Australian Personal Computer

ISSN 0725—4115 NZ $3.50
REGISTERED BY AUSTRALIA POST PUBLICATIONS VBP 3691

JUNE 1986 $3.50

REE!
ELBOURNE PC
SHOW CATALOGUE
WITH THIS ISSUE

AUSTRALIA'S TOP SELLING COMPUTER MAGAZINE

IBM'S LAPHELD — IT'S OFFICIAL
World exclusive: PC Convertible tested

Amstrad LOKVIEW

by Paul Jenkins

This short machine code routine displays the Caps Lock and Shift Lock state at the bottom of the screen. It is interrupt-driven and so works continually once it has been set up.

To set up the machine code program, use the Basic loader which checks for typing errors in the machine code data. Once the data has been entered correctly and POKED into memory, save it with SAVE"lokview.bin",b,&xxxx,&F0 where &xxxx is the start address of the machine code. It is important to save the machine code before it is called. As the program is relocatable, it can be installed anywhere in memory and run from there. The Binary Loader program loads the program just above HIMEM, and can be merged into any program where the status is to be displayed.

The assembly listing shows how the program works. Firstly, screen windows are set up; then an interrupt routine is added to the Frame Flyback event. The routine then returns to Basic, but the interrupt routine will be activated every fiftieth of a second. After a Mode command, the windows will be changed; they can be reset by WINDOW 1,80,1,24:WINDOW #7,1,80,25,25, or you can CALL the address at which LOKVIEW is stored.

```
10 ' ************************
20 ' *       LOKVIEW        *
30 ' *    Paul S Jenkins    *
40 ' *        1986          *
50 ' ************************
60 '
70 ' Basic Loader
80 '
90 '
100 address=HIMEM-&F0
110 entry=address
120 MEMORY address-1
130 RESTORE 290
140 FOR lin=290 TO 580 STEP 10
150   sum=0
160   FOR column=0 TO 7
170     READ byte$
180     value=VAL("&"+byte$)
190     POKE address,value
200     sum=sum+value
210     address=address+1
220   NEXT column
230   READ checksum
240   IF checksum<>sum THEN PRINT"Checksum
      error at line";lin:STOP
250 NEXT lin
260 PRINT"'LOKVIEW' machine-code now loaded"
270 PRINT"at address &";HEX$(entry,4)
280 END
290 DATA  21,E1,E9,22,30,00,F7,EB,1055
300 DATA  21,21,00,19,4E,23,46,79,395
310 DATA  B0,28,2F,E5,60,69,19,E5,947
320 DATA  4E,23,46,60,69,19,44,4D,554
330 DATA  E1,71,23,70,E1,23,18,E4,997
340 DATA  61,00,67,00,6D,00,74,00,425
350 DATA  7E,00,8E,00,91,00,AE,00,587
360 DATA  B2,00,B8,00,BD,00,00,00,551
370 DATA  00,00,3E,07,CD,B4,BB,11,658
380 DATA  18,00,21,18,4F,CD,66,BB,654
390 DATA  CD,6C,BB,AF,CD,B4,BB,11,1264
400 DATA  00,00,21,17,4F,CD,66,BB,629
410 DATA  CD,6C,BB,00,11,40,18,21,638
420 DATA  F9,FF,72,23,73,21,5C,00,893
430 DATA  3E,C9,77,21,C9,00,06,81,751
440 DATA  0E,00,11,79,00,C3,D7,BC,750
450 DATA  F5,C5,D5,E5,11,D2,00,CD,1316
460 DATA  21,BB,EB,7A,BE,20,05,7B,927
470 DATA  23,BE,28,03,CD,9F,00,11,649
480 DATA  D2,00,CD,21,BB,EB,72,23,1019
490 DATA  73,E1,D1,C1,F1,C9,3E,07,1253
500 DATA  CD,B4,BB,CD,6C,BB,AF,CD,1452
510 DATA  21,BB,EB,BA,C4,B7,00,BB,1207
520 DATA  C4,BC,00,C3,B4,BB,21,D4,1191
530 DATA  00,18,03,21,DD,00,7E,FE,661
