(continued from page 67)

position. See figure 2 for these bit values.

If the result of the And operation is 0, the key is being pressed. Bits 6 and 7 of all these key row values represent the conditions of the control and shift keys at the time of the scan — that is, if the value is less than 128D then shift is pressed; if less than 192D, CTRL is pressed.

For example, to check if A is pressed, after running the routine, do the following from Basic:

- Consider the value at # 83H — row 3.
- And value with 8D — bit position 3, column 3.
- If result 0, key is pressed.
- Check if control or shift are pressed as described, if necessary.

Figure 1. The keyboard matrix.

Row number	Bit position 0	1	2	3	4	5
9		3	—	G	Q	ESC
8		2	,	F	P	Z
7		1	;	E	O	Y
6		0	:	D	N	X
5	Lock	DEL	9	C	M	W
4		COPY	8	B	L	V
3	]	RET	7	A	K	U
2			6		J	T
1	[		5	/	I	S
0	SP		4	.	H	R
	0	1	2	3	4	5

Column number

Note: Read column number then row number for position code in Y after routine at # FE71H. E.g., A = 33D.

This approach to reading the keyboard opens a whole new range of key-column reading from column 0 which corresponds to bit 0, to column 5 which corresponds to bit 5. For example, if the A key is depressed register Y = 33, register X = 4 and Acc = 8 — i.e., column 3.

The following machine-code routine reads the keyboard and stores the Acc, X and Y in # 80E, # 81H and # 82H respectively, although only the Y register value is actually needed to find the depressed key.

```
20 71 FE   JSR# FE71
85 80      STA# 80
86 81      STX# 81
84 82      STY# 82
60         RTS
```

Note that the values are unaffected by shift and control, so to check for a shifted key, the value at B001D is less than 128D, and for a control key the value is less than 192.

If only an alphabet or number key, or for that matter any key with ASC11 code between # 30H and # 5AH, is being checked, the ASC11 code can be obtained by adding 14D to the value in the Y register. The one problem with the second routine is that only one key can be read at a time. This can be solved by using a small routine to make a bit map of the keyboard matrix.

The basic idea is to store the numbers 1 to 10D in sequence in the lower nybble of # B000D — port A of the 8255 PPIA — to drive each row of the matrix in turn and each time read the value at port B, which corresponds to the keyboard column, and store this in a buffer area to be looked up by the calling program.

Each of the 10 row values is stored in one byte, thus the whole keyboard requires only 10 bytes. My own preference is to use the memory between # 80H and # AF in zero page which is free.

Table 1.

# FDECH	— Scroll routine. Fill the screen with characters and try LINK# FDEC. LDY@0; JSR# FE16 produces half screen scroll.
# FE6BH	— Screen synchronisation routine.
# FD1AH	— CTRL-G routine.
# FD0BH	— Start screen, ACK routine.
# FD11H	— Page mode off, S1.
# FD0BH	— ESC routine.
# FF3FH	— RES routine.
# C9D8H	— Reset service routine.
# C9EYH	— Basic error routine.
# C000H	— Base of command table.
# D000H	— Base of FP command table.
# F000H	— Base of command table.
# F8BEH	— Base of command table.

The # represents hexadecimal; the @ immediate mode or Atom standard.

Screen print

N Higham,
Eccles, Manchester.

PET

THIS PROGRAM for all Pets — except old ROM — prints the contents of the screen on a Commodore 3022 or 4022 printer. When loaded, or appended to an existing program and called by Gosub 63000, the screen is dumped to the printer in the centre of the paper and surrounded by a box.

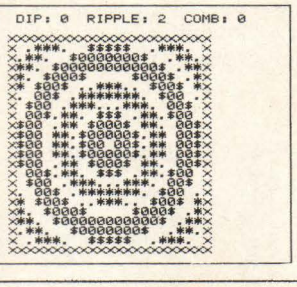
The routine is intelligent in that it executes a delay, the length of which depends linearly on the amount of reverse-field characters on the line just printed. This is to avoid the print head burning out if the screen contains a large amount of reverse field.

Line 63010 reduces the line spacing to give vertically contiguous graphics and line 63130 resets it. The printing can be halted at any time by pressing 'S' and restarted by pressing any key.

