set magnification '1' to '4' then copy from green/light to blue/dark cursor)	(Section 1,7,1)
		Ikon 2 Shrink (Set reduction factor '1' to '4' then copy from blue/dark to green/light cursor)	(Section 1,7,2)
	Ikon 4	Entering Text (Select horizontal or vertical text from sub menu, Set colour if you want)	(Section 1,8)
	Ikon 5	Scratch Pad	(Section 1,9)
	Sub Menu:	Ikon 1 Screen to Scratch Pad (Move the cursor then copy area of screen to scratch pad by pressing <ENTER>)	(Section 1,9)
		Ikon 2 Edit Scratch Pad (Select a colour using keys '1' to '0', then use <ENTER> to plot pixels)	(Section 1,9)
		Ikon 3 Scratch Pad to Screen (Press <ENTER> to copy scratch pad to area of screen outlined by cursor)	(Section 1,9)

1.17 MENU SCHEMATIC

GRAPHICS LIGHTPEN



SPEECH SYNTHESIZER

The DK'tronics speech synthesizer uses the popular SP0256 speech chip, which gives it an almost infinite vocabulary. It is supplied with a text to speech convertor for ease of speech output creation. Every thing you wish to be spoken is entered in normal English without having to use special control codes or characters. It is therefore extremely easy to use. The voicing of the words is completely user transparent and the computer can carry on with its normal running of a program while the speech chip is talking. The speech output from the SP0256 is mono and directed to both speakers via a powerfull stereo amplifier.

STEREO OUTPUT

To utilise the Amstrad stereo output on the back of the computer the interface has its own built in stereo amplifier which is connected to two high quality 4" speakers. This gives all sound output a totally new dimension and greatly improves the sound quality and volume over the computer's internal speaker. All programs that use sound in any way will now be output through the interface. The interface is also supplied with volume and balance controls.

SPEECH SYNTHESIS

The speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an almost unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chip's internal ROM.

TEXT TO SPEECH

Although there are only 26 letters in the alphabet, letters have a totally different sound when used in different words. For example, the "a" in "hay" is much longer and softer than in "hat". When you speak you automatically make adjustments because you know just how the word should sound. This is not quite so easy with a computer.
The machine code software supplied is mainly developed to this mode of operation, of which 3.5k is used for tables which contain the rules and exceptions to the English Language. For example I before E except after C. This therefore allows the user to enter words to be spoken in normal English.

NEW BASIC COMMANDS

There are 10 new Basic commands which control all the functions of the interface, making the synthesizer very easy to use, ie to say "Amstrad" you would enter: 10 PRINT "'Amstrad'". You can even control the speed at which it talks, or use the synthesizer to create sound effects on a fourth sound channel.

SPEECH SYNTHESIZER MANUAL

WARNING

This unit must be used in accordance with these instructions. Never plug in or remove the interface without first disconnecting the power from the computer. Failure to follow these instructions may result in damage to the interface or the computer.

2.1 INSTALLATION

The jack plug on the short flying lead must be connected to the stereo output on the rear of the computer next to the joystick socket. The speech interface can only be connected to the floppy disc/ expansion port. The speakers are plugged into the left and right sockets on the interface, the left socket is the left channel, the right socket is the right channel (as viewed from the front of the interface). The computer can now be switched on.

2.2 SETTING UP AND USING THE STEREO AMPLIFIER

The volume control is located the right hand side of the interface for the CPC 464 and on the left hand side of the interface for the CPC 6128. To check the volume setting type in the following line of program as a direct command.

```
SOUND 2,50,3000 (Enter).
```

You can now adjust the volume to the required level.

Just below the volume control is a small hole, this is the balance between the left and the right channels and is factory set and should not require adjustment.

To check the balance type in the following program.

```
10 SOUND 1,50,100:REM Left channel.
20 SOUND 4,50,100:REM Right channel.
30 GOTO 10

RUN <ENTER>.
```

You should hear the tone change between the left and right speakers. If it is necessary to adjust the balance you should use a small screwdriver. The internal speaker on the Amstrad should be turned down (right-hand end for the 464, rear right for the 6128 computer) as the mono output could distract from the effects of the stereo output.

Any sound that previously came out of the mono internal speaker will now be sent out via the interface in stereo. All programs that use the sound in any way (ie commercial software) will now output through the interface.

2.3 SPEECH SYNTHESIS

The Amstrad speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chips internal rom. In the past, Speech Synthesis has required large data bases to store words because every word had to have its own set of data. This method of speech synthesis is slightly clearer than the SP0256 speech chip but would require at least 5 mega bytes of memory to store the English Language. This is obviously not practical. Therefore, for the home computer the SP0256 is the ideal chip.

2.4.1 SOFTWARE LOADING DETAILS (cassette)

The software cassette supplied with your Amstrad Speech Synthesizer has been recorded in speed write 0 and speed write 1. To load press the green control key and enter (the numeric key pad enter). Note that if you are using the CPC 664 then you will require an external cassette player

The software is in two parts the first part is the relocater, this enables you to load and run the software in the 16384 to 39000 part of the basic memory map. This is because other software may be running in high memory (ie disc software). The second part is the machine code to run the text to speech converter, this is 4K in length.

After the first part has loaded the screen will clear and ask you for the load address. This should be a high address in the range of 16384 to 39000 (39000 is a good address). The routines will automatically lower HIMEM so that basic will not run into the machine code it is important to remember the load address if you intend using speech from machine code.

Once the second part has loaded the machine code will perform the initialisation routine and print the copyright message on the screen.

2.4.2 SOFTWARE LOADING DETAILS (ROM)

On power-up the Rom should identify itself with the sign-on message :-

```
        SPEECH ROM VER. 1.1
```

All that is required to initialise the Speech Rom is to type in '|speak' then press <ENTER>.

This will also test the unit by saying "DK'tronics speech synthesizer" and displaying the extra commands.

2.5 BASIC COMMAND EXTENSIONS

There are 10 new basic commands and an example of their use is given below.

|SPON Turn on the interrupt to read the buffer. (speech on)

|SPOF Turn off the interrupt and stop reading the buffer. (speech off)

|FEED,n This is used to feed the speech buffer direct and also for sound effects. The |FEED,n command is followed by a maximum of 30 data items seperated by commas.

|FLUS This command clears the speech and text buffers.

|SPED,n This command controls the speed at which the words are spoken. n, is a number from 0 to 15.

|OUTM,1 Sets access to 'text to speech' using print "'xxxx'" only, ie listing etc will not be spoken.

|OUTM,2 Sets access to 'text to speech' from all print outputs. Anything that outputs to the screen ie listings, Syntax errors, ready will be spoken. (BUT WITHOUT PRINTING THEM ON SCREEN).

|OUTM,3 As |OUTM,2 but text is spoken and printed on the screen.

NOTE: |OUTM,2 and |OUTM,3 can only be stopped by the break key.

Note during extensive printing in |OUTM 2 or 3, the computer may seem to be unresponsive to the Break key. This is due to the speech buffer being full and Basic waiting until there is some more space.

To get out of this keep the BREAK key pressed until Basic scans for it, then the computer will print BREAK followed by READY while continuing to say the contents of the buffer which may continue for up to a minute.

The two remaining commands are only available on the Rom version. They are:-

|LEFT, Returns the value of the free memory in the speech buffer. This is used as follows :-

```
1000 a%=0
1010 |left,@a%
1020 print a%
```

With the buffers empty the result would be 250 bytes.

|SAY, This command is the normal way of getting the unit to speak. It uses the text to speech convertors. The correct syntax for the command is:-

```
664, 6128 computers:        |say,"the cat sat on the mat"

464 computers:       a$="the cat sat on the mat";|say,@a$
```

2.6 SPEECH SYNTHESIZER

The speech synthesizer can be used in various modes.

a) DIRECT- WITHOUT SOFTWARE SUPPLIED.
b) USING |FEED COMMAND.
c) USING THE TEXT TO SPEECH CONVERTER.
d) USING PRINTING MODE COMMANDS.

2.6.1 SPEECH CONTROL DIRECT FROM BASIC

The speech chip is in the I/0 memory map at location &FBFE. It is possible to send data straight to this location but the correct allophones must be worked out and converted to data. Also the program must find out when the speech chip has finished saying each allophone by reading from I/O locations &FBFE and waiting until its value is less than 128.

The following simple example shows a program to do this, to send this data to the speech chip.

```
10 FOR X=1 TO 8:REM Length of data statement.
20 IF INP(&FBFE)>127 THEN 20:REM Wait
   for chip ready signal.
30 READ A
40 OUT &FBFE,A
50 NEXT X:STOP
60 DATA 26,16,55,17,39,26,21,0:REM Amstrad
```

This example is using the speech chip in its crudest form and requires that the component parts of the text to be spoken are converted to data. To do this use the allophone table on page 47, also see the dictionary on page 49. An example of the word "COMPUTER" is:-

```
WORD  C   O   M   P  U   T   E   R
DATA  42, 15, 16, 9, 49, 22, 13, 51, 0
```

NOTE: That at the end of the data we send a 0 to the speech chip, this is to stop the last sound from sounding forever.

2.6.2 USING THE |FEED COMMAND

This requires that the software supplied is first loaded (cassette version).

The |FEED command is an extended basic command. This command allows you to enter raw data into the speech buffer and output it under interrupt control (ie transparent). Once the data is fed into the buffer (this is done in fractions of a second) the computer can carry on with its next task. It is very similar to the direct basic mode (see section 2.6.1) and requires that the text is converted into data by using the allophone table, (see allophone table section 2.8.)

Example :-

```
|SPON
|FEED,33,19,4,42,20,4,17,39,23,56,12,41,55,
4,45,12,16,12,17,12,21,0
```

When you enter the above the computer will say DK'TRONICS LTD:-

See the dictionary on page 49 for other examples or construct your own. The maximum number of parameters that the |FEED command can accept is 30. The mode of operation is much easier than the direct basic mode but still requires that you convert the text into allophones. It has been included in the software mainly for sound effects and can be looked upon as the fourth sound channel.

Example.

```
|FEED,62,62,62,62,0
```

Will Produce a pulsating sound

```
|FEED,41,41,41,0
```

Will produce a knocking sound

You can try any numbers from 5-63 for different effects.

2.6.3 TEXT TO SPEECH CONVERTER

The machine-code software supplied is mainly devoted to this mode of operation. Of the 4K of machine code 3.5K is used as tables which contain the rules and exceptions of the English Language.

