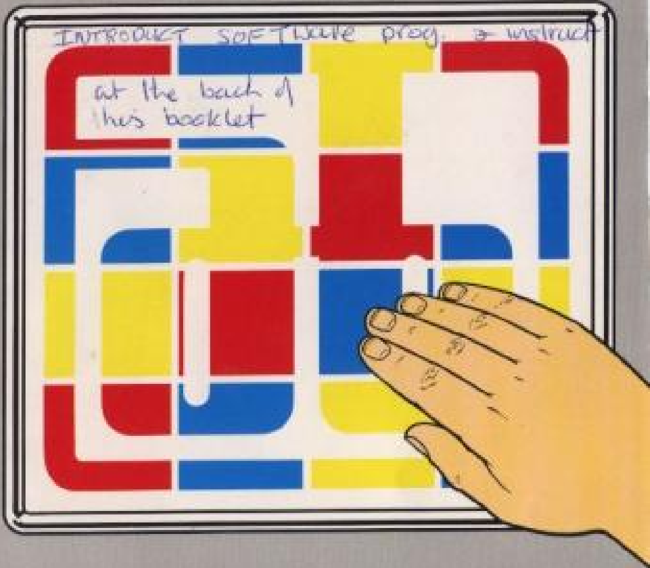
AB European Marketing, Wharfdale Road, Pentwyn, Cardiff, CF2 7HB.
Telephone: 0222 733485 Telex: 497686 Fax 0222 736699



INTRODUCT SOFTWARE prog. & instruct
at the back of
this booklet

CONCEPT KEYBOARD

USER GUIDE

BBC MICROCOMPUTER VERSION

Eastbury Lower Sch

CONCEPT KEYBOARD USER GUIDE
BBC MICROCOMPUTER VERSION

Written by Dr. I. C. Brown,
Senior Lecturer, City of London Polytechnic

Designed by Royds McCann Wales Ltd.

Published by AB European Marketing,
Wharfdale Road, Pentwyn,
Cardiff CF2 7HB.
Telephone: (0222) 733485.
Telex: 497686.
Fax: (0222) 736699.

The product described in this manual and products for use with it, are subject to continuous developments and improvement. All information of a technical nature and particulars of the product and its use (including the information in this manual) are given by A B Electronics in good faith. However, it is acknowledged that there may be errors or omissions in this manual. A list of details of any amendments or revisions to this manual can be obtained upon request from A B Electronics. A B Electronics welcome comments and suggestions relating to the product and this manual.

All correspondence should be addressed to:

AB European Marketing,
Wharfdale Road, Pentwyn,
Cardiff CF2 7HB.
Telephone: (0222) 733485.
Telex: 497686.
Fax: (0222) 736699.

All maintenance and service on the product must be carried out by A B Electronics. A B Electronics can accept no liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore A B Electronics shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

The author would like to thank Mr. M. P. Doyle who wrote the Provisional Documentation for the earlier models and Mr. David Stubley of Star Microterminals Ltd., for his help with the production of this guide.

CONTENTS

Preface
1. Introduction
2. Getting Started
3. Writing Programs
4. Adapting Existing Programs
5. Principles of CONCEPT Keyboard Program Design
6. Overlay Design Information
7. Technical Information
Appendix

PREFACE

There is always a temptation to skip introductions!

Although no physical damage can be done by jumping immediately to Chapter 2, it is worth stopping, fairly early on, to consider not only where the CONCEPT Keyboard plugs into your computer but also where this new "CONCEPT" fits into the overall strategy of using computers for educational and other purposes.

This booklet is certainly not the last word on the CONCEPT Keyboard. It is expected that an Advanced User Guide will follow. The present User Guide is written for the non-specialist user, avoiding unnecessary computer jargon and advanced programming techniques. The reader who requires technical information is referred to Chapter 7. We have tried to cater for the reader who boasts only a very limited knowledge of BASIC but nevertheless uses packaged programs for the conventional keyboard.

More and more specially designed CONCEPT Keyboard programs are appearing. Even if you intend only to use ready made software or perhaps use a package to convert conventional programs, it is hoped that you will enjoy creating a few simple CONCEPT Keyboard programs of your own. When you have understood the principles your imagination may well take you on to create new areas of application for this most versatile device. It is currently being used successfully in such fields as agriculture, banking, education, industry, computer aided design (CAD), hospital work, leisure activities and as an aid for the disabled.

1. INTRODUCTION

Back in the early days of computers, users were compelled to communicate with their machines by means of a 'typewriter' keyboard. This was something of a historical accident; the QWERTY Keyboard was an established convention, early computers were mostly for scientific or business applications and the stenographer's keyboard continued in use.

Today, computers are widely used in a variety of environments and it is no longer reasonable or necessary to use a full word-processing keyboard for every application. For example, if very young children are asked by a program to choose between two colours, they may now be given a keyboard consisting simply of two coloured regions. If trainees on an engineering course are asked to select a particular component, let them simply touch one of a selection of diagrams. There have been many cases in the past where the task of finding and depressing the right keys has been harder or more time consuming than the actual objective of the program. Not only are these problems removed by the CONCEPT Keyboard but a new approach to Computer Aided Learning is opened up.

Just as the computer's communication to the user is no longer restricted to text on the screen, but employs exciting and colourful graphics, so the CONCEPT Keyboard user can reply not only by means of text but by touching overlays which might include such items as pictures of animals, musical notes, the words of a reading scheme, diagrams or coloured shapes.