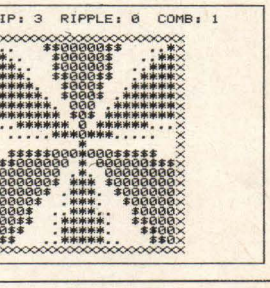
Line 63075 allows for a bug in the 4022 printer in that CHR$(254) prints a space instead of the required graphics character.

```
63000 REM ---- 'SCREENPRINT'  BY  N.J.HIGHAM ----
63005 REM  PRESS 'S' TO HALT PRINTING, ANY KEY TO RESTART
63007 CLR:P1=24:P2=36:REM FOR 4022
63008 REM FOR 3022 P1=18:P2=24
63010 OPEN6,4,6:PRINT#6,CHR$(P1):CLOSE6:PRINT"■"
63020 T=18:Z$="┌":FORI=1TO40:Z$=Z$+"─":NEXT:OPEN4,4:PRINT#4,TAB(T)Z$"┐"
63030 PRINT#4,TAB(T)"│ ";:DEFFND(X)=-(R>5)*(2000+400*X)
63040 E=33767:FORI=32768TOE:A=PEEK(I)
63050 GETA$:IFA$="S"THENPOKE158,0:WAIT158,1:GETA$:POKE158,0
63060 IFF=0ANDA>127THENF=1:PRINT#4,CHR$(T);:GOTO63075
63070 IFF=1ANDA<128THENF=0:PRINT#4,CHR$(146);
63075 IFA=254THENB$="■":GOTO63100
63080 B=-(A+64)*(A<64)-(A+128)*(63<AANDA<126)-(A+64)*(125<AANDA<128)
63090 B=B-(A-64)*(127<AANDA<191)-A*(190<AANDA<255)-191*(A=255):B$=CHR$(B)
63100 R=R+F:PRINT#4,B$;:X=X+1:IFX<40GOTO63120
63110 FORD=1TOFND(R):NEXT:R=0:X=0:F=0:PRINT#4,"■│";:IFI<ETHENPRINT#4,TAB(T)"│ ";
63120 NEXT:Z$="└":FORI=1TO40:Z$=Z$+"─":NEXT:PRINT#4,TAB(T)Z$"┘":PRINT#4:CLOSE4
63130 OPEN6,4,6:PRINT#6,CHR$(P2):CLOSE6
63200 OPEN4,4:PRINT"■RETURN■: P/FEED":PRINT"■SPACE■:  END
63210 A=PEEK(151):IFA=27THENPRINT#4
63220 IFA=6THENCLOSE4:POKE158,0:END
63230 GOTO63210
```

```
SIZE: 10  DIP: 0  RIPPLE: 2  COMB: 0
```



```
SIZE: 10  DIP: 3  RIPPLE: 0  COMB: 1
```



Storage technique

John Eade,
Stroud, Gloucestershire.

ZX-81

THE FREE-MEMORY routine by Paul Brittain, November 1981 page 67, is intended for use when writing programs. It must not hamper the program being written. If it is stored in a Rem statement in the first line, it will change all program addresses when finally deleted. In any case, there may be another machine-code routine which must occupy that position because of the absolute addresses within it.

There is, however, another way of storing machine-code routines which I have not seen mentioned elsewhere. This is simply to Poke it into any program line Rem statement and call it from the previous line using the NXTLIN variable — see ZX-81 manual page 178. Enter the two lines as follows:

```
9989 LET A = (PEEK 16425 + 256 × PEEK
     16426 + 5)
9990 REM XXXXXXXXXXXXXXXXXXXXX
     (that is 21 Xs)
9991 FOR N = A TO A + 21
9992 INPUT B
9993 POKE N,B
9994 PRINT PEEK N
9995 NEXT N
GOTO 9989
```

Enter: 175, 103, 111, 57, 237, 75, 28, 64, 237, 66, 229, 193, 253, 33, 0, 64, 62, 30, 237, 71, 201

Delete lines 9991, 9992, 9993, 9994, 9995

Edit line 9980 — delete LET A = and substitute PRINT USR

Do not forget to run the whole program if you can, so that the display file — if RAM is greater than 3.25K — and variables take the required space.

(continued on next page)