540 DATA  00,C8,CD,5A,BB,23,18,F6,987
550 DATA  00,00,00,00,00,00,00,00,0
560 DATA  00,00,00,18,20,43,41,50,268
570 DATA  53,20,18,00,18,20,53,48,350
580 DATA  49,46,54,20,18,00,00,00,283
590 ' lokldr
```

```
10 ' ************************
20 ' *       LOKVIEW        *
30 ' *    Paul S Jenkins    *
40 ' *        1986          *
50 ' ************************
60 '
70 ' Binary Loader
80 '
90 '
100 SYMBOL AFTER 256
110 address=HIMEM-&F0
120 MEMORY address-1
130 SYMBOL AFTER 240
140 LOAD"!lokview.bin",address
150 CALL address
160 PRINT"'LOKVIEW' is now loaded"
170 PRINT"at address &";HEX$(address,4)
180 PRINT"Press CAPS LOCK a few times"
190 PRINT"to see the routine working..."
200 PRINT:END
210 ' lokview
```

```
LOKVIEW Assembler Listing

.txt_win_enable..........&BB66
.txt_clear_window........&BB6C
.kl_new_frame_fly........&BCD7
.km_get_state............&BB21
.txt_output..............&BB5A

.start
LD A,&07
CALL txt_stream_select
LD DE,&0018
LD HL,&4F18
CALL txt_win_enable
CALL txt_clear_window
XOR A
CALL txt_stream_select
LD DE,&0000
LD HL,&4F17
CALL txt_win_enable
CALL txt_clear_window

.ret_only
NOP
LD DE,&1840
LD HL,entry
LD (HL),D
INC HL
LD (HL),E
LD HL,ret_only
LD A,&C9
LD (HL),A
LD HL,block
LD B,&81
LD C,&00
LD DE,i_routine
JP kl_new_frame_fly

.i_routine
PUSH AF
PUSH BC
PUSH DE
PUSH HL
LD DE,flags
CALL km_get_state
EX DE,HL
LD A,D
CP (HL)
JR NZ,change
LD A,E
INC HL
CP (HL)
JR Z,set_flags


.change
CALL show

.set_flags
LD DE,flags
CALL km_get_state
EX DE,HL
LD (HL),D
INC HL
LD (HL),E
POP HL
POP DE
POP BC
POP AF
RET

.show
LD A,&07
CALL txt_stream_select
CALL txt_clear_window
XOR A
CALL km_get_state
EX DE,HL
CP D
CALL NZ,caps_on
CP E
CALL NZ,shift_on
JP txt_stream_select

.caps_on
LD HL,caps_string
JR print

.shift_on
LD HL,shift_string

.print
LD A,(HL)
CP &00
RET Z
CALL txt_output
INC HL
JR print

.block
DEFS &09

.flags
DEFS &02

.caps_string
DEFB &18
DEF$ ' CAPS '
DEFB &18
DEFB &00

.shift_string
DEFB &18
DEF$ ' SHIFT '
DEFB &18
DEFB &00

END
```

Amstrad Interactive Cross-Referencing

by V Skala

Cross-referencing can be a powerful tool in program debugging. Cross-referencing listings usually provide a listing of the program, followed by a list of all the variables in the program and the lines on which they occur. This routine shows how to find a single variable and the lines on which it occurs, and is done by searching for the internal representation of the variable within the program. All Basic programs are tokenised, or converted into internal formats, called tokens.

On the Amstrad, Basic variables are coded, so that internally they are represented by the ASCII codes preceded by a prefix. This is 040000 for reals, 020000 for strings, and 01000 for integers. The code 0080 (in hex) is added to the code for the last character of the name. The structure of each line internally consists of line length (two bytes), line number (two bytes) and the coded text. A subroutine to find a single variable is given. The strange star character should be typed in as the $ symbol for strings (I assume that the character used is the Czechoslovakian equivalent).

