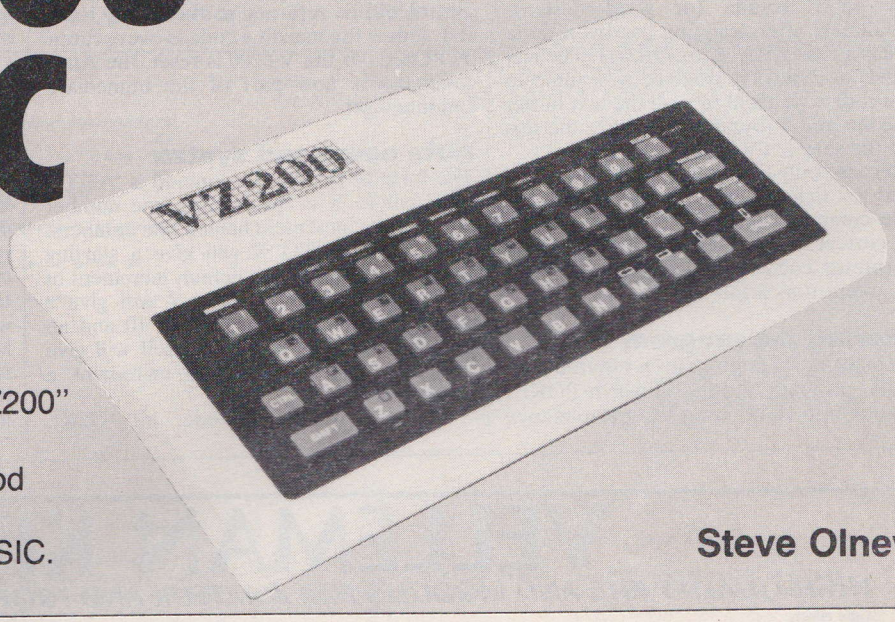
EXTENDING VZ 200 BASIC



Following on from a previous article ("More functions for the VZ200" — ETI March 1984) this article outlines a method of adding commands to the standard VZ200 BASIC.

Steve Olney

THE PREVIOUS article showed how to unlock several 'hidden' functions contained in the VZ200 BASIC ROM by entering the commands indirectly via a BASIC program itself. This approach meant that it was necessary to run the BASIC program each time the function was needed. This is very inconvenient and, as was hinted at in the previous article, a more elegant (and more convenient) approach would be to have the added functions accessed as if they were part of the original command set.

This article gives a method by which this can be done and gives a practical example by making the AUTO command part of the legal VZ200 BASIC command set.

The machine code necessary to achieve this is quite short because, as indicated in the previous article, the code which does the bulk of the work is already resident in the VZ200 BASIC ROM. It is only necessary to get the BASIC interpreter to recognise the auto line-numbering command (AUTO X, Y) as legal and then jump to the relevant code in ROM.

The method outlined here only applies to adding commands to the 'immediate execution mode'. (i.e: typing in commands without line numbers). It does not deal with commands that are to be used within programs.

How it works

Those who are only interested in the end result of adding the AUTO command to the legal commands can skip this section and go straight to the section dealing with entering the program. Those who are interested in how it works — read on!

The reason why it is possible to add commands to the standard VZ200 BASIC command set (thereby extending it) is that, in common with some other BASICs, at various points in the machine code in ROM, calls are made to locations in RAM. This makes it feasible to modify and/or extend the code at a later date. A common example is where a disk system is added later. An extended or enhanced BASIC can be implemented by downloading extra code off disk to the relevant called location. If all the code was executed in ROM then this could not be done.

In a non-disk system (such as the present VZ200) these called locations are usually initialised to '0C9H' (H means hex address of location), which is Z-80 machine code for Ret. So normally, when these RAM locations are jumped to via 'calls' from the BASIC ROM, execution returns immediately to the BASIC ROM via the 'Ret'.

Now, because the Ret's are in RAM, it is possible to change the Ret to a jump to extra code which will be executed before control is returned back to the BASIC ROM.

In the VZ200, all the calls from the BASIC ROM to RAM are to locations between 7952H and 79E2H. One of these exits will be used to add Auto X,Y to the legal command set.

The BASIC interpreter

Leaving the ROM exits for the moment, consider what happens when an 'immediate execution' command is entered. While the text is being typed in, the character codes for each key-press are being entered into a text buffer at around 79E8H. When Return is hit, the interpreter looks at what has been entered into the buffer. Scanning from left to right, it looks for 'reserved words' (words set aside for commands e.g: Print, List etc.). The BASIC ROM contains a list of these reserved words beginning at 1650H and ending at 1820H. This can be revealed by an ASCII dump of this block of memory (the first letter of each reserved word has 80H added to ASCII code which will result in garbage for that letter.)

The interpreter scans the text trying to find one or more of these reserved words. when one of these is found the reserved word text is replaced by a single byte or ▶

'token' (80H to 0FBH). The token is the offset into the list where the reserved word is located and is used as an index into another table which contains the address of the machine code for that command.