This is therefore the most important mode and allows speech to be entered in normal English without any converting of data by the user. The software must first be loaded.

The text to speech uses two new commands :-

|SPON This command turns on the speech interrupts.

|SPOF This command turns off the speech interrupts.

```
PRINT "'Amstrad'"
```

The above example is the Syntax for using text to speech.

The ' character after the first speech mark is the control character which tells the computer that what follows is not to be printed on the screen but to be sent to the speech routines.

The second ' character is the end mark.

All text to be sent to the speech chip in this mode must be enclosed by these characters.

The ' character is the shifted \ key which is to the right of the ? key

The text to speech converter is best explained by the following examples.

Enter this program into the computer:-

```
10 |SPON:REM Turn on speech interrupts.
20 PRINT "'THIS IS TALKING'"
30 PRINT "'AND SO IS THIS'"
```

Line 10 is only necessary at the start of the program.

NOTE: Graphics characters within the text to be spoken will be ignored or produce unusual representations.

The text to speech can handle 98% of all English words. There are a few which cause the text to speech slight problems. This is mainly because while there are rules for constructing words ie i before e except after c, there are more exceptions to the rules of the English Language. These problems can all be overcome by slighty misspelling of the word. Type in the following.

```
|SPON

PRINT "'SILICON'"
```

This word sounds wrong. However, type in

```
PRINT "'SILICKON'"
```

The word now sounds correct although it has been misspelt, with some words it may be necessary to experiment with spelling although this should be rare. Speech can also be sent to the speech chip using string variables or even string expressions.

EXAMPLE:-

```
10 LET A$="'SPEECH'"
20 |SPON
30 PRINT A$
```

Speech can be used from the input statement, eg:-

```
10 |SPON
20 INPUT "'WHAT IS YOUR NAME'";A$
```

Speech can be used using string expressions, eg:-

```
10 |SPON
20 FOR I=65 TO 90
30 PRINT "'";CHR$(I);"'";
40 NEXT I
```

2.6.4 TEXT TO SPEECH BUFFERS

The text to speech convertor uses two ram buffers, the text buffer to hold the words and the speech buffer to hold the data to be outputted by interrupts to the speech chip.

The text buffer can hold 100 characters while the speech buffer is bigger and can hold 250 allophones. It is possible to fill the text buffer and if this is allowed to occur then any characters sent to the buffer while it is full will be lost.

The speech buffer will hold about 45 seconds worth of speech and will continue to talk after the program has stopped or the break key has been used. There are two ways of stopping this, after all 45 seconds of unwanted speech could be a bit nauseating.

If you type in "|FLUS" <ENTER>

This will empty the buffers and stop all talking.

OR

"|SPOF" <ENTER>

this will stop the talking but leave the buffers with data still in them and the last allophone still sounding.

This could be useful for handling Breaks from BASIC.

EXAMPLE

```
10 |SPON
20 ON BREAK GOSUB 60
30 PRINT "'Once upon a time there was a beautiful princess'";
40 IF PEEK(39014)<200 THEN 40:REM    *
50 GOTO 30
60 |SPOF:REM Halt speaking of allophones
70 OUT &FBFE,0:REM Stop last sound
80 I$=INKEY$
90 IF I$=" " THEN STOP :REM Abort when the SPACEBAR is pressed.
100 IF ,I$="" THEN 80
110 |SPON:REM Restart speaking of allophones.
120 RETURN
```

*This value is load address + 14 which you first typed in during loading. It contains the amount of space remaining in the buffer. See section 2.7 on machine-code.)

2.6.5 PRINT MODE COMMANDS

|OUTM,1

This command allows access to text to speech using "'xxx'" only. Listings and Syntax errors etc will not be spoken. This is the default setting on first loading and using the software. This command is only of any real use to cancel |OUTM,2 or |OUTM,3.

|OUTM,2

This command allows access to text to speech conversion from all print outputs. Anything that outputs to the screen ie listings, Syntax errors etc will, be spoken. The outputs will not however, be printed. You can use the command to say your listings instead of printing them to the screen. This form of listing is fairly slow as the routine will fill the buffers and then wait until there is some more room in the buffer and fill it again etc. To stop this routine press the break key and type "|FLUS" then <ENTER> if you're fed up with the speech.

EXAMPLE

To say a listing

```
|OUTM,2 (ENTER)
LIST (ENTER)
```

|OUTM,3

This command is similar to |OUTM,2 but the output will be printed on the screen as well as being spoken. To stop this routine press the break key and type into the computer "|FLUS" then <ENTER> to stop the speech.

EXAMPLE

To say and print a listing

```
|OUTM,3 <ENTER>
LIST <ENTER>
```

|OUTM,2 and |OUTM,3 will wait for space in the buffer.
|OUTM,1 will not wait and funny effects can occur if some data is lost after the buffer fills up.

2.6.6 THE |SPED COMMAND

The |SPED,n controls the speed at which the speech chip will talk. This is useful as slightly slower sounds on complicated words are easier to understand. Type in the following.

```
10 |SPON :REM Turn on speech interrupts.
20 |SPED,0
30 PRINT "'THIS IS FAST'"
40 FOR X=1 TO 5000:NEXT X
50 |SPED,15
60 PRINT "'THIS IS SLOW'"
```

The number that you use after the |SPED command has to be in the range from 0 to 15.

The speed can be changed at any point in the program so you can switch from fast to slow as you wish. The |SPED,n command also works with the |FEED,n command.

EXAMPLE:-

```
10 |SPON :REM Turn on speech interrupts.
20 |SPED,1
30 |FEED,26,16,55,17,39,26,21,0
40 FOR X=1 TO 5000:NEXT X
50 |SPED,10
60 |FEED,42,23,16,9,49,31,17,52,0
```

2.7 MACHINE CODE CONTROL

The speech can be used from machine code in two different ways. The safest way is to send data straight to the speech chip at I/0 memory address &FBFE as the speech chip never moves in the I/0 memory map. To use the text to speech routines requires more care as the speech software can be loaded in at different addresses.

2.7.1 TEXT TO SPEECH MACHINE CODE CALLS

```
ORIGIN          INITIALISE ROUTINE
ORIGIN +  2     OUTPUTS ALLOPHONE IN ACCUMULATOR TO BUFFER
ORIGIN +  4     OUTPUT STRING POINTED TO BY HL ENDED BY ZERO BYTE.
ORIGIN +  6 = |SPON
ORIGIN +  8 = |SPOF
ORIGIN + 10 = |FLUS
ORIGIN + 12 = |SPED SPEED IN ACCUMULATOR
ORIGIN + 14     NUMBER OF FREE POSITIONS IN BUFFER
ORIGIN        = LOAD ADDRESS FOR SOFTWARE
```

If you loaded the software at 39000 then the |SPON routine is at 39006.

EXAMPLE OF TEXT TO SPEECH IN MACHINE-CODE

```
        CALL ORIGIN+6
        LD HL,STRING
        CALL ORIGIN+4
        RET
STRING: DEFB."DAVID",0
```

The last character in the string must be 0

If the length of the string is rather long you can check that there is then enough space in the buffer by LD A, (ORIGIN+14) the amount of free space is in the accumulator.

EXAMPLE OF CHANGING THE SPEED

```
LD A,10
CALL ORIGIN+12
RET
```

NOTE IF YOU DISABLE INTERRUPTS (DI) THEN THE SPEECH CHIP WILL STOP TALKING UNTIL THEY ARE ENABLED AGAIN (EI) AND ANY ALLOPHONE PRESENTLY BEING SPOKEN MAY CONTINUE SOUNDING UNTIL THE (EI) IS ISSUED.
SO YOU ARE ADVISED TO OUTPUT A ZERO DIRECTLY TO THE SPEECH CHIP.

BASIC EXAMPLE

```
OUT &FBFE,0.
```

MACHINE CODE EXAMPLE

```
LD BC,0FBFE (HEX)
LD A,0
OUT (C),A
```

2.8 ALLOPHONE TABLE

Pauses

0	PA1	(10mS)	use before voiced stops and afficates
1	PA2	(30mS)	use before voiced stops and afficates
2	PA3	(50mS)	before voiceless stops and voiced fricatives also between words
3	PA4	(100mS)	between clauses and sentences
4	PA5	(200mS)	between clauses and sentences

Short Vowels - These can be repeated

7	EH	E	bend
12	IH	I	fitting
15	AX	U	succeed
23	AO	AU	aught
24	AA	O	cot
26	AE	A	fat
30	UH	OO	cook

Long Vowels

5	OY	OY	toy
6	AY	Y	sky
19	IY	E	see
20	EY	EA	great
22	UW1	O	to
31	UW2	OO	food
32	AW	OU	out
53	OW	OW	snow
62	EL	L	angle

R-Coloured Vowels

(monosyllables)	47	XR	AI	hair
	51	ER	ER	computer
	52	ER2	IR	bird
(non-monosyllables)	58	OR	OR	store
	59	AR	AR	farm
	60	YR	R	clear

Affricates

10	JH	J	jury
50	CH	CH	church

Resonants

14	RR1	R	read
39	RR2	R	brain
49	YY1	U	computer
25	YY2	Y	yes
45	LL	L	luck
46	WW	W	wool

Voiced Fricatives

18	DH1	TH	they
54	DH2	TH	bathe
35	VV	V	even
43	ZZ	Z	zoo
38	ZH	GE	beige

Voiced Fricatives

29	TH	TH	thin
40	FF	F	fire
55	SS	S	sat (29,40,55, double for initial position)
27	HH1	H	he
57	HH2	H	hoe
37	SH	SH	shirt
48	WH	WH	whig