This routine could be used as the basis for a complete cross-referencing program.

```
 1 ij1=0:CLS
 2 GOTO 500
 10 DIM a¤(10)
 20 a¤(0)=STRING¤(200,"#")
 30 POKE @a¤(0),PEEK(&170)
 40 POKE @a¤(0)+1,&72
 50 POKE @a¤(0)+2,&1
 60 FOR j=1 TO LEN(a¤(0))
 70 PRINT HEX¤(ASC(MID¤(a¤(0),j)),2);" ";
 80 NEXT
 90 PRINT
 100 FOR j=&172 TO &180
 110 PRINT HEX¤(PEEK(j),2);" ";
 120 NEXT
 130 END
 500 DEFINT a-z
 510 INPUT "Give variable name",a¤
 520 GOSUB 1000
 530 REM q¤ contains an inner code FOR the name
 540 addr=&170:REM starting address
 550 length=0
 560 line1=0
 570 b¤="#"
 580 WHILE PEEK(addr)+256*PEEK(addr+1)<>0
 590 POKE @length,PEEK(addr)
 600 REM length:=(addr) (length<=255)
 610 POKE @b¤,PEEK(addr)
 620 POKE @b¤+1,PEEK(@addr)
 630 POKE @b¤+2,PEEK(@addr+1)
 640 POKE @line1,PEEK(addr+2)
 650 POKE @line1+1,PEEK(addr+3)
 660 IF INSTR(3,b¤,q¤)<>0 THEN PRINT USING
     "#####";line1;
 670 addr=UNT(addr+length)
 680 WEND
 690 END
1000 REM give an inner representation
     for the a¤
1010 q¤=CHR¤(0)+LEFT¤(a¤,LEN(a¤)-1)
1020 q¤=q¤+CHR¤(&80 OR ASC(RIGHT¤(a¤,1)))
1030 RETURN
```

BBC List Formatter

by B Carroll

MICROTEX 666

This program is available electronically through Microtex 666's software downloading service. It is accessed through Viatel page *66637#.

This program is well commented and fairly self-explanatory, and was originally intended to make the printer skip the perforations at the edge of listing paper. If the printer itself is set to do it, word processor output can often be spoiled.

The code produced should be *RUN from disk.

SEE OUR GREAT SELECTION OF COMPUTE! AND HAYDEN COMPUTER BOOKS

AT TECHNICAL BOOKS AND MAGAZINE COMPANY

HOLT SAUNDERS PTY LIMITED
9 Waltham Street, Artarmon NSW 2064
Phone: (02) 439 3633

TECHNICAL BOOKS AND MAGAZINE COMPANY
289 Swanson Street, Melbourne VIC 3000
Phone: (03) 663 3951

