Towards a VZ-Epson printer patch

Part 1

Larry Taylor

Fed up with your clackerty old printer and long for an upgrade to one of the popular Epson or Epson-type dot matrix printers? Compatibility with the VZ has always been a problem – until now.

FED UP with your clackerty GP-100, and its less than perfect print quality? Do you long to upgrade, but know that whatever you choose, it won't be totally friendly towards your VZ?

Are you the owner of an Epson-type printer, but suffer frustration, as I did, at its lack of compatability? If so, then take heart, there is hope. The answer is a *printer patch*, that is, a program specifically written to take the place of the existing ROM routines. In this case, the aim is to make the VZ fully compatible with Epson-type printers. Recently, after many hours spent reading and experimenting, I succeeded in producing just such a program.

Having first decided to take the plunge and purchase a VZ computer, I developed a very great need, some short time later, to be able to obtain a printout of my programming efforts. On close examination of available finances, I was left with a choice between the Seikosha GP-100, a slow, noisy machine featuring an unattractive print style, and the BMC BX-80, a noticeably quieter, faster printer, possessing several attractive fonts.

Although a seemingly easy decision, I was immediately faced with a dilemma. The former, whilst initially unattractive, especially so to anyone with sensitive hearing, had two very desirable features: namely, the ability to print the VZ's inverse and graphics characters, in addition to providing, via the COPY command, a dump of the HI-RES screen. These two factors very nearly persuaded me to choose the GP-100, but, after much deliberation, I opted for the superior print quality of the BX-80. In so doing, I resigned myself to having to go without the former's obvious advantages.

No one had at this stage even remotely hinted that I could have the best of both worlds by means of a software patch. Hindered by a lack of information and minimal understanding of computer and printer operations, I perservered with the rather primitive approach of removing all inverse and graphics characters from programs before doing a printout.

A start

Desperate to overcome this huge waste of time, I first began to deal with the problem of printing graphics characters. I realised that my printer was capable of dot graphics and that it should be able, whilst in this mode, to reproduce the shapes I desired. My early efforts, however, ended in frustration as the VZ steadfastly refused to interpret my data correctly. Only when I discovered that I could send the data directly out the ports, thus bypassing the VZ's printer driver routine, did I achieve any success.

Listing 1 gives an example of how this was accomplished. By referring to the table below, you may change the graphics block data in the listing to enable any of the other graphics charactrs to be printed. Later it will become clearer how the data to print each block was calculated.

GRAPHIC BLOCK DATA

	HEXIDECIMAL	DECIMAL
128	00 , 00	0 , 0
129	0F , 00	15 , 0
130	00 , 0F	0 , 15
131	0F , 0F	15 , 15
132	F0 , 00	240 , 0
133	FF , 00	255 , 0
134	F0 , 0F	240 , 15
135	FF , 0F	255 , 15
136	00 , F0	0 ,240
137	0F , F0	15 ,240
138	00 , FF	0 ,255
139	0F , FF	15 ,255
140	F0 , F0	240 ,240
141	FF , F0	255 ,240
142	F0 , FF	240 ,255
143	FF , FF	255 ,255

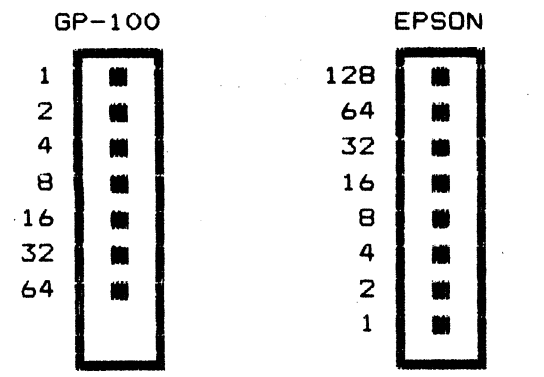
Being an avid user of Steve Olney's Extended Basic, I used my new-found knowledge to write an assembly routine, which linked into the listing routine of his program. It simply checked for graphics and inverse characters. Graphics characters were printed and inverse ones changed to non-inverse. Useful, but not totally satisfactory. On the way I had independently developed my own table of data (above), to print the graphics blocks, only to later discover that there exists in the VZ's ROM a set of data for graphics characters and another for inverse.