The traditional flow of information between a human and a computer system was as shown in Figure 1 below.

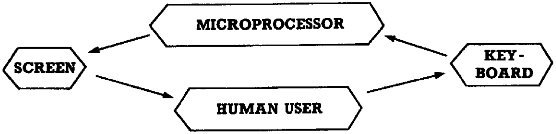


Figure 1

The human user sent information via the keyboard to the central processing unit. After doing its work, the microprocessor sent information to the user via the screen. With the arrival of the CONCEPT Keyboard an important new link is introduced into the chain. The CONCEPT Keyboard **conveys** information to the user. Information about the program is supplied to the user by the overlays **as well as** the VDU screen (Figure 2).

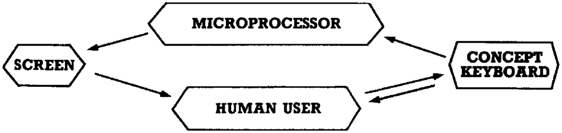


Figure 2

The relevant point for the CONCEPT Keyboard program designer is that the board should not only be seen as an input device, but also as a means of informing the user about the program (figure 3).

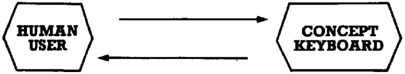
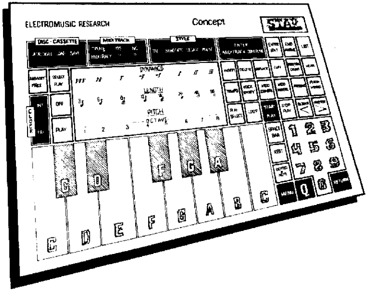


Figure 3

As a conveyor of information the CONCEPT Keyboard is very powerful and versatile. The QWERTY Keyboard does not help or invite the human to use it! It often initiates fear and usually requires training or practice before it can be used efficiently. The CONCEPT Keyboard overlay, on the other hand, can be designed to look informative and attractive and provide the most appropriate input medium for any particular application.



2. GETTING STARTED

Unpacking

First check that you have received the following items in addition to the User Guide.

- CONCEPT Keyboard
- Interface Lead
- Introductory Software
- Overlays

Plugging in

With your computer switched off, take the Interface Lead and

- Connect the plug labelled CONCEPT to the CONCEPT Keyboard.
- Connect the plug at the other end to the user port underneath the computer.

Switching On

Switch on the computer and the red light should come on in the top left hand corner of the CONCEPT Keyboard. When you touch the CONCEPT Keyboard surface it should emit a beep. (If it does not, try again after pressing the cell marked "bleep on/off" at the top of the board. This is an on-off switch for the bleep.).

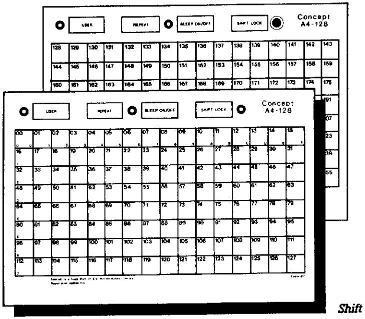
Introductory Software

You may now use the Introductory Software supplied with this keyboard. See Appendix for detailed instructions. (The software will not work on the A3-256 version).

Understanding the Hardware

On your CONCEPT Keyboard you will see 128 or 256 rectangles on the model you have chosen. Each one of these touch-sensitive areas or "cells", when pressed, sends a unique message to your computer. This coded signal is fed directly into the computer's memory.
In addition to the grid of cells there are some larger cells or "pads" at the top of the board. These are, from the left,

User pad	This may be ignored for the present.
Repeat Pad	Normally, when a cell is pressed the appropriate signal will be sent to the computer once, no matter how long you take to remove your finger. However, if you keep pressing any cell and then press the repeat pad the signal will be sent to the computer repeatedly until you remove your finger.
Bleep ON/OFF Pad	This acts as an ON/OFF switch for the bleep which is emitted when the system has sensed that a cell has been pressed. Press the pad once to silence the bleep. Press it again to restore it.
Shift pad *(See illustration page 11)*	(128 Cell models only) The provision of this pad enables 256 different signals to be produced from the 128 cells. The shift pad acts as an ON/OFF switch for the SHIFT facility. The ON/OFF status of the SHIFT is shown by the red light on the right hand side of the shift pad as follows: Light OFF – output in the range 0 to 127, Light ON – output in the range 128 to 255. The red light is known as a light emitting diode (L.E.D.) If the SHIFT is in (L.E.D. ON) the operation of some early software may be affected. To overcome this, press the SHIFT PAD until the L.E.D. is OFF. If use is to be made of the full range of outputs (0 – 255) the relevant programs must be used. In particular, listings for '256 Cell Models' should be selected where alternatives are offered later in this guide.



Shift Pad

3. WRITING PROGRAMS

Chapter 2 has shown how to send messages from the CONCEPT Keyboard to the computer's memory. So far the numbers have no meaning and when the computer receives them it does nothing with them (figure 4).

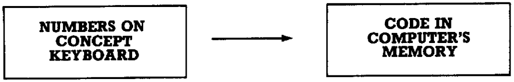


Figure 4

Two things are needed: an overlay for the CONCEPT Keyboard and a program for the computer.

An overlay is a sheet of paper or thin card which exactly covers the CONCEPT Keyboard and gives meaning to the cells beneath it. It may give the same meaning to a group of adjacent cells. Such a group is called a "response area" because all its cells produce the same response when pressed. Some cells may be given no meaning and left "dead"; nothing happens when you press them.