SEE THEM AT STAND No 121 AT THE PC SHOW, MELBOURNE

```
*L.
  100 REM      FILENAME: S.PRLIST.       For BBC Micro with BASIC I or II.
  105 :
  110 REM                  (c) 1985, B.Carroll.
  115 :
  120 REM Filters ASCII codes > &80 from printer stream to stop control char-
  125 REM acters being sent to printer if listings use Teletext colour codes.
  130 REM Lines are counted so that a gap is printed every 56 lines to avoid
  135 REM the perforations. Characters are also counted & line count is incre-
  140 REM mented for every 80 characters sent since the last new line.
  145 REM This keeps track of program lines which are longer than printer line.
  150 :
  155 MODE7:@%=0
  160 :
  165 t$="Listing Formatter for Canon PW-1080A"
  170 PRINTTAB(0,1)CHR$130CHR$141t$
  175 PRINTTAB(0,2)CHR$133CHR$141t$
  180 :
  185 PRINTTAB(0,4)CHR$134"This utility filters characters with"
  190 PRINTCHR$134"codes > &80 from the printer stream."
  195 PRINTCHR$134"It also prints a gap of 10 lines after"
  200 PRINTCHR$134"every 56 printer lines to avoid the"
  205 PRINTCHR$134"perforations of standard fanfold paper."
  210 :
  215 PRINTTAB(0,11)CHR$132"(Just press RETURN for &900)"
  220 PRINTTAB(0,10)CHR$131"ENTER ADDRESS FOR CODE (Hex)"CHR$135"&";
  225 INPUT""S$
  230 IF S$="" S$="900":PRINTTAB(31,10)"900"
  235 S%=EVAL("&"+S$)
  240 :
  245 FOR pass=0 TO 2 STEP 2
  250   P%=S%
  255   [OPTpass
  260   \
  265   \        Set Fn Key 0 to list program  (used only with *RUN PRLIST)
  270   \
  275   .start        LDX #(key MOD 256)
  280                 LDY #(key DIV 256)
  285                 JSR &FFF7              \Use OSCLI to set Fn Key 0
  290   \
  295   \             Clear screen and write message
  300   \
  305                 LDA #12
  310                 JSR &FFEE              \Use OSWRCH for CLS
  315                 LDX #0                 \Set counter for text
  320    read         LDA text,X             \Read text........
  325                 JSR &FFE3              \...& output using OSASCI...
  330                 INX
  335                 CMP #13                \....until end of string
  340                 BNE read
  345                 RTS                    \Wait for Fn Key 0 to be pressed
  350   \
  355   \             Point the WRCHvector to new code
  360   \
  365   .set          LDA &20E
  370                 STA old_wrchv          \Save old vector lo byte.
  375                 LDA &20F
  380                 STA old_wrchv+1        \& the hi byte.
  385                 LDA #(new_wrchv MOD 256)
  390                 STA &20E               \Enter new vector lo byte.
  395                 LDA #(new_wrchv DIV 256)
  400                 STA &20F               \& the hi byte.
  405                 LDA #0
  410                 STA countline          \Set line count to zero
  415                 STA countchar          \Set char count to zero
  420                 RTS                    \Wait for printer stream
  425   \
  430   \             Reset the WRCHvector to normal
  435   \
  440   .unset        LDA old_wrchv
  445                 STA &20E               \Restore old vector lo byte.
  450                 LDA old_wrchv+1
  455                 STA &20F               \& the hi byte.
  460                 RTS                    \Back to normal
  465   \
  470   \             Intercept output & count <CR>s
  475   \
  480   .new_wrchv    STA temp               \Save output character
  485                 CMP #13
  490                 BNE test               \If not CR test for T'text code
  495                 LDA #0
  500                 STA countchar          \Reset character count
  505                 INC countline          \Increase line count
  510                 LDA #56                \56 print lines/page
  515                 CMP countline
  520                 BNE recover            \To output routine if <56....
  525                 LDA #0
  530                 STA countline          \...else reset line count...
  535                 LDA #12
  540                 JSR output             \...& send form feed.
  545   \
  550   \             Test character & replace Teletext codes by space
  555   \
  560   .test         AND #&80               \Clear bits 0 to 6
  565                 CMP #0                 \Check whether bit 7 is clear
  570                 BEQ chars              \If it is, ASCII code < &80
  575                 LDA #32
  580                 STA temp               \If not, replace by a space.
  585   \
  590   \             Count characters sent to output
  595   \
  600   .chars        INC countchar
  605                 LDA #81                \Printer line is 80 characters
  610                 CMP countchar
  615                 BNE recover            \If less than 81 go to output...
  620                 INC countline          \...else add to line count...
  625                 LDA #0
  630                 STA countchar          \...and zero character count.
  635   \
  640   .recover      LDA temp               \Recover stored character...
  645   .output       JMP (old_wrchv)        \...& send to normal output routine.
  650   ]
  655   :
  660   REM Temporary stores & strings
  665   :
  670   old_wrchv=P%:temp=P%+2
  675   countline=P%+3:countchar=P%+5
  680   P%=P%+7
  685   :
  690   [OPTpass:.key:]
  695   $P%="K.0 |LL.07|MCA."+STR$set+"|M|BL.|M|CCA."+STR$unset+"|ML.00|M"
  700   P%=P%+50
  705   :
  710   [OPTpass:.text:]
  715   $P%=CHR$130+"Check printer, key F0 to print listing"
  720   P%=P%+40
  725   :
  730   NEXT pass
  735 :
  740 PRINTTAB(2,13)CHR$130"Copy....."'
  745 PRINTCHR$134"*SAVE PRLIST "~S%" "~P%" "~S%
  750 PRINTTAB(2,18)CHR$130"Alternatively....."
  755 PRINTTAB(0,20)CHR$134"CALL &"~start" to activate the complete"
  760 PRINTTAB(10)CHR$134"autolisting routine."'
  765 PRINTCHR$134"CALL &"~set" to set the print routine."
  770 PRINTCHR$134"CALL &"~unset" or BREAK to disable it."TAB(1,15)
  775 :
  780 @%=10:END
```