The graphics table occupies addresses from 02AFH to 02CEH, whilst the inverse data commences at 3B94H and ends at 3CD3H. The graphics shapes are stored in two-byte form and the inverse characters in five-byte blocks. Their existence makes it a simple enough matter to expand on the program in Listing 1 and print the graphics blocks using the ROM data instead of our own, as in Listing 2. The same may be done with the inverse characters and Listing 3 shows how this is accomplished. Unfortunately, you will notice that the resultant characters, when printed, are in fact upside down. To understand why this occurs, it is necessary to offer a brief explanation of the differences between the code values used to control firing of the pins in the printheads of Epson-type printers, and those of the GP-100 family.

The Epson-type printer

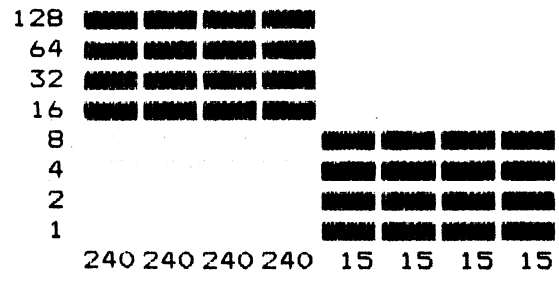
Printers of the Epson-type have eight addressable pins, while the GP-100 has the equivalent of seven pins only. In addition, the value 1, which fires the bottom pin on an Epson printer, actually triggers the top pin on the GP-100. The diagram below illustrates the differences.

COMPARISON OF PIN CODE VALUES



To calculate the code which is required to produce a particular dot pattern we simply have to add up the values of the corresponding pins. The representation of the graphics block, CHR$(137), can be used to demonstrate how this is done. You may recall that the data values used in Listing 1 to reproduce this particular character were 240 and 15. Notice how these codes correspond to the totals at the base of each column in the diagram. If we examine the first column on the left, we can see that only the top four pins have been fired. By totalling vertically the values assigned to those pins, we arrive at the sum of 240. The same procedure is used to determine the Epson compatible code for each of the remaining columns.

GRAPHICS BLOCK 137



It can be done

Nevertheless, data which has been prepared primarily for the GP-100, as is the case with the ROM tables, will produce inverted images if sent to an Epson printer. It is necessary, threfore, to convert the data before it can be used. Adding Listing 4 to Listing 3 will produce the desired result. I wouldn't however, advise any of you to hold your breath whilst waiting for the data to be printed. Hence, I have provided Listing 5, an assembler program, which effects the same result, only much more swiftly.

Having now managed to make the characters appear in their more conventional form, a closer examination of them will reveal numerous inaccuracies. Some, such as the 3 and 5, are more noticeable than others, but no less than a dozen of the characters are flawed. After progressing so far, this is a disappointing development but one which will prove, later, to be not insurmountable. In the interim, we need to explore further how we might utilise our somewhat imperfect data.

Fortunately, the designers of the ROM foresaw the possibility that potential users may want to use a different printer. As a result, a vector has been used to point to the location of the printer driver. All output to the printer is directed via a driver routine, which, among other things, checks for control codes and keeps track of line feeds. In the VZ, a block of the communications area of RAM from 7825H to 782CH has been set aside for printer operations, allowing temporary storage of values such as the number of lines printed. Of greatest interest to us is the contents of 7826H-7827H. This is the start of the driver routine, and the cause of our problems, because it is geared to expect that owners of VZeds will be using GP-100 type printers. However, since the previous address lies in RAM, it is possible to insert a pointer to our own driver routine at this location. Once accomplished, all future LPRINT and LLIST commands will be directed, ultimately, to our own printer routine.