A CONCEPT Keyboard program is one which enables the computer to process the encoded number which arrives in memory from the CONCEPT Keyboard. For example, the program may resemble any other BASIC program except that statements which wait for standard keyboard input (e.g. INPUT, GET) are replaced by a call to a short procedure.

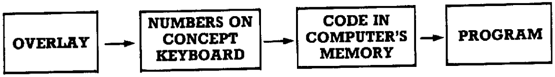


Figure 5

As a simplified illustration, suppose the user is intending to make a selection from a number of fruits. An apple on the overlay covers cells 36, 37, 52 and 53. A child presses part of the apple over a certain cell. The number of that cell, say 52, is encoded and sent to the computer's memory. The program recognizes the number 52 and, finding it in the list 36, 37, 52, 53, generates the appropriate response. We have seen that what is needed is a procedure which reads the contents of a particular location in the computer's memory and gives a value to some variable (we use CK) which may be used in a BASIC program. Such a procedure is given below for the BBC computer. (listing 1).

Even slow typists will see that the procedure will not take very long to type in! Non-experts will be pleased to know that it is not necessary to understand what the statements mean. Just enter them **accurately** at the end of each program for which you wish to use the CONCEPT Keyboard. This procedure will ensure that a variable called CK will be set equal to the number of the cell pressed.

```
3000 DEFPROCCK
3010 *FX 151,96,0
3020 *FX 151,98,0
3030 *FX 151,107,2
3040 A%=&96:X%=&6D
3050 REPEAT UNTIL
     USR(&FFF4)AND &100000
3060 X%=&60
3070 CK=(USR(&FFF4)AND
     &7F0000)DIV &10000
3080 ENDPROC
```

LISTING 1
(128 CELL MODELS)

```
3000 DEFPROCCK
3010 *FX 151,96,0
3020 *FX 151,98,0
3030 *FX 151,107,2
3040 A%=&96:X%=&6D
3050 REPEAT UNTIL
     USR(&FFF4)AND &100000
3060 X%=&60
3070 CK=(USR(&FFF4)AND
     &FF0000)DIV &10000
3080 ENDPROC
```

LISTING 1
(256 CELL MODELS)

THIS IS PROCCK. SAVE IT ON DISC OR TAPE FOR FUTURE USE.

If, in the main program, we use the statement

PROCCK

the program will pause at this statement until the CONCEPT Keyboard is touched. It will then continue with the variable CK set equal to the number of the touched cell.

A single choice of response

In the simplest case, all the cells form a single response area. A program may pause at given points until the user presses the CONCEPT Keyboard (anywhere). Listing 2 illustrates how such a strategy may be implemented.

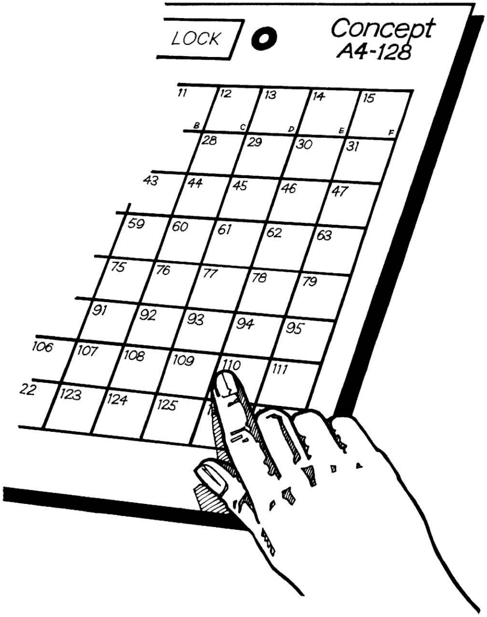
```
  10  PROCCK
  20  PRINTCK
1000  GOTO 10
2000  END
3000  DEFPROCCK
3010  *FX 151,96,0
3020  *FX 151,98,0
3030  *FX 151, 107,2
3040  A%=&96:X%=&6D
3050  REPEAT UNTIL
      USR(&FFF4)AND &100000
3060  X%=&60
3070  CK=(USR(&FFF4)AND
      &7F0000)DIV &10000
3080  ENDPROC
```

LISTING 2
(128 CELL MODELS)

```
  10  PROCCK
  20  PRINTCK
1000  GOTO 10
2000  END
3000  DEFPROCCK
3010  *FX 151,96,0
3020  *FX 151,98,0
3030  *FX 151,107,2
3040  A%=&96:X%=&6D
3050  REPEAT UNTIL
      USR(&FFF4)AND &100000
3060  X%=&60
3070  CK=(USR(&FFF4)AND
      &FF0000)DIV &10000
3080  ENDPROC
```

LISTING 2
(256 CELL MODELS)

Beginners may find this listing useful to provide their first experience of running a CONCEPT Keyboard program. The number, CK, of the touched cell is passed from the procedure to the main program. It is simply printed by line 20 to show the user that the program is working.



Two response areas

We now divide the CONCEPT Keyboard in half, down the middle. In order to inform our program which side has been touched, we reserve a variable S (say, which we term the "selector") and assign

S=0 if the left hand side is touched.
S=1 if the right hand side is touched.

This is achieved by adding

```
15 S=INT(CK MOD 16/8)
20 PRINTCK, S
```

to Listing 2. Running the program with the amended lines the reader will be able to check that the correct selector value S is given to any cell.