If the text cannot be resolved into reserved words or text which belongs to the reserved words, then a Syntax error message is generated. The trick is to intercept control of the interpreter just after the reserved list has been scanned and add code to re-scan the text to see if it contains the new command Auto X,Y.

By good fortune (or good design), immediately after scanning has been done there is a call to RAM (to 79B2H). The Ret (0C9H) at 79B2H is changed to a jump to extra code which will re-scan the text buffer for Auto and if found, will replace the text with the relevant token.

Because only the reserved word list is disabled (by deleting Auto from it), once the Auto command text has been replaced by the correct token (0B7H), the following interpreter code will recognise the token and accept it as legal.

Entering the program

The machine code program is entered via a BASIC program (Listing 1) which POKEs the code into RAM from Data statements. The BASIC program locates the machine code to high memory after resetting the BASIC top-of-memory pointer to below where the code will be POKEd. By this, the machine code program is located out of the way of any BASIC program to be entered later. This action is independent of memory size.

The machine code listing is shown for reference only. All that is necessary is to enter the BASIC Program, save it on tape, and from then on just run it before you start entering your BASIC program. If all is well, control will be returned to the Ready level and, unless the machine code is overwritten by POKEs or the VZ200 is reset, the Auto command is now part of the immediate command set.

Auto command syntax

The form of the Auto command is 'AUTO X,Y' where X is the starting line number and Y is the increment beteen line numbers.

Entering AUTO X will give a starting line number of X and a default increment of 10, while entering AUTO, Y will give a default starting line number of 10 and an increment of Y. AUTO by itself will give both the line number and increment a default of 10.

To exit the Auto mode, hit 'CTRL BREAK'. Entering the Auto mode with line numbers of statements already entered can be a useful single step checking and editing feature (see previous article).

Adding other commands

This method can be used for 'unlocking' other commands 'hidden' in the VZ200 BASIC ROM. As shown in the previous article, the commands TRON and TROFF are also accessible. In the time since that article was submitted it has been found that the code for a delete command (DEL X-Y), with the same syntax as the LIST command, is also present in the VZ200 BASIC ROM.

The listing for a BASIC program that 'unlocks' the 'hidden' code for the AUTO, TRON, TROFF and DEL commands is available from the author. It is of the same form as the program described here.

What next?

The above four extra commands have proved to be very useful and have resulted in significant time-savings in writing BASIC code. Other useful commands would be REN (line re-numbering), MERGE (merging small sub-programs on tape into one program — difficult, because it appears that the VZ200 CLOAD always loads a BASIC program to the location in

memory from which it was CSAVEd), DH and HD (allows decimal to hexa-decimal conversion, and vice-versa). These would be much more difficult to implement as there is no code present in the VZ200 BASIC ROM, so they will have to be written from scratch.

Cautions

Firstly, as this program uses code in the Version 2.0 BASIC ROM, users with other versions (if any) will have to check to see if the program works with their version.

Secondly, you may have already found that during normal program entry, occasionally the cursor will skip a line after you hit Return. This is of no real consequence — until now. Unfortunately the auto line-numbering code doesn't like this and responds by displaying the next line number as it should, but then positions the cursor at the beginning of the next line. Any BASIC statements or text entered on that line will be lost.

Each time Return is hit for a new line number, check to see that the cursor is on the same line as the new line number. If it isn't, hit Return again. This will skip to the next line number. Do this until the cursor is positioned on the same line as the new line number, then it is OK to enter statements. Unless you are fussy the missed line numbers should not be a problem. Of course, you can exit the auto mode (CTRL BREAK) and restart so as not to miss a line number.