Amstrad Toolkit

by J Moffitt

This Basic Toolkit loader program contains many data statements which are followed by a checksum. At the end of the listing, there is a general checksum which checks all the others. If a mistake is found in the program, it will stop with ERROR IN DATA AT LINE xxx; you must then edit that line and change the mistake. When is it POKEd into the RAM, the program may be saved and erased.

To start the toolkit type CALL 40000, which produces the usual message with some special additions. This message can be deleted with a higher call to the RAM (CALL 40003). The program can be saved by typing SPEED WRITE 1:SAVE "BASIC TOOLKIT" [ENTER]. To reload, type SYMBOL AFTER 32:RUN " (only characters after 32 may be redefined). The program can save at 4000 baud if TURBO is added to the previous command.

To check each block of the program, type CAT and rewind the tape to the correct place, then press PLAY.

The Amstrad CPC464 has a feature to add extra commands called RSXs (reserved system extensions). They are prefixed with a I,[SHIFT]&@.

The ROM must know the location of the command table, and must have a four-byte buffer. The command table contains the addresses of the commands, the name table contains the names. Fig 1 is a rough example in assembly language.

Where an @ appears this must be added, or the system may crash. IDPRINT must have a $; IDPRINT,"HELLO" will not do. The commands using variables not prefixed with @ can be entered with the value rather than the variables; ISCROLL,2 will do. If there are mistakes after the first run-through, the code should be reloaded. Any incorrect spellings will upset the machine and Basic can not be recalled. If a REM statement contains an RSX I, you must use II as a single I is deleted.

The following information is important:

×%=PEEK (45512)......This will set ×% to the present mode number.
×%=PEEK (46311)......This will set ×% to the value of shift lock, 255 on.
×%=PEEK (46312)......This will set ×% to the value of caps lock, 255 on.
×%=PEEK (45711)......This sets ×% to the present ITPATTERN value.
×%=PEEK (45712)......This sets ×% to the present IBPATTERN value.

```
Amstrad Assembler  (v1)   Code assembled @40000 (Any origin will do!)

           LD HL,buffer   Declare buffer address
           LD BC,comtab   Declare command table address
           CALL &BCD1     Call routine to set up the RSX commands in ROM.
           RET            Return to Basic.
buffer     DEFS 4         Define 4 spaces of machine buffer.
comtab     DEFW nmetab    Tell the machine where the names are.
           JP command1    Jump to the first command.
           JP command2    Jump to the second command.
           JP ...         Set all the command values.
nmetab     DEF$ 'ANAM'    Declare the command name,except the last character,
                          upper case letters only!
           DEFB 'E'+&80   Add 80(HEX) to the last character in the command.
           DEF$ 'BNAM'    Define the second name.
           DEFB 'T'+&80   Add 80(HEX) to the last character.
           DEF$ ...       Define all the command names.
           DEFB 00        Tell the computer where to end.
command1   ............   Normal machine code routine start here.
           RET
command2   ............   Second command.
           RET
.........  ............   Define all the other commands...
```