Four response areas

Let us divide the board into four equal rectangles (as shown in figure 6) and assign

S=0 if the top left area is touched,
S=1 if the top right area is touched,
S=2 if the bottom left area is touched,
S=3 if the bottom right area is touched.

This may be achieved by extending line 15 to read

```
15 S=INT(CK MOD 16/8)+2*INT(CK DIV 64) for 128 cell models
                                          (If SHIFT OFF)
```

OR

```
15 S=INT(CK MOD 16/8)+2*INT(CK DIV 128) for 256 cell models
```

(The same effect could be obtained by simply inserting a line which adds two to the selector if the bottom half of the board is touched, but the new line 15 suggests a more efficient way to proceed to more general cases).

Before proceeding it is worth typing in a simple example program. You should have most of it already from Listing 2. Lines 15 to 70 are new.

```
  10 PROCCK
   15 S=INT(CK MOD 16/8)+2*INT(CK DIV 64)
  20 CLS
  30 PRINT'''
  40 IF S=0 THEN PRINT "Apple"
  50 IF S=1 THEN PRINT "Orange"
  60 IF S=2 THEN PRINT "Pear"
  70 IF S=3 THEN PRINT "Banana"
1000 GOTO 10
2000 END
3000 DEFPROCCK
3010 *FX 151,96,0
3020 *FX 151,98,0
3030 *FX 151,107,2
3040 A%=&96:X%=&6D
3050 REPEAT UNTIL USR(&FFF4)AND &1000000
3060 X%=&60
3070 CK=(USR(&FFF4)AND &7F0000)DIV &10000
3080 ENDPROC
```

Listing 3
(128 CELL MODELS)

Note that for 256 cell models lines 15 and 3070 should read

```
   15 S=INT(CK MOD 16/8)+2*INT(CK DIV 128)
3070 CK=(USR(&FFF4)AND &FF0000)DIV &10000
```

For the overlay divide the board as in figure 6 and cover the regions marked 0,1,2,3 with pictures of an apple, an orange, a pear and a banana respectively. It is a simple matter to alter the program in lines 40 to 70 for a different set of pictures. The artwork of the overlay may either be drawn by hand or pictures from magazines etc. may be stuck down. In some cases actual objects such as coins may be attached to the overlay.

Sixteen response areas

We now proceed to divide the board into 16 equal rectangles and assign the selectors as shown in figure 7. The required replacement lines are:

```
15 S=INT(CK MOD 16/4)+4*INT(CK DIV 32) for 128 cell models
```

OR

```
15 S=INT(CK MOD 16/4)+4*INT(CK DIV 64) for 256 cell models
```

Then add to either version:

```
 80 IF S=4 THEN PRINT "Plum"
 90 IF S=5 THEN PRINT "Lemon"
100 IF S=6 THEN PRINT "Cherry"
110 IF S=7 THEN PRINT "Grape"
```

The general case

Readers will now be able to generalise these ideas to other symmetrical configurations using the MOD function to identify columns and the DIV function for rows. However, as required layouts become more complicated and less symmetrical, it is less appropriate to rely on mathematical formulae for the prescriptions of S. Instead we use a DATA statement which serves as a look-up table to relate each cell number, N, to a selector, S. A zero value of S will denote a "dead" cell, i.e. touching this part of the CONCEPT Keyboard will cause no action. This technique is best illustrated by means of examples.

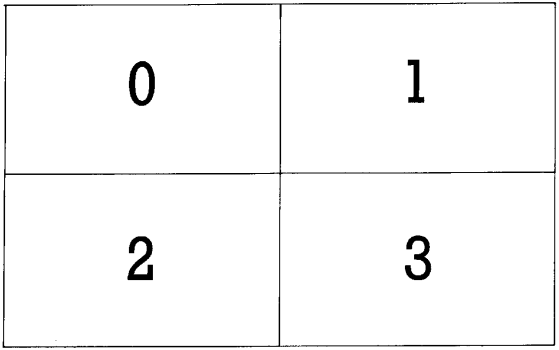


Figure 6

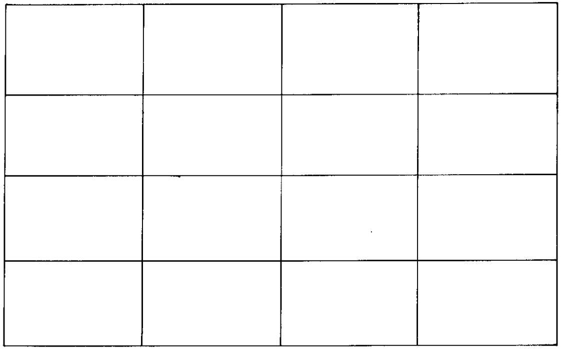


Figure 7

Suppose we wish to assign

S=1 if N=0 or 1 or 16 or 17,
S=2 if N=4 or 5 or 20 or 21
S=0 otherwise

This may be achieved by the following coding:

```
  10 PROCCK
  20 REPEAT
  30 READ N,S
  40 UNTIL CK=N OR N=-1
  50 PRINT S:RESTORE
  60 DATA 0,1,1,1,16,1,17,1,4,2,5,2,20,2,21,2,-1,0
1000 GOTO 10
```

LISTING 4 (to which listing 1 must be added)

Note that in the DATA statement the cell number alternates with the selector assigned to it. The list terminates with −1,0. The dummy negative value neatly ensures that when the list has been read thus far, all remaining cell numbers are given selector zero.