Voiced Stops

28	BB1	B	rib
63	BB2	B	big
21	DD1	D	could
33	DD2	D	do
36	GG1	GU	guest
61	GG2	G	go
34	GG3	IG	wig

Voiceless Stops

17	TT1	T	its
13	TT2	T	to
42	KK1	C	computer
41	KK2	K	sky
9	PP	P	pub

Nasal

16	MM	M	milk
11	NN1	N	earn
56	NN2	N	no
44	NG	NG	bans

2.9 DICTIONARY

alarm	15,45,59,16
bathe	63,20,54
bathing	63,20,54,12,44
bread	28,39,7,7,1,21
calendar	42,26,26,49,7,11,2,33,51
clown	42,45,32,11
checked	50,7,7,3,41,2,13
checkers	50,7,7,3,42,51,43
checks	50,7,7,3,42,55
collide	8,15,45,6,21
cookie	8,30,42,19
correct	42,52,7,7,2,41,2,17
correcting	42,52,7,7,2,41,2,13,12,44
crown	42,39,32,11
daughter	33,23,13,51
divided	33,12,39,6,2,33,12,2,21
engage	7,7,1,11,36,20,2,10
engages	7,7,1,11,36,20,2,10,12,43
enrage	7,11,14,20,2,10
enrages	7,11,14,20,2,10,12,43
escape	7,55,55,3,42,7,3,9
escapes	7,55,55,3,42,2,3,9,55
equal	19,2,3,8,48,15,62
error	7,47,58
fir	40,52
freezer	40,40,14,19,43,51
freezing	40,40,14,19,43,12,44
gauge	36,20,2,10
gauges	36,20,2,10,12,43
hello	27,7,45,15,53
hour	22,51
intrigue	12,11,3,13,39,19,1,34
intrigues	12,11,3,13,39,19,1,34,43
investigate	12,12,11,35,7,7,55,2,3,13,12,1,36,20,2,13
investigater	12,12,11,35,7,7,55,2,3,13,12,1,36,20,2,13,51
investigates	12,12,11,35,7,7,7,55,2,3,13,12,1,36,20,2,17,55
key	42,19
legislating	45,7,7,2,10,10,55,55,45,20,2,3,13,12,44
legislated	45,7,7,2,10,10,55,55,45,20,2,3,13,12,21
letter	45,7,7,3,13,51
little	45,12,12,3,13,52
memories	16,7,7,52,19,43
month	16,11,12
nipped	11,12,12,2,3,9,3,17
nips	11,12,12,2,3,9,55
pin	9,12,12,11
pinning	9,12,12,11,44
pledge	9,45,7,7,3,10
pledges	9,45,7,7,3,10,12,43
plus	9,45,15,15,55,55
rays	14,7,20,43
red	14,7,7,1,21
robots	14,53,2,63,24,3,17,55
second	55,55,7,3,42,12,11,2,21
sincere	55,55,12,12,11,55,55,60
sincerity	55,55,12,12,11,55,55,7,7,14,12,2,3,13,19
speak	55,55,3,19,3,41
spelled	55,55,3,9,7,7,62,3,21
spellers	55,55,3,9,7,7,62,52,43
spells	55,55,3,9,7,7,62,43
started	55,55,3,12,59,3,13,12,1,21
starting	55,55,3,13,59,3,13,12,44
stop	55,55,3,17,24,24,3,9
stopper	55,55,3,17,24,24,3,9,51
stops	55,55,3,17,24,24,3,9,55
subject	55,55,15,2,28,2,10,7,7,3,41,3,13
sweated	55,55,46,7,7,3,13,12,3,21
sweaters	55,55,46,7,7,3,13,61
sweats	55,55,46,7,7,3,13,55
switched	55,5,48,12,12,3,50,3,13
switching	55,55,48,12,12,3,50,12,44
systems	55,55,12,12,55,3,13,7,16,43
talked	13,23,23,3,41,3,13
talkers	13,23,23,3,42,51,43
talks	13,23,23,41,55
threaded	29,14,7,7,2,21,12,2,21
threaders	29,14,7,7,2,33,51,43
threads	29,14,7,7,2,33,43
time	13,24,6,16
uncle	15,44,3,8,62
whaler	46,20,45,51
whales	46,20,62,43
year	25,60

64K and 256K MEMORY EXPANSIONS

These units are available for the CPC 464, 664 and 6128 computers.

By using the 64K upgrade the 464 and 664 computers will have the same amount **and configuration** of RAM as the CPC 6128. The 256K gives an extra 192K on top of this!. The expansion will allow the use of CP/M PLUS (R) as supplied with the CPC 6128 with its massive 61K TPA opening up an even larger software base for Amstrad users. There is also a utility for increasing the TPA on CP/M 2.2 to 61K.

The Ram can be accessed by means of bank switching using a single I/O port. Memory is actually switched in and out of the 64K Z80 address space in 16K sub-blocks, as are the ROMS. The port determines which particular combination of the original four 16K sub-blocks and any new sub-blocks from the expansion RAM will occupy the 64K address space at any time. Control of the I/O port can be from either BASIC or machine-code.

To use the additional 64K/256K of RAM, the expansion is supplied with bank switching software (although it can be switched without this software).

The software adds some extra BASIC commands, RSXs, which make it possible to use the second 64K (or 3rd, 4th and 5th in the case of the 256K expansion) for storage for screens, windows, graphics and BASIC arrays. This ability means that you can write much larger BASIC programs, as most of the memory on the unexpanded CPC464/664 is normally used for arrays, variables and graphics.

The additional BASIC commands are;-

¦BANK,n	Map a bank of 16K directly into memory space.
¦SWAP	Alternate between the low an high screens.
¦LOW	Change to the low screen.
¦HIGH	Change to the high screen. (Default screen).
¦SAVES,n	Store a screen to a 16K bank.
¦LOADS,n	Retrieve a screen from a 16K bank.
¦SAVEW,w,n	Store a windows's contents into expansion RAM.
¦LOAD,w,n	Load a window with data from the expansion RAM.
¦SAVED,n,s,l	Transfer original RAM to expansion RAM.
¦LOADD,n,s,l	Load original RAM from expansion RAM.
¦PEEK,n,s,v	Read the value of a byte in expansion RAM.
¦POKE,n,s,v	Change a byte in the expansion RAM.

These commands make such features as pull down menus, full screen animation and large spread-sheet type programs or Data-Bases very easily programmed from BASIC as never before possible on the unexpanded CPC464, 664 computers.

CP/M 2.2 and CP/M PLUS are registered trademarks of Digital Research Inc.

BANK SWITCHED RAM MANUAL FOR 64K & 256K EXPANSIONS

WARNING

Ensure that the power to your Amstrad computer is switched OFF before you fit the interface to the expansion socket. Failure to comply with these instructions may cause permanent damage to the RAM pack or the computer.

3.1 Installation

Power down your Amstrad computer. Plug the RAM pack into the socket on the back of the computer. On the CPC 464 this socket is labelled 'Floppy Disc', on the CPC 664 and CPC 6128 the socket is labelled 'Expansion'. Other expansions such as the Amstrad Disc interface for the CPC 464, DK'tronics Lightpen and Speech Synthesiser, or ROM expansions can be fitted into the expansion socket on the back of the RAM pack. Now switch on the computer.

The computer should power up as normal. If it fails to do so, check that all the connections are correctly made. Note that all DK'tronics products have a key location on the connector to ensure that there can be no alignment problems. OTHER interfaces may not have this keyway (the Amstrad disc interface is the most familar example). Hence any connection problems will usually lie between the RAM pack and these expansions. If this is the case, try reconnecting the interfaces BEFORE inserting the RAM pack into the computer. This will give you a better view when lining up the pins.

If the computer fails to power up, or crashes on power up (Miscellaneous patterns all over the screen!), the monitor may cut out the power to the computer. On the colour monitor, just switch the MONITOR off and then attempt to reconnect as above. The monochrome monitor may have to remain swtiched off for several seconds before power will be reinstated to the computer.

It is very unlikely that the computer will fail to power up with the RAM pack alone. If this is the case, then the fault will probably lie with the RAM pack. * Return the RAM pack to DK'tronics if this is the case.

* IT IS ESSENTIAL THAT YOU COMPLETE YOUR WARRANTY REGISTRATION CARD AND RETURN IT TO US IMMEDIATELY UPON PURCHASING THIS PRODUCT FROM YOUR DEALER (UK ONLY).

3.2 USING YOUR EXTRA RAM

There are two ways to use the extra RAM. There is a cassette/disc supplied with the RAM pack containing extensions to BASIC. Here the extra RAM can be used simply from BASIC programs. Alternatively, the RAM is accessible both from BASIC and machine code using the OUT command. The experienced programmer will be able to use the RAM for whatever he pleases and write custom software for that purpose. Commercial programs will no doubt use this approach.

The second method is described in detail in section 3.10. The first way is explained in the following chapters:-

With the computer set up as above, load the RSX software from the cassette/disc supplied:-

If the cassette is being used on disc systems type '|TAPE' and press <ENTER>

b) Type 'RUN "BANK"' and press <ENTER>.
c) The loading sequence is described in detail in your Amstrad user manual.
d) When the program has finished loading, you will be asked to enter a loading address. Just press <ENTER> for now. (See section 3.11.)
e) The computer will test the RAM and then print out how much RAM you have got, then the computer memory will be clear ready for your own programs.

The cassette tape contains the same programs on both sides so that if one side fails to load, the other is there as a backup.

Subsequent programs on the cassette are extracts from this manual which may be loaded from tape if you do not want to type them out.

3.3 RAM TEST

When the RSX code is first loaded, it does an extensive RAM test. Should the RAM not function correctly the program will inform you that an error has been found. Along with this, it will print out diagnostic information to help in the repair of the RAM pack.

In the unlikely event that an error is found, please note the information that is given and return the RAM pack for replacement or repair. (See warranty registration note page 3- 2.)

3.4 EXTENDED BASIC COMMANDS

There are a total of twelve extra commands provided by the RSXs on tape. Some may have parameters, some will not. Sometimes the command may have different formats and numbers of parameters. We have tried to discuss each command in its simplest form and later sections will describe added parameters which make the command more flexible and economic on memory.

You may have noticed that during the RAM test, the computer printed out the number of the 'bank' it was testing. Each bank is 16K of memory. For the 64K expansion there are 4 bankswhile the 256K RAM pack has 16 banks. To access a particular part of the expansions's memory there has to be a bank number and possibly a bank address.

For example, type:- '|SAVES,1' and press <ENTER>

The computer will respond with READY. What you have done is to store what was on the screen into bank 1.

Now clear the screen using CLS. To get the screen's contents back, type:- '|LOADS,1' and press <ENTER>

You can save as many screens as you have memory for. That means four screens on the 64K RAM and sixteen screens for the 256K RAM.

Screen displays could be created from another program or drawn using a lightpen. Store these on tape or disc then load them back into RAM for use throughout the program. Screen displays which take a long time to create within a program, for example mazes, can be created once, then stored for instant use whenever necessary.