Fig 1 An assembler version of the command table

```
:CAPSON.............Turn capitals lock on.
:CAPSOFF............Turn capitals lock off.
:SHIFTON............Turn shift lock on.
:SHIFTOFF...........Turn shift lock off.
:MOTORON............Turn cassette motor on.
:MOTOROFF...........Turn cassette motor off.
:DPRINT,@x$.........Double height printing.
                    Eg.
                    A$="HELLO THIS IS DOUBLE":CLS::DPRINT,@a$:LOCATE 1,10
:RESET..............Reset to the default colours/mode.
:COLOUR,x%..........Changes graphics colour.
:TAGCHR,x%..........Places a character at the graphics cursor.
:PUTCHR,x%..........Places any character on the screen, including control codes'
                    Eg.
                    FOR F=1 to 255::PUTCHR,F:NEXT
x%=0::COPYCHR,@x%...Takes character at the cursor and places it in x%, if two
                    characters are the same the the first is read.
:TURBO..............Changes save speed to around 4000 baud.
:FAST...............Changes save speed to around 2000 baud.
:SLOW...............Changes save speed to around 1000 baud.
:HOME...............Moves the cursor to the "home" position.
:ESCOFF.............Completely disables the ESC key.:INKEY returns 252 for ESC.
x%=0::INKEY,@x%.....Waits for the next key press and puts the character in x%.
:INVERSE............Swaps pen and paper inks.
:BELL...............Sound a "bell".
:CURSORON...........Turns on the cursor for INPUTs.
:CUSROROFF..........Turns off the cursor for INPUTs.
:SCREENON...........Turns the screen on.
:SCREENOFF..........Turns the screen off, useful for security checks on progr
                    so that the code is not seen.
:TPATTERN,x%........Changes the text colour to a set pattern, 0 to 255.
:BPATTERN,x%........Changes the background colour to a set pattern, 0 to 255.
:FRAME..............Waits for the next frame fly-back, makes sprite movement
                    better.
:KEYBOARD...........Resets the keyboard, to the default, all functions keys
                    reset.
:CLEARINPUT.........Clears the keyboard buffer.
:OVERON.............Turns on "over", all characters are placed on top of each
                    other.
:OVEROFF............Turns off "over".
:SCROLL.............Scrolls the screen 25 lines up.
:SCROLL,x%..........Scrolls the screen x% lines up.
:SCROLL,a%,b%,c%,d%,e%,f%,g%...
                    Scrolls a screen window with a%,b%,c%,d% being the normal
                    columns and rows as WINDOW, e% for up/down (0=up,1=down).
                    f% for number of scrolls and e% for the pattern to fill
                    the vacant lines to, 0 to 255 as :BPATTERN.
```