We have now proceeded part way to installing a valuable routine for owners of Epson-type printers, but we are still unable to make use of the COPY command. The primary advantage of which is that it allows a dump of the HI-RES screen to be made to the printer. Implementing this very desirable feature will prove to be somewhat more challenging.

LISTING 1 : PRINT A SINGLE GRAPHICS BLOCK

```
100 REM ######################################
101 REM # PUT PRINTER IN GRAPHICS MODE       #
102 REM ######################################
110 LPRINTCHR$(27);CHR$(75);
120 FOR T=1 TO 2
130   READ D:GOSUB 510
140 NEXT T
200 REM ######################################
205 REM # READ EACH DATA VALUE IN TURN       #
210 REM # AND THEN PRINT IT FOUR TIMES       #
215 REM ######################################
220 FOR N%=1 TO 2
230   READ D
240   GOSUB 510:GOSUB 510
250   GOSUB 510:GOSUB 510
400 NEXT N%
410 LPRINT:END
500 REM ######################################
501 REM # OUTPUT TO PRINTER VIA THE PORTS    #
502 REM ######################################
510 IF INP(0)<>254 THEN GOTO510
520 OUT 13,D:OUT 14,D
530 RETURN
540 REM ######################################
545 REM # NUMBER OF BYTES TO BE PRINTED      #
550 REM # IN LOW BYTE, HIGH BYTE FORM        #
555 REM ######################################
560 DATA 8,0
565 REM ######################################
570 REM # GRAPHIC BLOCK DATA                 #
575 REM ######################################
580 DATA 240,15
```

LISTING 2 : PRINT THE ROM GRAPHICS BLOCKS

```
100 REM ######################################
101 REM # PUT PRINTER IN GRAPHICS MODE       #
102 REM ######################################
110 LPRINTCHR$(27);CHR$(75);
120 FOR T=1 TO 2
130   READ D:GOSUB 510
140 NEXT T
150 REM ######################################
151 REM # LOCATION GRAPHICS TABLE 02CEH      #
152 REM ######################################
```

```
160 M=687
200 REM ######################################
205 REM # READ DATA FOR GRAPHICS BLOCKS      #
210 REM # AND PRINT EACH VALUE 4 TIMES       #
215 REM ######################################
220 FOR N%=1 TO 32
230   D=PEEK(M)-128 :M=M+1
240   GOSUB 510:GOSUB 510
250   GOSUB 510:GOSUB 510
260 REM ######################################
265 REM # THIS LINE SEPARATES CHARACTERS     #
270 REM # FROM EACH OTHER BY A DOT WIDTH     #
275 REM ######################################
280  IF N%/2 = INT(N%/2) THEN D=0 :GOSUB 510
400 NEXT N%
410 LPRINT:END
500 REM ######################################
501 REM # OUTPUT TO PRINTER VIA PORTS        #
502 REM ######################################
510 IF INP(0)<>254 THEN GOTO510
520 OUT 13,D:OUT 14,D
530 RETURN
540 REM ######################################
545 REM # NUMBER OF BYTES TO BE PRINTED      #
550 REM # IN LOW BYTE, HIGH BYTE FORM        #
555 REM ######################################
560 DATA 144,0
```

LISTING 3 : PRINT THE ROM INVERSE CHARACTERS

```
100 REM ######################################
101 REM # PUT PRINTER IN GRAPHICS MODE       #
102 REM ######################################
110 LPRINTCHR$(27);CHR$(75);
120 FOR T=1 TO 2
130   READ D:GOSUB 510
140 NEXT T
150 REM ######################################
151 REM # LOCATION OF INVERSE TABLE 3B94H    #
152 REM ######################################
160 M=15252
200 REM ######################################
201 REM # NUMBER OF INVERSE CHARACTERS       #
202 REM ######################################
210 FOR N%=1 TO 64
220       D=255:GOSUB 510
230 REM ######################################
231 REM # NUMBER OF BYTES PER CHARACTER      #
232 REM ######################################
240   FOR R%=1 TO 5
250       D=PEEK(M):M=M+1
339 REM ######################################
340 REM # PRINT ONE COLUMN                   #
341 REM ######################################
350       GOSUB 510
360   NEXT
370     D=255:GOSUB 510
400 NEXT N%
410 LPRINT:END
500 REM ######################################
501 REM # OUTPUT TO PRINTER VIA THE PORTS    #
502 REM ######################################
510 IF INP(0)<>254 THEN GOTO510
520 OUT 13,D:OUT 14,D
530 RETURN
535 REM ######################################
540 REM # NUMBER OF BYTES TO BE PRINTED      #
550 REM # IN LOW BYTE, HIGH BYTE FORM        #
555 REM ######################################
560 DATA 192,1
```