Machine Code Source Listing

```
;  ****************************************************
;  **                                                **
;  **  BASIC AUTO LINE-NUMBERING UTILITY FOR THE VZ200  **
;  **          COPYRIGHT (C) 1984 BY STEVE OLNEY       **
;  **         200 Terrace Rd. North Richmond 2754       **
;  **                                                **
;  ****************************************************
;
;  MACHINE CODE PROGRAM (POKE'd from the Basic program)
;
;  Actual origin depends on the size of the memory in the
;  VZ200 used.
;
START   ORG     0000H
;
;  Save registers to be used
;
REGSAV  PUSH    AF
        PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    IX
;
;  This code scans the text buffer for the 'AUTO' command.
;
AUTOSC  LD      B,03            ;Number of bytes to scan
        LD      IX,AUTTXT       ;Pointer to 'AUTO' text table
SCAN1   INC     HL              ;Adjust to next byte in buffer
        LD      A,(IX+00)       ;Get first byte of table
        CP      (HL)            ;Compare with byte in buffer
        JR      NZ,EXIT-$       ;If not equal then exit
        INC     IX              ;Move to next byte in table
        DJNZ    SCAN1-$         ;Loop back until 3 bytes done
;
;  Execution drops through to here if all 3 bytes match.
;  The 'AUTO' text is replaced with its token (0B7hex) and
;  the rest of the text (operands if any) is closed up behind
;  the token.
;
FNDAUT  PUSH    HL              ;Save end of 'AUTO' in buffer
        DEC     HL              ;Move back to beginning of
        DEC     HL              ;'AUTO' text in buffer
        LD      (HL),0B7H       ;Replace first byte with token
        LD      BC,0000H        ;for 'AUTO'
        POP     DE              ;End of 'AUTO' text in buffer
        EX      DE,HL           ;HL=end of 'AUTO',DE=token
        INC     DE              ;Adjust DE to next byte
SKIP    INC     HL              ;Adjust HL to next byte
NEXT    LD      A,(HL)          ;Get byte from text buffer
        OR      A               ;Is it zero ?
        JR      Z,ENDLIN-$      ;If zero then end of line
        CP      20H             ;Is it a space ?
        JR      Z,SKIP-$        ;Yes ? Then skip to next byte
        LDI                     ;No ? Then transfer byte
        JR      NEXT-$          ;forward and continue
;  Line in text buffer must terminate with three zero bytes
;  and register 'C' must contain the new line length
;
ENDLIN  LD      (DE),A          ;Terminate line with three
        INC     DE              ;zero bytes.
        LD      (DE),A
        INC     DE
        LD      (DE),A
        LD      A,C             ;New text byte count-1, add 6
        CPL                     ;to complemented negative no.
        ADD     A,06            ;to adjust to line length+1
        LD      (LINLEN),A      ;and store it
;
;  Restore registers
;
RESREG  POP     IX
        POP     HL
        POP     DE
        POP     BC              ;Do this just to empty stack
        POP     AF
        LD      BC,(LINLEN)     ;Restore BC with new line
        LD      B,00H           ;length on return to ROM
        RET
;
;  Auto command not found so we return to ROM without
;  altering text or 'C' register.
;
EXIT    POP     IX
        POP     HL
        POP     DE
        POP     BC
        POP     AF
        RET
;  Text table for the 'AUTO' command. Because the 'TO' in
;  'AUTO' is a reserved word, it will have already been token-
;  ised. The token for 'TO' is 0BDH.
;
AUTTXT  DEFB    'A'             ;ASCII "A"
        DEFB    'U'             ;ASCII "U"
        DEFB    0BDH            ;Token for "TO"
LINLEN  DEFS    2
```

LISTING 1

```
0 REM     ******************************************************
10 '      ** USE THE SHORT FORM "'" FOR THE REST OF THE "REM"S **
20 '      **                                                  **
30 '      **  BASIC AUTO LINE-NUMBERING UTILITY FOR THE VZ200  **
40 '      **          COPYRIGHT (C) 1984 BY STEVE OLNEY       **
50 '      **         200 TERRACE RD. NORTH RICHMOND 2754      **
60 '      **   "AUTOBAS" TAPE FILE #17-B  9/5/84 VERSION 1.2  **
70 '      **                                                  **
80 '      ******************************************************
90 '
100 RB=100:TM=(PEEK(30897)+PEEK(30898)*256)-RB:'GET TOP OF
110 MS=INT(TM/256):LS=TM-MS*256:'               MEMORY AND MOVE
120 POKE30897,LS:POKE30898,MS:'                 DOWN 100 BYTES
200 CLEAR50:'                          RESET BASIC STACK PTR
230 TM=(PEEK(30897)+PEEK(30898)*256):'  NEW TOP OF MEMORY
235 M1=INT((TM+1)/256):L1=TM+1-M1*256:' NEXT LOC'N ABOVE T.O.M.
240 ST=TM:IFST>32767THENST=ST-65536:'   START OF M/C PROG.-1
250 FORI=1TO82:'   LOAD 82 BYTES OF MACHINE CODE INTO RESERVED
255   READD:'      AREA ABOVE BASIC TOP OF MEMORY
260   POKEST+I,D
265   CS=CS+D:'    UPDATE CHECKSUM TOTAL
270 NEXTI
275 IFCS<>9861THENPRINT"- ERROR IN DATA ENTRY -":END:' CHECKSUM
280 FORI=1TO3:READLB,OS:TS=TM+OS:'  BECAUSE PROGRAM IS RELOCATED
290   MT=INT(TS/256):LT=TS-MT*256:' ABSOLUTE LOCATIONS NEED TO
300   POKEST+LB,LT:POKEST+LB+1,MT:' LOADED
310 NEXTI
365 ' ALTER "RET" AT 79B2 HEX TO JUMP TO START OF MACHINE CODE
370 POKE31155,L1:POKE31156,M1:POKE31154,195
380 POKE30862,249:POKE30863,0:' LOAD CALL TO "READY" ROUTINE
390 X=USR(0):'                  AND GO TO IT
395 ' DECIMAL EQUIVALENT OF MACHINE CODE PROGRAM INSTRUCTIONS
400 DATA245,197,213,229,221,229,6,3,221,33,79,0,35,221,126,0
410 DATA190,32,53,221,35,16,245,229,43,43,54,183,1,0,0,209,235
420 DATA19,35,126,183,40,8,254,32,40,247,237,160,24,244,18,19
430 DATA18,19,18,121,47,198,6,50,82,0,221,225,225,209,193
440 DATA241,237,75,82,0,6,0,201,221,225,225,209,193,241,201
450 DATA65,85,189
460 DATA11,80,58,83,68,83
```