The command can be summarised:-

|SAVES, [bank] save data to bank
|LOADS, [bank] load data from bank

3.5 WINDOWS AND PULLDOWN MENUS

One of the features that makes the Amstrad's windows less flexible than those on larger business machines, is the fact that the contents of a window which overlaps another are lost when the other window is used.

There are two new commands which allow the contents of windows to be saved and reloaded from RAM. This will allow the use of true pulldown menus, that can cover text, but not remove it.

EXAMPLE:-

```
NEW
 10 MODE 1
 20 FOR i=0,05 TO 1 STEP 0.05 : REM Draw grid on screen
 30      MOVE 640*i,0 : DRAW 640*i,400
 40      MOVE 0,400*i : DRAW 640,400*i
 50 NEXT i
 60 WHILE INKEY$="" : WEND : REM Wait for a key press


 70 WINDOW#1,INT(RND(1)*19+1),INT(RND(0)*9+INT(RND(1)*5+17)),
     INT(RND(1)*14+1),INT(RND(0)*14+INT(RND(1)*10+5))
 80 PEN#1,2 : PAPER#1,3
 90 |SAVEW,1,1 : REM Save contents of window into RAM
100 CLS#1 : REM Clear window
```

```
110 WHILE INKEY$="" : REM Wait for 2nd key press
120 PRINT#1, "This is a window"
130 WEND
140 |LOADW,1,1 : REM restore window's contents
150 GOTO 60
```

The above program uses two new commands: |LOADW and |SAVEW. As you are probably aware, there are eight windows (0-7) which can be defined. The first parameter is the reference to a window. The second is the bank number.

|SAVEW, [window number], [bank] save window to bank

|LOADW, [window number], [bank] load window from bank

See the chapters in the user manual about window for more details.

3.5.1 MORE WINDOWING

A window of any size, even the whole screen, will fit into a single bank of expansion RAM. This is fine if your window is nearly a full screen or will vary in size like the above example. On the other hand if your window was defined as 10 x 10 in Mode 1, then the amount of memory needed to store this window would be less than 16K. In fact only 1600 bytes are needed (see below). Thus to use a whole bank would mean wasting over 14K of memory.

To deal with this problem, the RSX window command can take an extra parameter to define where you want the window's contents to reside in the RAM bank. In the 10 x 10 window you could place the data anywhere between 0 and 14783. The command can be written:-

|SAVEW, [window number], [bank], [bank address]

|LOADW, [window number], [bank], [bank address]

The bank address ia an address between 0 and 16383. The amount of data in bytes used to store a window needs to be taken away from the top value and this leaves the range between which the data can be stored. If you put the data at the bottom of the RAM bank, at address 0, then the memory from 1600 to 16383 is free for other windows or data arrays.

HOW TO CALCULATE A WINDOW'S SIZE

In order to have more than one window per bank, you need to know how much memory the window will take up. If the window will vary in size between two limits, use the higher of the two. Depending on which mode you are using, the figures are calculated as below.

In each case: X1 is the left most x coordinate
              X2 is the right most x coordinate
              Y1 is the top y coordinate
              Y2 is the bottom y coordinate

```
MODE 0        SIZE=(X2-X1+1) * 4 * (Y2-Y1+1) * 8

MODE 1        SIZE=(X2-X1+1) * 2 * (Y2-Y1+1) * 8

MODE 2        SIZE=(X2-X1+1) * (Y2-Y1+1) * 8
```

The computer will give an error if the window is too large to fit in the space you have allotted for it. Also if the size is mis-calculated the windows may overlap in the bank and cause strange effects.

EXAMPLE 2:-

```
 10 PEN 1 : PAPER 0 : MODE 1
 20 size=14 * 2 * 10 * 8
 30 LOCATE 1,13 : PRINT " 'n' for new window 'd' to remove window"
 40 WINDOW 1,14,1,10 : PAPER 3 : CLS
 50 bankaddress=0 : level=0
 60 PRINT#level, "Window";level
 70 keypress$=LOWER$(INKEY$)
 80 IF keypress$="n" THEN GOSUB 110
 90 IF keypress$="d" THEN GOSUB 190
100 GOTO 60
110 If level=7 THEN RETURN
120 level=level+1
130 WINDOW#level,1+level*3,14+level*3,1+level*2,10+level*2
140 |SAVEW,level,1,bankaddress
150 bankaddress=bankaddress+size
160 PEN#level,0 : PAPER#level,(level AND 1)+1
170 CLS#level
180 RETURN
190 IF level=0 THEN RETURN
200 bankaddress=bankaddress - size
210 |LOADW,level,1,bankaddress
220 level=level-1
230 RETURN
```

The above program only uses one bank of RAM but all 8 windows are defined. The variable 'level' is used to stand for the level of windows and the variable 'bankaddress' points to the next free place in the bank RAM.

3.6 ARRAYS, VARIABLES AND STRINGS

There are two general purpose data moving commands to allow data from the program to be moved to and from the RAM pack.

These two commands are:-

```
|SAVE, [bank], [Start location], [length], [bank address]
|LOADD, [bank], [Start location], [length], [bank address]
```

The first parameter references which bank you want to use. The start location is a memory address where there is some data. The amount of data is given as the length. Optionally a bank address can be given to allow more than one type of data to be stored in the RAM.

It is possible to save all kinds of data using these commands, but we will firstly discuss how to save simple numerical arrays these being the easiest to understand.

Say for example that your program deals with stock control of up to 60 items. You may have a string array containing the names and a numerical array containing the number of each item you have in stock.

This would use about 1K for the names and 300 bytes for the stock figures. However what if you update the stock value every week and you want to keep the last year of stock on record! Or even the last five years. Now the figures would take up about 15K or even 75K.

These could be comfortably stored on disc, or even tape for a year's stock, and the data read every time a calculation was needed, but you will probably agree that a long time would be spent waiting for reading the data each time a distribution is calculated for each item.

Obviously, it would be easier to load all the records into RAM, then access the data immediately:-

Instead of defining an array of dimensions 'stock(60,52)' taking over 15K of valuable RAM which could be used for programs, define and array 'stock(60)'. Read all the data from disc a week at a time, and store each week of data into bank RAM. To do this you need to know two things. One, where does the array lie in memory? and two, how many bytes is it necessary to save?

1) Where is an array stored?
The address of any variable can be quickly found using the '@' before a variable. For example, dimension the above array:-

```
DIM stock (60)
```

Now type; PRINT @stock(0)

The computer will reply by giving the memory address where the first element of the array is stored. Try:-

```
PRINT @stock(1)
```

The number returned will be five higher in value. This is the address of the second variable.

The '@' prefix will work in front of any variable. The first item of an array is obviously '@stock(0)'. If you are using multi-dimensional arrays, the first item is '@stock(0,0)' or '@stock(0,0,0,0)' depending on the number of dimensions.

2) How long is an array?

First of all, different types of array take different numbers of bytes per element. For real numbers, there are 5 bytes per element. Integer arrays take 2 bytes per element. String arrays are of variable length. And will be dealt with later.

Next, the number of dimensions and elements needs to be taken into account. Remember that elements start from 0. This means that an array of 'stock(60)' has 61 elements. Whether or not you prefer to use the 0 element is up to you, but if you forget it, there could be some unexplainable bugs appearing in your program. Once you know the real number of elements in every dimension, simply multiply together all the dimensions to find out the total number of elements in all dimensions.

```
For example: 'stock(60)' has a total of 61 elements.
             'stock(60,52)' has 61 * 53 elements = 3233 elements.
             'stock%(10,5,12)' has 11 * 6 * 13 elements = 858 in all.
```

To find the total memory, multiply the total number of elements by the amount of memory needed by each element.

```
For example: 'stock(60)' takes 61 * 5 = 305 bytes.
             'stock(60,52)' takes 3233 * 5 = 16165 bytes.
             'stock%(10,5,12)' takes 858 * 2 = 1716 bytes in all.
```

The array we are using is 304 bytes long, and starts at @stock(0). In a single bank of Ram we can store 305 bytes about 53 times. The bank address starts at 0 and goes up in steps of 305 bytes:-

0 305 610 915 1220 1525 etc.

We shall store week 1 at bank address 305, week 2 at address 610 and so on for all 52 weeks.

Data for test purposes could be written onto disc or tape by the program below. Once the test file is written, keep it for use while you are developing your program.

```
10 OPENOUT "stock.dat"
20 FOR week=1 TO 52
30    FOR item=1 TO 60
40       Print#90, INT(RND(1)*3000+100 )
40    NEXT item
60 NEXT week
70 CLOSEOUT
80 END
```

Now type 'NEW' and enter the following program;-

```
  10 DIM stock(60)
  20 INPUT "read file (y/n)";ans$
  30 IF LOWER$(ans$)="y" or LOWER$(ans$)="yes" THEN GOSUB 1000
  40 REM rest of program  ....
1000 REM subroutine to read data from disc.
1010 OPENIN "stock.dat"
1020 FOR week=1 TO 52
1030      FOR item=1 TO 60
1040         INPUT#9, stock(item)
1050      NEXT item
1060      |SAVED,4,@stock(0),61*5,week*305
1070 NEXT week
1080 CLOSEIN
1090 RETURN
```

The above program could be used to read the file from disc or tape. Once the file is in bank RAM, the contents will stay there for use until the computer is switched off, or some other data is put in that bank. This means that data need only be read once from disc, then the program can be rerun without losing the data. This could also be useful too if you wish to write a number of programs to use the same data.

Once the data is in memory, you can access each week's data simply by reloading the stock array. Add the section below to draw a bar graph for a given section.

```
100 MODE 2
110 LOCATE 1,1
120 INPUT "Which item to analyse",itemno
130 IF itemno<1 OR itemno>60 THEN 120
140 CLS : LOCATE 30,1
150 PRINT "Bar Chart For Item"; itemno
160 LOCATE 10,25
170 PRINT"Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec";
    REM 3 spaces between each.
180 FOR loop=0 TO 4
190    LOCATE 1,24-loop*5
200    PRINT STR$(loop);"000"
210 NEXT loop
220 MOVE 60,328 : DRAW 60,0 : DRAW 61,0 : DRAW 61,368 :
    MOVE 640,24 : DRAW 48,24
230 FOR loop=1 TO 4
240    MOVE 48,loop*80+24 : DRAW 60,loop*80+24
250 NEXT loop
260 FOR week=1 TO 52
270    If week/2=week/2 THEN n=1 ELSE n=2
280 |LOADD,4,@stock(0),61*5,week*305
290 ycoord=( stock(itemno)/4000*320 ) AND 4092
300 FOR xcoord=1 TO 11 STEP n
310    MOVE 49+xcoord+week*11,ycoord+26 :
       DRAW 49+ xcoord+week*11,26
320 NEXT xcoord
330 NEXT week:GOTO 110
```

3.6.1 MORE ARRAYS, VARIABLES AND STRINGS

If you have a program that uses all the memory of the computer due to needing a large array, you can use the bank RAM for storing data without even dimensioning an array.