As a further example let us suppose that cells 50-53 and 66-69 are to be overlaid with a picture of a car while 58-61 and 74-77 are covered with a picture of a bus. A program may be developed by altering and adding to listing 4 along the following lines:

```
50 IF S=1 PRINT "Car"
 55 IF S=2 PRINT "Bus"
 57 RESTORE
 60 DATA 50,1,51,1,52,1,53,1,66,1,67,1,68,1,69,1,58,
              2,59,2,60,2,61,2,74,2,75,2,76,2,77,2,-1,0
```

In the most general case, if we wish to assign

S=1 if N=X1,X2, . . . , or XR,
S=2 if N=Y1,Y2, . . . , or YS,
....
S=W if N=Z1,Z2 . . . , or ZT,
S=0 otherwise

then the required DATA statement will be of the form

60 DATA X1,1,X2,1, . . . , XR,1,Y1,2,Y2,2, . . . , YS,2, . . . ,Z1,W,Z2,W, . . . , ZT,W−1,0

Special case: ASCII codes
For some applications we may wish to use all of the 128 cells as individual response areas. In particular the ASCII code (American Standard Code for Information Interchange) may be used to associate each cell with a character. To demonstrate this, simply replace line 20 in Listing 2 with

```
    20 PRINT CHR$(CK)
```

and use Figure 8. Note that each upper case letter is precisely two rows above its lower case equivalent.

Be warned! Strange things will happen if you touch the top two rows of cells. These cells correspond to control characters which can similarly wreck conventional programs unless appropriate childproofing action is taken.

ASCII codes

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC 1	DC 2	DC 3	DC 4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SPACE	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DELETE

Figure 8

4. ADAPTING EXISTING PROGRAMS

It should be emphasised that converted QWERTY software is unlikely to make the best use of CONCEPT hardware. Good CONCEPT Keyboard programs will be designed specifically as such, bearing in mind the objective and the intended flow of information between the user, the keyboard and the screen (see figure 2, chapter 1).

Nevertheless many readers will find they have programs which can profitably be converted. In any case it is worth understanding how it is done.

These are two ways of making conventional keyboard programs available for use with the CONCEPT Keyboard:

a. use a package designed for this purpose,

b. do-it-yourself.

a. Those who choose to use a package are now referred to the STARSET Pack. By following the instructions, the user can create a "machine code file". This resides in the computer's memory and allows the unaltered source program to accept input from either the CONCEPT Keyboard or the QWERTY Keyboard. STARSET also facilitates overlay design; a picture of the CONCEPT Keyboard is presented on the screen and the cursor control (or "arrow") keys may be used to assign meanings to the cells. There is no need to look up cell numbers and enter them as data.

b. There are four stages in converting a program for CONCEPT Keyboard use.

  i. Change the statements that handle input. e.g. a line like A$=GET$ may be changed to a procedure call PROCCK.

  ii. Alter any user-instructions so that they refer to CONCEPT Keyboards overlays. e.g. "Press any key to continue" may become "Press 'CONTINUE'" or "Press the blue square".

  iii. Make any adjustments to the logic of the program so that the values of the variables of PROCCK e.g. CK, S are passed to the variables of the existing programs.

  iv. Remember to add PROCCK to the program. It may be necessary to place an END statement before the listing of this procedure.

Then of course an overlay will need to be designed. A rough and ready overlay may be produced quickly using felt-tip pens. When your program merits quality art-work on the overlay, protect the original with transparent film. If making photocopies, ensure that the distances of the cells from the edges of the paper are the same as on the original.

To illustrate how the above steps may be carried out it is best to consider an example. The following lines may be part of a large program, much of which does not need changing.

```
10 PRINT "'Hit any key to continue.'"
20 A$—GET$
30 PRINT "PROGRAM CONTINUES"
40 REM A statement appears on the screen.
50 REM The user decides whether it is true.
60 PRINT "Press 'T' for TRUE, 'F' for FALSE."
70 R$=GET$
80 PRINT R$
90 REM Program continues
```

Listing 5

CONVERT THIS TO LISTING 6 BELOW

Following step (i.) above we replace lines 20 and 70 with PROCCK. For step (ii.) we change lines 10 and 60. See listing below. Step (iii.) is needed because the original program proceeds with the variable R$ set equal either to "T" or "F". When we return from PROCCK we have S=0 or S=1. We need lines 72 and 75 to link the old logic to the new. The final step (iv.) is to enter lines 2999 to 3080.

The overlay consists simply of a line down the middle with the word "true" displayed on the left and "false" on the right.

When all is done, line 80 verifies that both listings have the same effect on the rest of the program.

```
  10 PRINT "'Press CONCEPT to continue.'"
  20 PROCCK
  30 PRINT "PROGRAM CONTINUES"
  40 REM A statement appears on the screen
  50 REM The user decides whether it is true
  60 PRINT "Press TRUE or FALSE."
  70 PROCCK
  72 S=INT (CK MOD 16/8)
  75 IF S=0 THEN R$="T" ELSE R$="F"
  80 PRINT R$
  90 REM Program continues.
2999 END
3000 DEFPROCCK
3010 *FX 151,96,0
3020 *FX 151,98,0
```

```
3030 *FX 151,107,2
3040 A%=&96:X%=&6D
3050 REPEAT UNTIL USR(&FFF4) AND &100000
3060 X%=&60
3070 CK= (USR(&FFF4) AND &7F0000) DIV &10000
3080 ENDPROC
```