The command list

```
100 REM
110 REM RSX EXTENDED BASIC LOADER
120 REM
130 REM  (C)1985 Justin Moffitt
140 REM
150 MODE 2:count=0:MEMORY 39999
160 FOR value=40000 TO 40989 STEP 22
170  sum=0
180  FOR place=0 TO 21
190   READ number$
200   POKE value+place,VAL("&"+number$)
210   sum=sum+VAL("&"+number$)
220  NEXT
230  READ l$:main=main+VAL(l$)
240  IF VAL(l$)<>sum THEN 740
250  count=count+10
260 NEXT
270 IF main<>109426 THEN PRINT "ERROR IN MAIN SUM" ELSE CALL 40000:'NEW
280 DATA CD,43,9E,01,51,9C,21,4D,9C,CD,D1,BC,C9,22,FF,03,28,B3,9C,C3,4D,9E,2834
290 DATA C3,53,9E,C3,71,9E,C3,A8,9F,C3,A9,9E,C3,B0,9E,C3,D0,9E,C3,A2,9E,C3,3746
300 DATA C5,9E,C3,8B,9E,C3,FB,9E,C3,59,9E,C3,5F,9E,C3,BE,9F,C3,BB,9E,C3,9A,3771
310 DATA 9E,C3,96,9E,C3,83,9E,C3,87,9E,C3,ED,9E,C3,E2,9E,C3,9E,9E,C3,65,9E,3767
320 DATA C3,6B,9E,C3,77,9E,C3,7D,9E,C3,D4,9E,C3,DB,9E,C3,92,9E,C3,02,9F,C3,3597
330 DATA 08,9F,C3,14,9F,43,41,50,53,4F,CE,43,41,50,53,4F,46,C6,42,45,4C,CC,2181
340 DATA 44,50,52,49,4E,D4,43,4F,4C,4F,55,D2,49,4E,4B,45,D9,49,4E,56,45,52,2089
350 DATA 53,C5,54,41,47,43,48,D2,43,4F,50,59,43,48,D2,50,55,54,43,48,D2,43,2178
360 DATA 4C,45,41,52,49,4E,50,55,D4,53,48,49,46,54,4F,CE,53,48,49,46,54,4F,1948
370 DATA 46,C6,53,43,52,4F,4C,CC,52,45,53,45,D4,48,4F,4D,C5,46,52,41,4D,C5,2290
380 DATA 4D,4F,54,4F,52,4F,CE,4D,4F,54,4F,52,4F,46,C6,4F,56,45,52,4F,CE,4F,2114
390 DATA 56,45,52,4F,46,C6,45,53,43,4F,46,C6,43,55,52,53,4F,52,4F,CE,43,55,2065
400 DATA 52,53,4F,52,4F,46,C6,53,43,52,45,45,4E,4F,CE,53,43,52,45,45,4E,4F,1933
410 DATA 46,C6,54,50,41,54,54,45,52,CE,42,50,41,54,54,45,52,CE,4B,45,59,42,2057
420 DATA 4F,41,52,C4,54,55,52,42,CF,46,41,53,D4,53,4C,4F,D7,00,7E,B7,C8,CD,2543
430 DATA 5A,BB,23,18,F7,0A,20,41,6D,73,74,72,61,64,20,36,34,4B,20,4D,69,63,1867
440 DATA 72,6F,63,6F,6D,70,75,74,65,72,20,20,28,76,31,29,0A,0A,0D,20,A4,31,1694
450 DATA 39,38,34,20,41,6D,73,74,72,61,64,20,43,6F,6E,73,75,6D,65,72,20,45,1890
460 DATA 6C,65,63,74,72,6F,6E,69,63,73,20,70,6C,63,20,20,20,20,20,20,20,20,1685
470 DATA 20,20,20,20,20,20,20,20,4C,6F,63,6F,6D,6F,74,69,76,65,20,53,6F,66,1641
480 DATA 74,77,61,72,65,20,4C,74,64,2E,20,20,20,20,20,20,20,20,20,20,20,20,1301
490 DATA 20,20,20,20,20,20,20,20,20,61,6E,64,20,4A,75,73,74,69,6E,20,4D,6F,1484
500 DATA 66,66,69,74,74,2E,0A,0A,0D,20,45,58,54,45,4E,44,45,44,20,42,41,53,1491
510 DATA 49,43,20,31,2E,30,20,0A,00,CD,B8,9E,21,8F,9D,CD,B6,9D,C9,3E,FF,32,2304
520 DATA E8,B4,C9,3E,00,32,E8,B4,C9,3E,FF,32,E7,B4,C9,3E,00,32,E7,B4,C9,3E,3103
530 DATA 03,CD,5A,BB,C9,3E,02,CD,5A,BB,C9,3E,07,CD,5A,BB,C9,3E,06,CD,5A,BB,2735
540 DATA C9,3E,15,CD,5A,BB,C9,CD,6E,BC,C9,CD,71,BC,C9,DD,7E,00,CD,5D,BB,C9,3411
550 DATA CD,03,BB,C9,CD,19,BD,C9,CD,70,BB,C9,CD,48,BB,C9,DD,7E,00,CD,FC,BB,3577
560 DATA C9,DD,7E,00,CD,DE,BB,C9,CD,18,BB,DD,6E,00,DD,66,01,77,C9,CD,FF,BB,3401
570 DATA CD,BA,BB,CD,4E,BB,C9,CD,60,BB,DD,6E,00,DD,66,01,77,C9,CD,9C,BB,C9,3461
580 DATA DD,7E,00,32,8F,B2,C9,DD,7E,00,32,90,B2,C9,3E,16,CD,5A,BB,3E,00,CD,2672
590 DATA 5A,BB,C9,3E,16,CD,5A,BB,3E,01,CD,5A,BB,C9,CD,06,B9,CD,ED,1C,CD,09,2870
600 DATA B9,C9,21,5F,00,3E,05,CD,68,BC,C9,21,A7,00,3E,32,CD,68,BC,C9,21,4D,2399
610 DATA 01,3E,19,CD,68,BC,C9,CD,03,B9,CD,06,B9,3A,3C,9F,CD,A5,BB,38,05,CD,2675
620 DATA 06,B9,18,03,CD,09,B9,11,3D,9F,01,08,00,ED,B0,C9,00,00,00,00,00,00,1477
630 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,01,10,00,0017
640 DATA 21,3D,9F,11,45,9F,ED,A0,28,ED,A0,78,B1,20,F7,C9,3E,FE,21,45,9F,CD,2894
650 DATA A8,BB,3E,FF,21,4D,9F,CD,A8,BB,C9,CD,78,BB,E5,3E,FE,CD,5D,BB,E1,E5,3698
660 DATA 2C,CD,75,BB,3E,FF,CD,5D,BB,E1,24,CD,75,BB,C9,7B,32,3C,9F,CD,1D,9F,3111
670 DATA CD,55,9F,CD,68,9F,CD,79,9F,C9,00,00,00,00,1A,F5,13,1A,4F,13,1A,47,2114
680 DATA C5,0A,5F,CD,93,9F,C1,F1,03,3D,C8,F5,18,F2,FE,00,28,2D,FE,01,28,2D,2701
690 DATA FE,07,C0,DD,7E,00,DD,4E,02,DD,46,04,DD,5E,06,DD,6E,08,DD,56,0A,DD,2594
700 DATA 66,0C,18,24,F5,E5,C5,D5,CD,50,BC,D1,C1,E1,F1,0D,C8,18,F1,0E,19,18,2940
710 DATA 03,DD,4E,00,C5,CD,17,BC,50,C1,3E,00,06,01,1E,18,6F,67,F5,79,FE,00,2145
720 DATA 28,03,F1,18,D3,F1,C9,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0961
730 END
740 PRINT "ERROR IN LINE":count+280
```