For example if you have a two dimensional array 'sales%(365,30)' to store the amounts of certain types of stock you sell for each day in one year. Even though you are using intergers, the array uses over 22K of memory.

Instead of having the whole array in BASIC memory, each element can be accessed by using a subroutine to read out a value, and one to store a value.

```
10000 REM load 'store%' from bank memory using 'year' & 'type'.
10010 p=( year*31+type)*2
10020 bank=1 : IF p >=16000 THEN p=p-16000 : bank=2
10030 |LOADD, bank, @store%, 2, p
10040 RETURN
11000 rem copy 'store%' to bank using 'year' & type'
11010 p=( year*31+type )*2
11020 bank=1 : IF p >=16000 THEN p=p-16000 : bank=2
11030 |SAVED, bank, @store%,2, p
11040 RETURN
```

Two banks are used, 1 and 2, and the variables 'year' and 'type' are used to reference which element is required. On line 10030 and 11030, there are just 2 bytes moved to and from the bank RAM because we are using integers. The '*2' in lines 10010 and 11010 reflect the fact that an integer is stored in two bytes. If real variables were used, 5 bytes would need to be used instead. Lines 10020 and 11020 decide whether the element is in the first bank or the second.

If the array is to be filled with data from tape or disc, there is no need to initially clear the values to nil. If you want all elements preset to zero then the easiest way is to save a blank screen into each bank at the start of the program:-

```
10 MODE 1 : PAPER 0 : CLS
20 |SAVES,1
30 |SAVES,2
```

3.6.2 STRING STORAGE

The major obstacle in storing strings is that they can vary in length and can be stored anywhere in memory, including in a BASIC program. One method of storing string arrays is outlined below. However you may find an easier way to store strings than the one described below when you consider exactly what you want to do.

Suppose that you wanted to store 500 names, up to 20 characters long each. A bank is seperated into units of memory 21 bytes each so that strings can be randomly accessed. In each 21 byte segment there is one string, and one byte to say how many letters there are in that stirng. That means that we will use a total of just over 10K. If we use the variable 'name' to specify the string we want then we can enter two subroutines one to put a string from bank 1 into 'name$' and one to store the contents of 'name$' into RAM bank number 1:-

```
20000 REM assign 'name$' to string number 'name'
20010 b$="                     " : REM 21 spaces
20020 |LOADD, 1, PEEK( @b$+1 )+PEEK( @b$+2 )*256, 21, name*21
20030 name$=MID$(b$, 2, ASC(b$) ) : RETURN

21000 REM Store 'name$' in bank as element 'name'
21010 b$='                     " : REM 21 spsaces
21020 MID$( b$,1,21 )=CHR$( LEN(name$) )+name$
21030 |SAVED, 1, PEEK( @b$+1 )+PEEK( @b$+2 )*256, 21, name*21
21040 RETURN
```

A dummy string b$ is used to form the element before it is saved into RAM. The first character is set to the length of 'name$'. The latter 20 characters are where the contents of 'name$' are stored. Then 21 characters are copied into bank RAM. When the string is retrieved the characters are copied out and 'name$' is set to the right length by looking at the first character.

String storage would come into its own if all the words were of the same length because there would be no wasteage. For example a word quiz program using five, six and seven letter words. A bank of RAM could be used for each length of word. A loader program would set up the data into the RAM, then another could be CHAINed and use up to 36K of RAM for program.

A number array could also be stored in bank RAM to index the first letters and so aid the speed of access to a particular word.

3.7 ANIMATION AND PICTURE SHOWS

We have seen how screens and windows can be stored and retrieved. Animation is the act of putting pictures on the screen quickly enough so that the eyes see something move. With 64K or 256K of memory whole screens can be stored away, then put on the screen to produce animation.

You may have noticed in section 4 that when a screen loads onto the screen, you can see each line appear. To illustrate, type in the following program:-

```
 10 MODE 1
 20 BORDER 0
 30 FOR col=0 TO 3
 40    INK col,0
 50 NEXT col
 60 FOR col=0 TO 3
 70    PAPER col : CLS
 80 |SAVES,col+1
 90 NEXT col
100 INK 0,1 : INK 1,6 : INK 2,21 : INK 3,13
110 PEN 1 : PAPER 0
120 WHILE INKEY$=""
130    FOR screen=1 TO 4
140       |LOADS, screen
150    NEXT screen
160 WEND
170 END
```

The program saves four coloured screens into bank RAM, then loads them up in sequence. Unfortunately, the effect is a striped pattern.

In order to create animation which is easy on the eye the computer needs to create the screen display, then instantly display it.

Three new instructions that allow this to be done are:-

|LOW, |HIGH and |SWAP

Before the commands can be understood it is necessary to know how the Amstrad's screen can be used. The normal screen is located at 49152. However the Amstrad is capable of viewing a screen anywhere in memory in 16K blocks. The first block at 0 and the third block at 32768 are difficult to use for screen as the computer uses these as part of the BASIC interpreter. The block of memory at 16384 is free for use as long as BASICs HIMEM is lowered to below 16384. Using this, we have called the original screen the high screen and the new screen at 16384 is called the low screen. To go from one to the other just use:-

|LOW to set the low screen in action
|HIGH to reset the high screen
|SWAP to swap from low to high and vice-versa

Whenever the swap is made, the computer is told, and all further text and graphics appear on the selected screen.

To use this facility of swapping from one screen to another instantly, the screen and window commands can have an added parameter which tells the computer to load or save the data to and from the alternate screen.

The new forms can be written:-

```
|SAVES, [ bank ], [ swap ]
|LOADS, [ bank ], [ swap ]
|SAVEW, [ window number ], [ bank ], [ bank address ], [ swap ]
|LOADW, [ window number ], [ bank ], [ bank address ], [ swap ]
```

If the swap value is zero, by default, then the command will act on the screen that is presently being displayed. Alternatively, if the value is one the computer will load and save data from the screen which is not being displayed. When the work is done, the computer can swap screens and the effect is that the screen appears to change instantly.

In the above program type these lines:-

```
  5 MEMORY 16383 : |HIGH
135 IF screen\2=screen/2 THEN t=TIME : WHILE TIME <t+20 : WEND
140 |LOADS, screen, 1 : |SWAP
```

Now that the computer can build up the screen while another is being displayed there is no pattern. The coloured screen appears to change instantly.

Due to the fact that the bank memory moves into the address space at 16K it takes longer for the transfer of screens to be made to the low screen than to the high screen. Hence line 135 delays the computer as it is to load the high screen. This means the time each screen is on the screen remains the same. Try removing line 135 to see the difference.

If a longer delay were to be put between line 140 and 150 you would get a picture show effect. Alternatively, you could select screens when a key is pressed.

On a small scale, a window could be defined and graphics could be rapidly displayed without resorting to swapping screens.

Note, that of less use is the fact that the contents of screens and windows can be saved from a screen which is not on display simply by adding a one for the swap parameter. For example if you want to load a series of screens from tape or disc, load them into the low (16K) screen. Messages generated by the tape system need not be switched off as the screen's contents will not be changed in the low memory screen.

```
10 LOAD "screen1",16384 : |SAVES, 3, 1
```

The above will load a screen then save it to bank 3. The screen the user sees can have something else on it.

3.8 ADVANCED PROGRAMMING

This section introduces one new command and some other programming aspects which you may find useful.

The new command is:-

```
     |ASKRAM, [ enquiry ], [ variable ]
```

The command allows certain constants to be found by the program you are writing. For example it can return the number of banks available to the program as this will change depending on whether you are using the 64K RAM pack or the 256K RAM pack. The 'enquiry' value is a number 1 to 3 which selects what you want to know. The answer is placed in an INTEGER variable defined by the second parameter.

```
     1000 a%=0 : |ASKRAM, 1, @a% .. will assign a% to the amount of RAM
     1100 a%=0 ; |ASKRAM, 2, @a% .. will assign a% to the number of
     banks

     1200 a%=0 ; |ASKRAM, 3, @a% .. will set a% to 0 or 1 depending on
     whether there is a problem with the RAM
```

The last command can be used to make sure the RAM is there and ready to use if in your programs you do not want to have to load the RSX loader first. It is possible to load just the RSX machine code on its own;-

```
      20 MODE 1 ; PRINT "Program Loading!"
      30 I=HIMEM
      40 MEMORY 9999
      50 LOAD "rsx", 10000
      60 I=I-( PEEK( 10004 )+PEEK( 10005 )*256+1 )
      70 POKE 10002, I-INT( I/256 )*256
      80 POKE 10003, INT( I/256 )
      90 PRINT CHR$(30);CHR$(21);
     100 CALL 10000
     110 PRINT CHR$(30);CHR$(6);
     120 a%=0 ; |ASKRAM, 3, @a%
     130 IF a% THEN PRINT "RAM is faulty" ; END
     140 CLEAR ; MEMORY PEEK( 10002 )+PEEK( 10003 )*256-1
     160 CHAIN "part2"
```

The program above will load the RSX machine code and put it into memory. Nothing will be printed on the screen unless the RAM proves to be faulty or not even there! The program 'part2' would be the bulk of the program. Loadng the program in two parts saves reloading the RSX code every time the program is run.

The code has to be loaded in at 10000 in memory before it is relocated for use. The 16 bit value in locations 10002 and 10003 is the place you want the code to be located at. Another 16 bit value in loacations 10004 and 10005 contains the length of the code which is moved higher in memory. Nearly 1K of the program is only need once - the relocation and the RAM test programs, and hence this part is not moved higher in RAM.

If you want to use user defined graphics then add the following lines:-

```
      10 SYMBOL AFTER 256
     150 SYMBOL AFTER 0
```

The value in line 150 will be different depending on how many user defined graphics you want.