Listing 6

Note on 128 and 256 cell models

In general it is necessary to use a different line 3070 in PROCCK for each model (see listing 2). This is to ensure that CK values are in the right range. The above program is not interested in the cell numbers themselves, (only in their remainder after division by 16) so the two alternatives are not needed here. However, if it is ever required to make a 256 model behave as if it were a 128 model (e.g. to use software written for the 128) then the following lines should be added to PROCCK.

```
3071 IF(CK DIV 16)MOD 2=0 THEN CK=CK-(CK DIV 16)*8
      ELSE CK=CK-(CK DIV 16)*8-8
```

Cell Auto Repeat Routine

The following program will repeatedly print the value of the cell being touched until the user removes his finger. This type of input is useful for cursor controlling or any other variable which needs continual adjustment by the user during the program.

```
 10 CLS
 20 *FX151,96,0
 30 *FX151,98,0
 40 *FX151,107,2
 50 A%=&96:X%=&60
 60 CK1=(USR(&FFF4) AND &7F0000) DIV &10000
 70 CK2=(USR(&FFF4) AND &7F0000) DIV &10000
 80 CK3=(USR(&FFF4) AND &7F0000) DIV &10000
 90 IF(CK1=CK2) AND (CK1=CK3) THEN PRINT CK1
100 GOTO 60
```

5. PRINCIPLES OF CONCEPT KEYBOARD PROGRAM DESIGN

Psychologists analyse how we learn. We use our senses:

1. Seeing,
2. Hearing,
3. Feeling and Doing,
4. Tasting,
5. Smelling.

There is no doubt that different individuals make use of their senses to very different extents depending on their stage of development and the nature of what they are learning. Whichever is our dominant mode of learning, we will learn better if we are well motivated and if the method of learning suits our particular approach.

Looking down the above list of senses we will soon see that in the history of computer aided learning, more has been done to assist the visual learner than others. Only recently has speech output been widely available on computers and speech input is still far from common. Also rare are programs that say (e.g. in cookery or chemistry): go away and taste or smell something and report back to the computer.

General agreement has surely been reached on the undesirability of users sitting for long periods with eyes glued to the screen. Multi-sensory packages are to be encouraged (e.g. use of a light pen and speech synthesiser along with the CONCEPT Keyboard). Enlightened software producers are now supplying not just programs but a range of worksheets and support materials which encourage doing as well as looking.

Character formation by moving the finger on the overlay is an example of input by doing.

More generally the CONCEPT Keyboard presents opportunities for **visually stimulating input.** Whereas visual output is no longer confined to be just text, but is often colourful and exciting the CONCEPT Keyboard releases the learner from being constrained to make inputs in the form of strings of text. Well designed and attractive overlays should characterise good CONCEPT Keyboard software packages.

This is particularly important

i. When the subject matter is essentially pictorial or diagrammatic;

ii. for learners who think in pictures rather than words;

iii. when the user will find it much easier to touch a picture than spell out a word;

iv. when it is desirable to make the program independent of language, e.g. pre-reading or for adults of different nationalities.

We must always remember the point, made in the introduction, that the CONCEPT Keyboard overlay can give out information to the user as well as facilitating the receiving of input.

In everyday life a message like "no smoking", "ladies", "gents", "don't overtake" is often conveyed by means of a picture. (The term "icon" is gaining currency for such a symbol.) If the use of icons is desirable for any of the situations in (i.) to (iv.) above, then it is the task of the CONCEPT Keyboard programmer to create overlays accordingly.

What principles should be used in overlay design?

1. Needless to say, overlays should be clear, bold and unambiguous.

2. Plenty of space should be left between different response areas where possible.

3. Where appropriate, dead areas should be assigned which give no response and thus allow users somewhere to rest their hands.

4. A set of programs should have the same layout patterns. For example, whether a user is asked to choose from four colours, four animals or four fruits, the four response areas should appear in the same position on each overlay. In particular, control responses such as "PRESS TO GO ON" should always be in the same place. The adoption of this principle will assist users as they become used to familiar patterns. It will, of course, save programming effort.

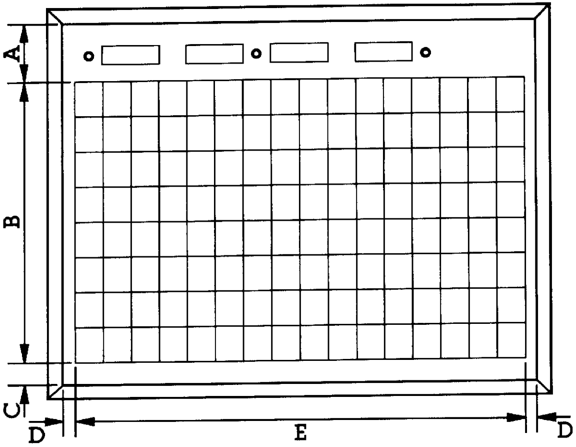
5. Token entry should be encouraged. That is to say the user is only required to press one response area per entry. This has many advantages. Programs use less memory; users restricted movement control will find it easier; and the software may be adapted easily for other input devices designed for special needs.

6. Where programs use several overlays it is important that the right overlay is in place at the right time. Apart from asking the user to check that the current overlay has the right name or number, the following method can be used. Leave three special response areas (in the same postion on each overlay) for the purpose of identification. Colour these three response areas red, yellow, green but in a different order on each overlay. Whenever a new overlay should be in place, the user can be asked to press red, then yellow, then green. There are six ways of pressing three different response areas in order, so this way the CONCEPT Keyboard can identify which of six overlays is in place. The use of one extra colour will provide for up to 24 overlays.