LISTING 4 : CONVERT THE DATA FOR THE EPSON PRINTER

```
260 REM ######################################
261 REM # CHANGE CODE FROM GP-100 TO EPSON   #
262 REM ######################################
270       IF D=189 OR D=255 THEN 320
280       V=0:E=0
290      FOR F%=7 TO 0 STEP -1
300       P=2^F%:IF D<P THEN 320
310       E=E+2^V:D=D-P
320       V=V+1
330      NEXT:D=E
```

LISTING 5 : PRINT THE ROM INVERSE CHARACTERS

```
0001 ;######################
0002 ;#  PUT PRINTER IN    #
0003 ;#  GRAPHICS  MODE    #
0004 ;######################
0005       LD    A,27
0006       CALL  3ABAH
0007       LD    A,75
0008       CALL  3ABAH
0009       LD    A,192
0010       CALL  3ABAH
0011       LD    A,1
0012       CALL  3ABAH
0013 ;######################
0014 ;# LOCATION OF THE    #
0015 ;#  INVERSE TABLE     #
0016 ;######################
0017       LD    HL,3B94H
0018 ;######################
0019 ;# NUMBER OF INVERSE# #
0020 ;#     CHARACTERS     #
0021 ;######################
0022       LD    B,64
0023 NEXT  PUSH  BC
0024       LD    A,255
0025       CALL  3ABAH
0026 ;######################
0027 ;# NUMBER OF BYTES    #
0028 ;#  PER CHARACTER     #
0029 ;######################
0030       LD    B,5
0031 PRNT  LD    A,(HL)
0032       CALL  CVRT
0033       CALL  3ABAH
0034       INC   HL
0035       DJNZ  PRNT
0036       LD    A,255
0037       CALL  3ABAH
0038       POP   BC
0039       DJNZ  NEXT
0040       RET
0041 ;######################
0042 ;# CHANGE CODE FROM   #
0043 ;# GP-100 TO EPSON    #
0044 ;######################
0045 CVRT  PUSH  BC
0046       LD    B,8
0047 ROTA  RR    A
0048       RL    C
0049       DJNZ  ROTA
0050       LD    A,C
0051       POP   BC
0052       RET
```

— from page 30

chromium to resist corrosion) and a solid "beta alumina" electrolyte separates anode and cathode. The cell is sealed and filled with argon.

During discharge, sodium ions pass through the electrolyte from anode to cathode, forming sodium sulphide at the cathode, the reaction generating the current. Recharging is achieved as with other storage batteries, by passing a current through it in reverse. One problem, though. These cells will only deliver power when operated above 270 degrees Celsius. They have an operating temperature ceiling of 410 degrees C. They must be heated to 'start up' and to maintain them within the operating temperature range, they have to be fully charged and then at least 80% discharged each day. If unused for nine hours, temperature falls below the 270 degrees C.

Sodium-sulphur cells exhibit a terminal voltage of around 2 V and may last some five years or 6000 charge-discharge cycles, which betters the typical lead-acid battery life cycle. In addition, its terminal voltage remains constant until it reaches about 70% of its discharge capacity before tapering off.

Suggested application encompass commercial vehicles such as delivery vans and buses, and military submarines. Satellite applications are also suggested as sodium-sulphur cells are only 20% of the weight of equivalent NiCad batteries of the same Ah output.