In your program you may want to have a number of different styles of character set. After you issue a SYMBOL AFTER command, HIEMEM is set just below the user defined graphics. Hence it is possible to use the |LOADD and |SAVED commands to move graphics to and from the graphics characters.

If you have a program that defines the character set, the definitions can be saved and loaded into bank RAM so that a program may have multiple chartacter sets.

```
       10 SYMBOL AFTER 0
       20 chars=HIMEM+1
       30 REM define symbols here
     1000 SAVE "set1.grp", B, chars, 2048
```

This program will save you character set onto disc or tape.

On your final program you may wish to load a number of sets:-

```
     10 SYMBOL AFTER 0
     20 chars=HIMEM+1
     30 LOAD "set1.grp", chars : |SAVED, 1, chars, 2048, 0
     40 LOAD "set2.grp", chars : |SAVED, 1, chars, 2048, 2048
     50 LOAD "set4.grp", chars : |SAVED, 1, chars, 2048, 4096
```

The reason the variable 'chars' is set up is because the value of HIMEM alters when the disc or tape is accessed.

During the program, a subroutine could be used to select a character set:-

```
     1000 REM given the variable 'set', load the characters
     1010 |LOADD, 1, chars, 2048, ( set-1 )*2048
     1020 RETURN
```

Note that the variable 'set' is used. In the above loading sequence sets 1 to 3 will be valid. More or less could be added as it suits you.

All of this setting up can be done on the loader program, just once. When the program is subsequently run, there is no need to re-load the bank RAM.

The setting of chars can be found whenever needed by:-

```
     200 CLEAR : SYMBOL AFTER 0 : chars = HIMEM+1
```

This will remove any disc buffers that have been set up and 'chars' will indeed point to the characters.

3.9 PEEKING AND POKING

There are two commands which allow the memory in the banks to be viewed and changed byte by byte.

```
     |POKE, [ bank ], [ bank address ], [ value ]
     |PEEK, [ bank ], [ bank address ], [ variable ]
```

|POKE works in a similar way to the original POKE. You need to supply a bank number in addition to the normal address and value. The bank address is in the range 0 to 16383 or 0 to 16K.

|PEEK is a command rather than the normal function. The bank and bank address are the same as for |POKE. To find out the value, you need to supply an integer variable in a similar way to the |ASKRAM command.

For example:-

```
     10 value%=0
     20 |PEEK, 3, 12345, @value%
     30 PRINT value%
```

The above will read the byte from location 12345 in bank number 3. The @ character tells the RSX extension where the variable is in memory so that its contents can be changed to the byte required.

|PEEK and |POKE are not really commands for the beginner, in fact they have only been included for the more advanced programmer who wishes to use the bank RAM in his own way.

Another advanced command which has been included for the experienced programmer is |BANK.

```
     |BANK, [ bank number ]
```

The command is followed by one parameter. If this parameter is not present, a zero is assumed. The bank referenced is mapped into the address space at 16K to 32K. A bank number of zero will map the original RAM back in, numbers 1 to the maximum bank number will map that bank into the address space. If a bank is mapped in, the computer will use the bank memory istead of the normal RAM. However, the screen will still be taken from the original RAM if |LOW was issued. The advantage of this is that the whole memory can be used for programming instead of having to set the top of memory to 16383. The disadvantage is that if the program is halted while the low screen is being displayed, the computer will write screen data into the BASIC program - causing chaos.

Make sure you are accustomed to using |BANK, |POKE and |PEEK before you risk creating a large program using them. Save the program frequently in case you make a mistake and lose your work.

3.10 PROGRAMMING WITHOUT RSXS

With no RSX software the programmer can still access the memory from the RAM banks. To use the RAM yourself, some degree of understanding of the memory map of the Amstrad is necessary.

From both BASIC and machine code, the original block of memory from 16384 to 32767 CANNOT be used for program. Hence in BASIC, you need to set the top of memory to 16383. Machine code is free to use any memory that it can normally except the block mentioned.

The extra RAM is mapped into the addresses 16384 to 32767 in 16 banks,Once the bank is mapped in, you can do anything with the RAM you normally would. It is inadvisable to use the bank RAM for machine code because if you subsequently change the bank, the program disappears! Nevertheless, programs can be written to run in banks and indeed in the original 16K block that is banked out, but it is necessary to do the bank changing outside of this memory range. In BASIC it would be extremely difficult to use the banked RAM for extra programs, but not impossible, but we shall leave that possiblility up to you!

The way that banks are selected is defined below:-

```
     IN BASIC: Where 'bank' is the number of the bank to map in
     OUT &7F00, 196+( bank AND 3 )+( bank AND 28 )*2
     NOTE: the bank numbers in this case START AT 0
```

For 64K expansions the banks are 0 to 3. On the 256K, bank numbers are 0 to 15.

To reset the original bank:-

```
     OUT &7F00, 192
     IN MACHINE CODE: Where the bank number is in the accumulator. (A)

SELECT: PUSH BC          ; select bank A (save all registers except a
        LD C,A           ; and flags)
        AND 3            ; bank AND 3 ) +
        LD B,A
        LD A,C
        AND 28           ;(bank AND 28)*2
        ADD A,A
        OR B
        OR 196           ; + 196
        LD BC,07F00H     ;BC=&7F00
        OUT (C),A
        POP BC
        RET
```

Again the bank number in the accumulator starts at 0. To reset the original bank:-

```
RESET: PUSH BC             ;save reg.
       LD BC,07F00H        ;select control port
       LD A,192            ;select code for original bank
       OUT (C),A           ;restore bank
       POP BC              ;restore reg.
       RET
```

3.11 TECHNICAL DETAILS

3.11.1 The Load Address

The software which loads from tape is relocatable. However the areas of memory in which the program can go is limited to between 32768 and the top of memory. This is because the banked RAM appears in the block 16384 to 32767. (See previous chapter for explanation of why!) Below the 16K boundary, the RSX command table will no longer funtion. Hence, during relocation, the code is loaded at 10000 in memory and moves to a place higher in memory. Pressing <ENTER> while loading, will automatically select the highest location available. Alternatively you may wish to load the code to a lower address and reserve some space for your own programs.

3.11.2 Saving to Disc

The software on the cassette is NOT protected. Hence to save it onto disc or even onto another tape at speed write 1 is a matter of loading the data into memory, then saving it.

1) Type '|TAPE' and press <ENTER> (for disc systems)
2) LOAD "bank"
3) MEMORY 9999
4) LOAD "rsx", 10000
5) Type '|DISC' or set SPEED WRITE as desired
6) SAVE "bank"
7) SAVE "rsx", B, 10000, 4000

3.11.3 INCREASING CP/M 2.2 TPA

Run the program called "TPA" from either cassette or disc, depending if you have a 464/664 or 6128 computer. When instructed to, place a CP/M 2.2 system disc into drive A.

The program will now write a file called INCTPA.COM onto your system disc.

Boot up CP/M 2.2 then immediately after the 'A>' promt type in 'INCTPA' then press <ENTER>. Your system disc will now be altered to run with a 61K TPA.

Commercial Program Compatability

The RAM expansion is compatible with the banked RAM supplied as standard with the CPC6128. This means that a number of programs wirtten for the CPC6128 will now work on the CPC464 and CPC664 range of computers.

In fact the RSX software provided will work on the CPC6128.

The bank switching software in its supplied state will only access 256K of banked memory, that is 16 banks. If you add more memory the RSX software can be told to access a full 512K of banked memory (32 banks) by poking location 10006 with 1. See section 3.8 for explanation of how to load the RSX software on its own.

```
    55 POKE 10006, 1
```

This line will do the trick!

If a commercial program fails to work on your CPC 464/664 then try the suggestions below.

1) The software may be using the new firmware vector at &BD58. If this is the case, try running the RSX program before running your application program.

   Some programs which will function correctly after the RSX softwaretape has been loaded in are Tasman's Tasword (R) word processor, Tas- spell and Tasprint for the CPC6128. In conjunction with these, Campbell Systems' Masterfile 128 will provide a 64K filespace and interfaces with Tasmans's software.

2) Some software, whether loaded from disc/tape or booted from a background ROM will check the ROM identity by using the firmware call &B915.

   There is one more command included in the RSX software on tape whichwill cause a CPC464 or 664 to emulate the ROM identity of the CPC6128.

   Type '|EMULATE' and press <ENTER>

   Any programs that call the ROM identity routine will now be informed that the computer is a CPC6128 and may now work correctly.

3) The software may use some features of the CPC6128 ROM which are unavailable on the CPC464 and CPC664 machines. In this instance, you may be able to get information on how the program can be altered to work on the CPC464 or 664 from the manufacturers of the program in question.

3.11.4 Using CP/M 2.2 (R)

CP/M 2.2 (R) as supplied with all Amstrad computers will function as normal with the Extra memory fitted. However if you run the INCTPA.COM utility the TPA available to CP/M 2.2 will be increased to 61K.

Programs of your own devising written under this operating system are free to use the extra memory. See section 3.10 for details of how to use the extra memory from machine code.

3.12 ERROR MESSAGES

While you are using the RSX software, there will be some occasions when the computer does not understand, or cannot carry out what you have instructed. The software may issue some error messages in addition to the normal messages that the computer will give. The errors and why they are likely to occur are outlined below:-

Bad bank command — Given if you have given the wrong number of parameters or if a variable is not present where there should be one.

Bank unavailable — You have tried to access a bank which is not present on your system.

Bad bank parameter — You have referenced a bank which can never be fitted to the computer.

Bad bank address — The address you have given is out of range: bank addresses range from 0 to 16383.

Value invalid — The bank address may be too large for the block of data defined. The parameter for |ASKRAM is other than 1, 2 or 3. The size or a block to be saved is larger that 16K.

Bad window definition

The window referenced in |SAVEW or |LOADW is above 7.

3.13 REFERENCE OF RSX COMMANDS

All the additional commands are listed below as a reminder to their functions and syntax.