As the number of CONCEPT Keyboard users steadily increases, a volume of experience naturally emerges and, with this, standard conventions or codes of practice for CONCEPT Keyboard use. If you have any expertise to share you are invited to write to us.

AB European Marketing has started a newsletter to keep users informed of the latest developments.

6. OVERLAY DESIGN INFORMATION

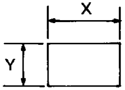


Overlay Dimensions

	A4	A3	A2
A	35	46	42
B	143	240	368
C	33	12	12
D	7	19	19
E	284	384	558

All dim. in m.m.

Cell Dimensions



	A4	A3	A3	A2	A2
	128	128	256	128	256
X	18	24	24	35	35
Y	18	30	15	46	23

7. TECHNICAL INFORMATION

INTRODUCTION

The CONCEPT Keyboard is an input selection array consisting of either a 16 × 8 matrix - 128 cell or 16 × 16 matrix - 256 cell.
The output from each cell is a unique code in the range hexadecimal 00 at the top left hand side to hexadecimal 7F (FF on the 256 cell) at the bottom right hand side.

Cell activitation uses the widely proved flexible membrane technology. The CONCEPT Keyboard is designed to allow the user by means of interchangeable overlays to select the character layout best suited for each application. The overlays can be written or drawn on paper for trial or other transient purposes, but can readily be produced on suitable films if required.

The touch area is a flat, wipe clean, scratch resistant, polycarbonate surface mounted in a low profile aluminium case.

Internal Mode of Operation
Each cell in the matrix is electronically inspected approximately 100 times a second. When the internal circuitry detects a cell being touched the inspection procedure then halts, the CONCEPT Keyboard will now idle for a period of 40 milliseconds whilst it confirms that a cell has been intentionally touched and not accidentally brushed – once the internal circuitry has satisfied itself that a cell has been positively touched, it then instructs the strobes to 'fire'. The strobes change state for a period of 5 milliseconds ±20%.

The standard unit (no serial option fitted), has a parallel output, that is, when the strobe 'fires' the data bits are all present at the same instant of time. This requires a separate wire for each data bit and the strobe.

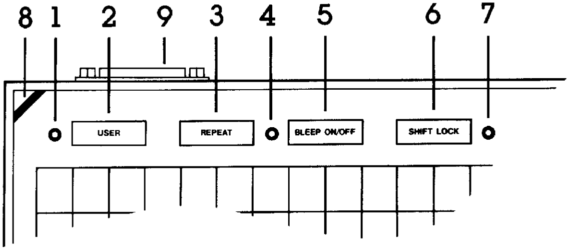
6. Shift Pad — Applicable to 128 cell CONCEPT Keyboards only - Toggles the state of the most significant data bit allowing access to Hexadecimal codes 80 to FF.

7. Shift LED — Indicates the state of the shift pad.
LED ON — Shift IN – MSB HIGH
LED OFF = Shift OUT – MSB LOW

8. Overlay Retaining Clips.

9. 25 way 'D' Plug with socket retaining mechanism.

CONTROLS AND INDICATORS



1. Power on LED — Indicates the unit is on.

2. User Pad — Touch to make switch - outside the keyboard matrix. May be used for RESET, BREAK, CLEAR, etc. Maximum resistive load 5 volts at 20 mA

3. Repeat Pad — When touched along with a matrix cell will cause the matrix cell code to be repeated.

4. Data Accept LED — Visual indication when a matrix cell is touched.

5. Bleep ON/OFF Pad — Toggle pad which will turn the audio bleep on or off.

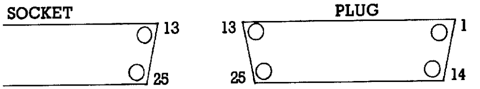
16
17
18
19
20
21
22
23
24
25

atrix cell is touched the strobe will change state for 5 milliseconds data bits are settled and stable prior to the strobes 'firing'. Both

Note 1 e buffered and each will drive 2 low power SHOTTKY TTL
Optional i oads.

BEFORE
DOUBLE
TIONS M data bits are fully buffered each will drive two LSTTL loads.

AND CONFIGURE INFORMATION

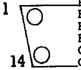


Viewed from solder side

Strobes
When a m
±20%. The
strobes a
(LSTTL) l

Data Bits
The eight

CONNECT



CONCEPT		ACORN BBC B (USER PORT)
User Pad		Not Used
User Pad		Not Used
Data bit 7 – Most Significant Bit (MSB) – Shift bit on 128 cell		20-PB7
Not Used		–
System Ground – 0Volts		19–(0 volts)
Data Bit 6		18–PB6
Data Bit 5		16–PB5
Data Bit 4		14–PB4
Data Bit 3		12–PB3
Data Bit 2		10–PB2
Data Bit 1		8–PB1
Data Bit 0 – Least Significant Bit (LSB)		6–PB0
Negative Strobe		2–CB1
Positive Strobe		Not Used
RS 422 –OUT A	Note 1 Serial Option	Not Used
RS 422 – OUT B	Note 1 Serial Option	Not Used
RS 232C/V24	Note 1 Serial Option	Not Used
Current Loop Positive	Note 1 Serial Option	Not Used
Current Loop Negative	Note 1 Serial Option	Not Used
RS423	Note 1 Serial Option	Not Used
Not Used		–
Not Used		–
Not Used		–
Unregulated input – 8 to 12 Volts D.C. only	Note 1	Not Used
+5 Volt supply ±0.25 Volts only.		1–(+5 volts)

Pin 1
2
3

tems – not fitted as standard.