VZ Pause

VZ Pause is a short routine for the VZ-200 which enables the computer to be 'paused' at any time. A pause can be initiated by pressing Shift-X. A short beeb will be produced to confirm that a pause has begun and pause can be terminated by pressing Shift-C, and again a short beeb will confirm this. The routine uses interrupts, and so will work with any software that does not disturb these interrupts. To use, type in the routine, and then CSAVE it immediately, as the program self-destructs when run. When the program is run, the pause facility becomes operational.

The program works in the following fashion:

● Lines 10-20 lower the RAMTOP to create space for a short machine language program

● Lines 30-40 set the address for the interrupt exit

● Lines 50-80 POKE the machine language program into the memory

● Line 90 makes the interrupt operational

● Line 100 clears the Basic routine from memory. This is necessary to prevent the system crashing should the routine be RUN twice.

```
1Ø TM=PEEK(3Ø898)*256+PEEK(3Ø897)-35
2Ø POKE3Ø897,TM-INT(TM/256)*256:POKE3Ø898,INT(TM/256)
3Ø TM=TM+1
4Ø POKE3Ø846,TM-INT(TM/256)*256:POKE3Ø847,INT(TM/256)
5Ø TM=TM-65536
6Ø FORA=ØTO31
7Ø READB:POKETM+A,B
8Ø NEXT
9Ø POKE3Ø845,2Ø5
1ØØ NEW
11Ø DATA33,15Ø,Ø,1,7Ø,Ø,58,251,1Ø4,254,121,192,2Ø5,92,52,58,251
12Ø DATA1Ø4,254,115,32,249,33,2ØØ,Ø,1,6Ø,Ø,2Ø5,92,52,2Ø1
```