SCREENS

```
|SAVES, [ bank ], [ swap ]
|LOADS, [ bank ], [ swap ]
```

WINDOWS

```
|SAVEW, [ window number], [ bank ], [ bank address ], [ swap ]
|LOADW, [ window number], [ bank ], [ bank address ], [ swap ]
```

DATA BLOCKS

```
|SAVED, [ bank ], [ Start location ], [ length ], [ bank address ]
```

ANIMATION

```
|LOW  (Low screen)
|HIGH (High screen)
|SWAP (Alternate between High and Low screens)
```

OTHER

```
|POKE, [ bank ], [ bank address ], [ value ]
|PEEK, [ bank ], [ bank address ], [ variable ]
|BANK, [ bank ]

|ASKRAM, [ equiry ], [ variable ] ([ enquiry ], 1 = RAM, 2 = banks,
3 = error occured?)
```

DEFINITIONS

[bank]	Bank number 1-4 or 1-6 for 64K and 256K expansions.
[bank address]	Address within bank 0 to 16383.
[swap]	0 or omitted means act on present screen, 1 means act on alternate screen.
[start location] and [length]	Define a block of original memory.
[variable]	Give the location of an integer variable to be assigned for example @b%.

3.14 Technical details (hardware organisation)

These interfaces add either a single block (64K) or four blocks (4 x 64K) of RAM to an existing CPC464, 664 or 6128.

Thus, if 64K (one block) is added to a 464, the total memory is two blocks, 128K.

For a given set-up, calculate the total number of 64K blocks, this will determine which of the block select codes mentioned later are relevant to your system. The blocks are referred to by number, block one is the original 64K, block two is equivalent to the second block present in the 6128, and so on.

Memory is actually switched in and out of the 64K Z80 address space in 16K subblocks (as are the ROMs). Which particular combination of the original four 16K sub-blocks used, and any 'new' sub-blocks from ram beyond the original 64K, is called the memory map. The map is determined by an 8-bit code byte sent to the gate array control port, &7F00, with the two top bits set to 1. The following description of the codes referes only to the remaining six bits, D5-D0.

Control Codes
Bits D2-D0 control the way 16K sub-blocks are arranged in the Z80 memory space, bits D5-D3 control selection of whichever 'new' 64K block is to be used.

Bits D2-D0 - 16K Map Codes
These bits select one of the eight maps into the 64K as follows:-

CODE	0	1	2	3	4	5	6	7
SUBBLOCK								
C000-FFFF	3	3*	3*	3*	3	3	3	3
8000-BFFF	2	2	2*	2	2	2	2	2
4000-7FFF	1	1	1*	3	0*	1*	2*	3*
0000-3FFF	0	0	0*	0	0	0	0	0

The numbers 0, 1, 2, 3 refer to the four 16K sub-blocks in a 64K block in the obvious way. The star (*) indicates that the memory is from a 'new' block, ie block 2 or higher, otherwise the 'original', block 1, is implied.

Thus, code 0 selects the original, unmapped 64K, code 2 selects a completely new block of 64K, the other codes are a mixture.

Notes

1. On power-up, code 0 is selected.

2. The VDU circuitry <u>always</u> reads from the original 64K (block 1),independently of the code.

3. If code 3 is used, reads from &4000 to &7FFF, on CPC 464/664 machines, will only return the correct data if the upper ROM is disabled. This is at variance with CPC 6128 operation, but is unlikely to be a significant difference.

4. If code 3 is used, addresses &4000 to &7FFF must not be used to run programs, they are intended for VDU or data access only.

Bits D5-D3, 64K Block Select Codes

D5	D4	D3	BLOCK
0	0	0	2 (ie, 'new' memory sub-blocks came from block 2, as in CPC 6128)
0	0	1	3
0	1	0	4
0	1	1	5

Which of the above codes are relevant to your machine depends on total memory (see previous remarks).

Notes

1. On power-up, code 0, 0, 0 is selected.

2. Bits D5, D4, D3 above 'count up' as blocks are selected. This may assist the programmer.

3. if 2 x 256K memory expanders are fitted to the machine, and option links are set appropriately, all patterns on D5-D3 can be used, giving a maximum of 512K extra memory.

3.15 CUSTOMISING YOUR CP/M PLUS SYSTEM DISC

Converting A 464 Keyboard Scan To That Of A 6128.

Some CPM PLUS programs will not run correctly on the 464 computer because of the way the 6128 scans the keyboard.

The following program and instructions convert (fool), CPM PLUS into thinking that it is running on a 6128.

1. Make a working copy of both sides of your system disc (sides 1 and 2).

   a) Use your standard CP/M 2.2 System disc and type '|CPM'.

   b) For Single drive systems use DISCCOPY. For dual drives use COPYDISC. (Remember to do both sides).

c) Put your original system disc away in a safe place to keep as a back-up incase your working disc is damaged,

2, Put 'BANK' and 'RSX' onto side 1 of the working disc,

a) Follow the instructions in the manual to get a copy onto disc, Alternatively, use 'FILECOPY BANK,BAS' and 'FILECOPY RSX,BIN' if you have transferred the software to disc already,

3, Reset the machine, then enter CP/M PLUS, Only one drive is necessary but if you have two you must disconnect the second drive because all the changes are to be made on the working disc and with a single drive the computer can use both sides,

a) RUN "BANK" then press <ENTER> in response to LOAD ADDRESS?,

b) Type in '|EMULATE:|CPM' then press <ENTER>,

4, Type in the following, pressing <ENTER> after every line unless otherwise written, Turn the disc over when the computer asks;-

```
ED PATCH, ASM
i
ORG 100H
XRA A
STA 0FDEFH
JMP 0
END
<CTRL>Z                     do not press <ENTER>
e
ED PROFILE,SUB
i
PATCH
<CTRL>Z                     do not press <ENTER>
e
```

(Insert disc containing **MAC,COM FILE**)

```
B;MAC PATCH
B;HEXCOM PATCH
ERA PATCH,HEX
ERA PATCH,SYM
ERA PATCH,PRN
ERA PATCH,ASM
```

5, It is possible to alter the CP/M PLUS disc to boot up without loading the DK'tronics BANK program first, Type the following if you want your system disc altered in this way;-

```
B;SAVE
B;SID C10CPM3,EMS
S1E0
C9
<CTRL>C                       do not press <ENTER>
C10CPM3,EMS
Y
100
6500
```

The disc will now boot up without BANK being run,

64/256K SILICON DISC

The 64/256K silicon discs are designed to be used with at least one normal disc drive attached. Data can be transferred onto the silicon disc from a normal disc. Application programs can then work on the data at vastly increased speed, especially on systems with only one normal drive. Support Software is contained in an expansion ROM.

The Silicon Disc is compatible with all DK'tronics peripherals.

There are two environments in which to use the silicon disc:-

1) From BASIC under AMSDOS.
2) Within CP/M

1) From BASIC:

When the silicon disc is activated it will find out if there is a B drive or not. Using this information, the silicon disc is implemented as drive B or C. If there are two normal drives then an extra command '|C'is added. The silicon disc can then be accessed by logging on the drive by using |B or' |C. Alternatively specifying the drive letter in a file name will have the same effect.The silicon disc will react as would normal Amstrad disc drives.

At the start of a session using the computer, the data can be transferred to the silicon disc drive using the external command '|LOADDISC'. When this data is updated it can be stored on the normal disc using the '|SAVEDISC' command.

Once the silicon disc has been initialised it accepts all the normal AMSDOS commands. (LOAD,SAVE,CAT and disc files etc).

Even when the computer is reset (except by switching off the power) the contents of the silicon disc are kept intact. This means that it is possible to use CP/M and BASIC programs on the same data files without having to continually change discs.

2) From CP/M :

The utility |SETDISC will write a COM file on a copy of your CP/M system disc. This program when called from CP/M will implement an additional drive as drive B or C. Using the SETUP program you can get this program to run whenever you boot into CP/M.

Once the drive is implemented, CP/M will treat it like the normal drive(s). Data can be transferred to and from the silicon disc via the PIP utility as normal.

The silicon disc is especially useful for single drive CP/M systems as the disc containing the program is often nearly full and needs to stay in the drive. The silicon disc offers a cheap second drive for serious business applications.

CP/M is a registered trademark of Digital Reserch Inc.

64K Silicon disc

This is available to 6128 users without the need for further expansion as it utilises the second 64K of banked RAM,which is fitted as standard, as a Ram Disc.

464/664 users will need to fit the DK'tronics 64K RAM pack.

The 64K silicon disc operating system cannot be used with CP/M PLUS because they would both be trying to use the second 64K of memory.

SILICON DISC MANUAL

WARNING

Ensure that the power to your Amstrad computer is switched OFF before you fit the Silicon Disc to the expansion socket. Failure to comply with these instructions may cause permanent damage to the device or the computer.

4.1 Installation

This manual covers two versions of the silicon disc, the 64K and the 256K.

The software in ROM is compatible with the CPC 464/664 and 6128 computers. All programs in this manual have been designed to run on the CPC 464 where the CPC 664/6128 are UPWARD compatible. You may however wish to use the extra features of the BASIC 1.1 machines.

The 64K silicon disc is contained in one unit (Operating system ROM), while the 256K silicon disc is in two units (256K RAM and Operating system ROM).

Power down the computer and fit the Silicon Disc unit(s) onto the rear of your Amstrad. On the CPC 464 this socket is labelled 'Floppy Disc', on the CPC 664/6128 it is labelled 'Expansion'.
The Silicon Disc will only work in conjunction with the normal disc system. On the CPC 464 the FDI-1 MUST be fitted. Other expansions may be fitted into the expansion socket on the rear of the Silicon Disc. now switch on the computer.

The computer should power up as normal. If it fails to do so, check that all the connections are correctly made. Note that all DK'tronics products have a key location on the connector to ensure that there can be no alignment problems. OTHER interfaces may not use this keyway (the Amstrad disc interface is the most familiar example). Hence any connection problems will usually lie between the Silicon Disc and these expansions. If this is the case, try reconnecting the interfaces BEFORE inserting the Siliocn Disc onto the expansion slot. This will give you a better view when lining up the pins.

If the computer DOES fail to power up due to misalignment problems the monitor may cut out the power to the computer unitl the MONITOR is switched off. This is not due to overload but is safety device built into the monitor to protect the computer and interfaces. On the colour monitor just switch the MONITOR off, reconnect the units as above, then switch on again. The monochrome monitor will have to remain switched off for quite a few seconds before the reset is cleared. Reconnect as above, wait for a few seconds, then switch the monitor back on.

It is very unlikely that the computer (CPC 664/6128) will fail to power up with the Silicon disc alone. If this is the case, then the fault will probably lie with the Silicon Disc. * Return the Unit to DK'tronics if this is the case.

