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.

| #FDEC#   | Scroll routine. Fill the screen with characters and try LINK # FDEC. LDYG0; JSR #FE16 produces half screen scroll. |
| #F6BH#   | Screen synchronisation routine. |
| #FD1AH#  | CTRL-G routine. |
| #FD0BH#  | Start screen, ACK routine. |
| #FD11H#  | Page mode off, S1. |
| #FDDBH#  | ESC routine. |
| #FFDH#   | RES routine. |
| #C9BH#   | Reset service routine. |
| #CEEY#   | Basic error routine. |
| #C00H#   | Base of command table. |
| #D00H#   | Base of FP command table. |
| #F09H#   | Base of command table. |
| #F8E#   | Base of command table. |

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

Screen print

N Higham, Eccles, Manchester.

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 center 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.

Storage technique

John Eade, Stroud, Gloucestershire.

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 place it into any program line Rem statement and call it from the previous line using the NXTLIN variable — see ZX-S1 manual page 178. Enter the two lines as follows:

```
9989 LET A = (#PEEK 16425 + 256 x PEEK 16426) + 51
9990 REM XXXXXXXXXXXXXXXXXXXX (that is 21 Xs)
```

Also:

```
9991 FOR N = A TO A + 21
9992 INPUT B
```

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.

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
88 81 STX # 81
84 82 STY # 82
60 RTS
```

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

If only an alphabet or number key, or for that any key with ASCII code between #30H and #5AH, is being checked, the ASCII code can be obtained by adding 14D to the value in the Y register. The one problem mentioned elsewhere is simply to Poke it into any program line Rem statement and call it from the previous line using the NXTLIN variable — see ZX-S1 manual page 178. Enter the two lines as follows:

```
9989 LET A = (#PEEK 16425 + 256 x PEEK 16426) + 51
9990 REM XXXXXXXXXXXXXXXXXXXX (that is 21 Xs)
```

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.