4
5
6
7
8
9
10
11
12
13
14

APPLYING POWER TO THE KEYBOARD CHECK AND CHECK YOUR CONNECTIONS – INCORRECT CONNEC-
AY PERMANENTLY DAMAGE THE UNIT.

ACII CODE
American Standard Code For Information Interchange

Concept Cell Number	ASCII Char	Binary	Code LSB	Hexa-Decimal Code	Comments
0	NUL	000	0000	00	Control – @ Null
1	SOH	000	0001	01	Control – A Start of Heading
2	STX	000	0010	02	Control – B Start of Text
3	ETX	000	0011	03	Control – C End of Text
4	EOT	000	0100	04	Control – D End of Transmission
5	ENQ	000	0101	05	Control – E Enquiry
6	ACK	000	0110	06	Control – F Acknowledge
7	BEL	000	0111	07	Control – G Bell [Audible Signal]
8	BS	000	1000	08	Control – H Back Space
9	HT	000	1001	09	Control – I Horizontal Tab
10	LF	000	1010	0A	Control – J Line Feed
11	VT	000	1011	0B	Control – K Vertical Tab
12	FF	000	1100	0C	Control – L Form Feed
13	CR	000	1101	0D	Control – M Carriage Return
14	SO	000	1110	0E	Control – N Shift Out
15	SI	000	1111	0F	Control – O Shift In
16	DLE	001	0000	10	Control – P Data Link Escape
17	DCI	001	0001	11	Control – Q Device Control 1
18	DC2	001	0010	12	Control – R Device Control 2
19	DC3	001	0011	13	Control – S Device Control 3
20	DC4	001	0100	14	Control – T Device Control 4
21	NAK	001	0101	15	Control – U Negative Acknowledge
22	SYN	001	0110	16	Control – V Synchronous Idle
23	ETB	001	0111	17	Control – W End of Ts Block
24	CAN	001	1000	18	Control – X Cancel
25	EM	001	1001	19	Control – Y End of Medium
26	SUB	001	1010	1A	Control – Z Substitute
27	ESC	001	1011	1B	Control – [Escape
28	FS	001	1100	1C	Control – / File Separator
29	GS	001	1101	1D	Control –] Group Separator
30	RS	001	1110	1E	Control – ∧ Record Separator
31	US	001	1111	1F	Control – _ Unit Separator
32	SP	010	0000	20	Space
33	!	010	0001	21	
34	"	010	0010	22	
35	#	010	0011	23	
36	$	010	0100	24	
37	%	010	0101	25	
38	&	010	0110	26	
39	'	010	0111	27	
40	(	010	1000	28	
41	)	010	1001	29	
42	*	010	1010	2A	
43	+	010	1011	2B	
44	,	010	1100	2C	
45	-	010	1101	2D	
46	.	010	1110	2E	
47	/	010	1111	2F	

Concept Cell Number	ASCII Char	Binary	Code LSB Code	Hex Number	Concept Cell	ASCII Char	Binary Code	Code LSB	Hex
48	0	011	0000	30	96	'	110	0000	60
49	1	011	0001	31	97	a	110	0001	61
50	2	011	0010	32	98	b	110	0010	62
51	3	011	0011	33	99	c	110	0011	63
52	4	011	0100	34	100	d	100	0100	64
53	5	011	0101	35	101	e	110	0101	65
54	6	011	0110	36	102	f	110	0110	66
55	7	011	0111	37	103	g	110	0111	67
56	8	011	1000	38	104	h	110	1000	68
57	9	011	1001	39	105	i	110	1001	69
58	:	011	1010	3A	106	j	110	1010	6A
59	;	011	1011	3B	107	k	110	1011	6B
60	<	011	1100	3C	108	l	110	1100	6C
61	—	011	1101	3D	109	m	110	1101	6D
62	>	011	1110	3E	110	n	110	1110	6E
63	?	011	1111	3F	111	o	110	1111	6F
64	@	100	0000	40	112	p	111	0000	70
65	A	100	0001	41	113	q	111	0001	71
66	B	100	0010	42	114	r	111	0010	72
67	C	100	0011	43	115	s	111	0011	73
68	D	100	0100	44	116	t	111	0100	74
69	E	100	0101	45	117	u	111	0101	75
70	F	100	0110	46	118	v	111	0110	76
71	G	100	0111	47	119	w	111	0111	77
72	H	100	1000	48	120	x	111	1000	78
73	I	100	1001	49	121	y	111	1001	79
74	J	100	1010	4A	122	z	111	1010	7A
75	K	100	1011	4B	123	{	111	1011	7B
76	L	100	1100	4C	124	|	111	1100	7C
77	M	100	1101	4D	125	}	111	1101	7D
78	N	100	1110	4E	126	~	111	1110	7E
79	O	100	1111	4F	127	DEL	111	1111	7F
80	P	101	0000	50					
81	Q	101	0001	51					
82	R	101	0010	52					
83	S	101	0011	53					
84	T	101	0100	54					
85	U	101	0101	55					
86	V	101	0110	56					
87	W	101	0111	57					
88	X	101	1000	58					
89	Y	101	1001	59					
90	Z	101	1010	5A					
91	[	101	1011	5B					
92	/	101	1100	5C					
93	]	101	1101	5D					
94	∧	101	1110	5E					
95	_	101	1111	5F					