* IT IS ESSENTIAL THAT YOU COMPLETE YOUR WARRANTY REGISTRATION CARD AND RETURN IT TO US IMMEDIATELY UPON PURCHASING THIS PRODUCT FROM YOUR DEALER (UK ONLY).

4.2 FIRST STEPS

When the computer is powered up the ROM should sign itself on as:-

```
    'Silicon Disc 0.1'  (for use with  64K BANKED RAM)
    'Silicon Disc 1.1'  (for use with 256K BANKED RAM)
```

The silicon disc requires a store of 450 bytes which are taken from the memory store before BASIC is entered. The Amstrad disc ROM must set up all the disc commands before the Silicon disc is started and so trying to access the silicon disc now will give an error of drive not present.

Type '|SDISC' and the computer will implement the silicon drive.

Once the Silicon disc has been called an additional drive will be available to the system:-

On a single drive system the drive is 'B'
On a dual drive system there is a new drive 'C'

Try typing '|B:CAT' or '|C:CAT'.

All the commands which are normally used in BASIC for the standard drives now apply also to the silicon disc. The only difference you should notice is the difference in access time for files and programs.

When the computer is first switched on the Silicon Disc is empty. The catalog shows that there is 62/254K free. With 2K for the directory that's 64/256K!

Obviously that is not a lot of good, you want to put programs onto the silicon disc and take them off. There are two commands which allow bulk data to be moved from a normal disc (Drive A) to the silicon disc and vice-versa:-

```
    |LOADDISC Move normal disc's contents to silicon disc
    |SAVEDISC Put data back onto a normal disc
```

The first command might be used at the start of a progamming session to move all your data to the silicon drive. The second would be used at the end to move all the data back to a normal disc. **(Remember the contents of the silicon disc are lost if the computer is switched off!.)**

The '|LOADDISC' command transfers ALL the files from the normal disc to the Silicon disc. With a maximum size of 178K you will never fill the 256K Silicon Disc from a normal disc. However if you try to transfer more than 62K of data from the normal disc to the 64k Silicon Disc then the system will lock up.

The '|SAVEDISC' command also transfers all the files from the Silicon Disc to the normal disc but will not work if there is too much data on the silicon disc. Some BAK files etc will have to be removed first. If you wish to selectively save files it is best to use PIP in CP/M.

Note that both the |LOADDISC and |SAVEDISC commands use BASIC memory to store the disc catalog so that the transfer is completed as fast as possible. This means that BASIC programs are lost. Hence both commands are best used before or after a programming session.

Remember also that each disc copy will REPLACE the original contents of the drive you're copying to even if no files are present on the disc being copied. The '|SDISC' command must be issued after using the '|LOADDISC', '|SAVEDISC' commands if you want to continue using the Silicon drive.

Once the silicon disc has some programs on it, these can be loaded simply by using the standard LOAD and SAVE commands. Try loading some of your long programs from normal disc and then Silicon Disc to see the increase in speed.

The |LOADDISC and |SAVEDISC commands automatically check whether you mind the BASIC area being used. If you respond with 'n' then they will abort back to BASIC, otherwise they will carry on. If, however, you add an optional parameter of 1 then the check will not be made and the commands will take action straight away.

The |SAVEDISC command can also take a parameter of 2 which tells it to save all files onto a normal disc except the COM files. This is useful if you have too much data on the silicon disc so that it would fill the normal disc. You may not mind losing copies of your COM files as these are often stored seperately on your system, disc.

4.3 COMPATIBILITY

BASIC programs using:-

```
    SAVE, LOAD, MERGE, CHAIN, RUN", CHAIN MERGE, CAT, OPENIN, OPENOUT,
    CLOSEIN, CLOSEOUT, INPUT #9, PRINT #9, LINE INPUT #9, WRITE #9,
    LIST #9, EOF
```

will function as normal. If the program was written to access only drive A then it will need to be altered to allow for drive B or C. Alternatively the drive letter could be specified:-

```
   eg SAVE"C:PROGRAM1.BAS" or RUN"B:DEMO"
```

All the standard external commands as supplied with AMSDOS will funciton:-

```
    |A, |B, |C (dual drive only), |DISC (.IN and .OUT),
    |TAPE (.IN and .OUT), |CPM, |DIR, |DRIVE, |ERA, |REN, |USER
```

Typing |C or |B will set the silicon disc as default and all the above commands will function on the silicon disc unless the drive letter is specified.

The Silicon Disc will activate the Amstrad's error trapping if this is set up and the error number returned by DERR and ERR are the same.

Some errors cannot occur on the Silicon Disc:-

```
    DERR = 149: Disc changed with files open

    and Drive: disc missing
        Drive: disc is write protected
        Drive: read fail
        Drive: write fail
```

The silicon disc cannot be removed, so there is no problem with logging the disc on or writing files to it. Because the disc is RAM there can be no read or write errors. The normal disc drives will issue all the error messages that it normally does.

The command |TAPE will switch in the tape unit as normal; |DISC will switch both the normal disc dirve(s) and the silicon disc back in.

The filenames are identical for both normal and silicon drives. The system will default to .BAS or .BIN extensions where applicable, also .$$$ files are created before files are closed and .BAK files for one level backup.

4.4 USING YOUR SILICON DISC IN CP/M 2.2

Before the silicon disc can be used under CP/M, the silicon disc program needs to be stored on a copy of your system disc. With the computer just powered up, and a copy of your system disc in drive A, if you have not already got a copy of your system disc to use then make a copy as described in Amstrad's manual:-

Type '|SETCPM' and press <ENTER>

The computer will write a file onto the system disc called'SDISC.COM'

Enter CP/M by typing '|CPM' the press the <ENTER> key.

When the 'A>' prompt appears, type 'MOVCPM 176 *' and press the <ENTER> key then type in 'SYSGEN *' and press the <ENTER> key.

This will move the top of CP/M down so that there is some spare memory for the program to use. This change will not affect the majority of your programs as the amount of memory used is so small. However the Amstrad DISCOPY, COPYDISC and other programs use the whole of the memory so it is best to keep an unaltered system disc for when you want to copy discs. The use of PIP is not affected so it is probably best to use PIP (|LOADDISC. and |SAVEDISC) in conjunction with the Silicon disc when you want to copy discs.

If you now type 'SDISC', the computer will start up with the Silicon disc. You do not need to type MOVCPM and SYSGEN every time, just SDISC. The Silicon disc can be set to automatically start when you boot CPM:-

Type 'SETUP' and press <ENTER>

The first question SETUP will ask is to set up the initial command buffer. Select 'n' option so that you can change the start up buffer. Now enter 'SDISC↑M' the '↑M' will cause the computer to type the SDISC command whenever CPM is entered. Answer 'y' to all the other questions and 'y' to altering the system disc. Note that the '↑' key is located on the same key as the '£' key.

The system disc is now finished. Try Resetting and booting CPM to see how it works.

Once SDISC has been activated, an extra drive will be available as drive B or C depending on whether you have a second normal drive.

All the standard CPM programs will now use the silicon disc if you specify the silicon drive's letter.

4.4.1 USING YOUR SILICON DISC IN CP/M PLUS

NOTE that the 64K Silicon Disc cannot be used under CP/M PLUS because there will be a conflicting access to the banked RAM.

Before the 256K Silicon disc can be used under CP/M PLUS a patch to the early morning start-up file (.ems) is required. This will support a RAM drive as C:. The following steps are necessary:-

1) Create a new disc with a copy of the .ems file

2) From BASIC type in '|SETCPMPLUS' then press <ENTER>. This will place a copy of the patcher.com file onto the disc.

3) Boot CP/M PLUS using this disc in drive A: The new disc must have at least 25K free !.

4) Immediately following the A> type 'patcher' then press <ENTER>. The patcher program will now create a new version of the .ems file erasing the old copy at the same time.

5) Reboot CP/M plus using this new file. The silicon disc will now log on automatically as drive C:.

The capacity of this disc will be one of the following :-

a) 190K - if only a 256K DK'tronics memory expansion is available.

b) 254K - if only the DK'tronics silicon disc RAM is available. (compatible with current CP/M 2.2 version)

c) 444K - If both the DK'tronics 256K memory expansion and the silicon disc are available.

The new .ems file automatically works out how much extra RAM you have on the system and configures it accordingly.
The RAM disc directory is automatically initialised to contain no files the first time this new .ems file is run. However, if the files have been placed in the RAM disc on a previous occasion, the directory is not formatted, assuming the computer has not been switched off. This allows you to enter CP/M 2.2 or AMSDOS and then return to CP/M PLUS with your files still present in drive C:.

4.5 ADVANCED NOTES

Both under AMSDOS and CPM the individual file's write disable and system status are complied with.

The use of the silicon disc from machine code can be easily implemented by using FAR calls to the external commands and by using the cassette calls as documented in the Amstrad manual. All the interfaces are the same so existing software sould run fine. The external commands which read and write to specific sectors, format and move the head on the normal two drives (CTRLA to CTRLL) will not function on the silicon drive and in fact may issue an error message that the drive is not present. Any direct reading and writing can be done by bank switching the 64/256K of memory into the main memory map 16K at a time.

The 16K memory block is switched in and out of the main memory map, (16384 to 32767), by sending the appropriate code to the port location &7F00.

There are 4 banks of 16K for the 64k banked RAM and 16 banks of 16K for the 256K banked RAM. The numbers to send to the port are as follows:-

64K Silicon Disc

196, 197, 198, 199

Therefore if you wish to select a bank you wold use the following :-

```
    OUT &7F00,( BANK + 196 )            where BANK has a value of 0 - 3
```

256K Silicon Disc

228,229,230,231,236,237,238,239,244,245,246,247,252,253,254,255

There is a simple formula to calculate the correct value for the bank you want:-

```
    VALUE=228+(BANK AND 3)+8*(BANK/4)            where BANK is 0 - 15
```

Therefore if you wish to select a bank you wold use the following :-

```
    OUT &7F00, VALUE
```

To restore the original 16K block VALUE would be set to 192.

The first bank holds the catalog and the first 14 blocks of 1K.

dk'tronics Limited,
Longs Industrial Estate, Englands Lane, Gorleston, Great Yarmouth,
Norfolk NR31 6BE, England. Tel: (0493) 602926 (5 Lines).

Printed by Lowestoft Printing Co. Ltd., Walton Road Tel: (0502) 2502.

AMSTRAD

PERIPHERAL – TECHNICAL MANUAL

EDITION 2

dk'tronics®
Limited