# * WIZZDOM *

A newsletter for owners of the Dick Smith WIZZARD and Funvision computers.
Issue 3 - written by Barry Klein.
January 1985

## EDITORIAL

During November, DSE released a "16k RAM expansion module" for the WIZZARD. This unit may have 16k of memory in it, but as far as I can determine without opening it, you can only access 14k at most, or even less if it does not disable the I/O (printer) interface. The memory can only be accessed through BASIC by POKEs and PEEKs: it does not add to the capabilities of BASIC in any real way. I therefore reccommend that you do not buy it. At the present time, there does not seem to be any move by DSE to produce a version of BASIC that can use the 16k expansion properly.

## The WIZZARD memory maps.

Yes maps: there are two independant memory maps for the WIZZARD: one for the processor and one for the Video Display Processor (VDP).
NOTE: this article is very technical, but if you can wade through it, it may help you understand the PRESTIDIGITATIONS.
The memory maps are drawn on the next page.
The processor memory map is associated with the games and BASIC cartridges: they provide the program which operates the processor. The BASIC language is interpreted by the program in the BASIC cartridge. This means that the BASIC program is stored as ASCII characters and analysed at the time of execution (when you type "RUN"). The games cartridges, however contain programs that the processor can execute directly, without any analysis.
The only RWM (read/write memory) in the processor map is 1k from 0 to 1023, and repeated to 4095. This is used by BASIC to store variable items such as: where to write the next character on the screen (PRINT statement), where to RETURN after a GOSUB etc. None of the information associated with BASIC variables is stored here.
The VDP memory map is entirely RWM, 16k of it. It is used to store the BASIC program that you type in, the values of the variables and the data relating to the screen display.
Your BASIC program is stored, one character per byte,in the Video RWM as ASCII characters: ie the values stored correspond to the values of the characters obtained with the ASC function. In the display area, however, 160 is added to the character before storing.
By changing certain locations in processor memory, it is possible to write characters into places they would not normally go e.g. you can PRINT into program area or DATA area and you can make DATA statements write into program area or display area.

```
          Processor                                    Video
FFFF                      65535              3FFF                     16383
F800  2k ROMØ             63488              3C00 1k                  15360
F7FF                      63487              3BFF                     15359
F000  2k?                 61440                   8k  BASIC
EFFF                      61439                   program
E800  I/O Interface       59392              1C00                      7168
E7FF                      59391              1BFF                      7167
      10k memory
       expansion?
C000                      49152              13C0                      5024
BFFF  16k ROM1            49151              13BF 32  bytes            5023
      (BASIC in                              1380 Colour table         4992
8000   upper 8k)          32768              137F 128 bytes            4991
                                             1300 Sprite attribute     4864
7FFF                      32767              12FF 768 bytes            4863
      16k ROM2                               1000 Video display        4096 _
      (BASIC in                                                              |
4000   upper 4k)          16384              OFFF                      4095  |
3FFF 16383                                                                  2k
3000 VDP write            12288                                             sprite
2FFF 4k              ,    12287              0900                      2304 shape
2000 VDP read              8192              08FF 256 bytes            2303  |
1FFF 4k PIA                8191              0800 DATA area            2048 _|
1000 (KBD, sound)          4096              07FF 2k character gen.    2047
OFFF 4k RWM                4095              0000 256 *   8 bytes         0
0000 1k from   0000           0
```

WIZZARD memory maps as deduced from circuits & BASIC

The following locations (in the processor map) specify where data will be written into video RWM (in the Video map).

| | | |
|---|---|---|
| 219 high byte | | |
| 218 and 846 added together – low byte | | PRINT statement |
| 170 high byte | | |
| 171 low byte | | DATA statement |
| 214 high byte | | |
| 213 low byte | | program entry |

The values required for the high and low bytes can be calculated as :

```
  nn10 H=INT(A/256)          A is the address in video RWM
  nn20 L=A-256*H             L is the low byte value
  nn30 H=H+192               H is the high byte value (the additional 192
                                 is required to set the VDP in the right mode)
```

You need to be very careful about what you do if you want to change the program entry locations 214 and 213. You may irretrievably corrupt your program if you do the wrong thing. The same applies to PRINTing or writing DATA into the program area (High byte in range 220 to 255).

## CURSES

There is an error with the DIM statement which, however, should not cause problems in most cases. The error is that whatever character is in the position that the left bracket '(' should be,is accepted as correct. This will only cause problems if you omit the bracket e.g. instead of DIM A(20), you input DIM A20) : this will be taken as DIM A(0).
If you attempt to enter a program that is too big, you get the message "SYNTAX ERROR" You will normally get the message while you are entering the program line. When you do, you must stop entering the program or else you can get a corrupted program which can't be corrected. NOTE: deleting a program line by just entering the line number does not delete it from program memory. If you run out of program memory, immediately CSAVE it and then CLOAD it again. This will eliminate lines which have been edited or deleted during program entry.

## SPELLS

The following instruction will print the number of bytes left in program memory. It can be used in a program or as a direct command:

```
   PRINT 256*(251-PEEK(214))+255-PEEK(213)
```

Try entering this as line 9999 , then, while entering a program, RUN 9999 or GOTO 9999 from time to time. Include line deletions and edits.

## PRESTIDIGITATIONS

### 'PRINTING' with DATA statements.

It is possible to write characters to the screen using DATA statements. Locations 170 and 171 should be set to the equivalent values that you'd set locations 219 and 218 respectively. There will not be any scrolling and if you attempt to set L171 past 255, you will get ERROR 19 - data area overflow. The characters following the word DATA will be copied to the screen, followed by the 'RETURN' character. The DATA statement writes the true ASCII value, not the modified value used by PRINT. If you want to use genuine DATA statements later, remember to set locations 170 and 171 to 200 and 0 respectively.

CHARACTER TABLE

| 'DATA' | ASCII | 'CHAR' | 'DATA' | ASCII | 'CHAR' |
|---|---|---|---|---|---|
| ! | 33 | 129 | Ø | 48 | 144 |
| " | 34 | 130 * | 1 | 49 | 145 |
| # | 35 | 131 | ? | 63 | 159 |
| $ | 36 | 132 | A | 65 | 161 |

NOTE: characters with values over 127 cannot be PRINTed.

* quotes characters must be paired in DATA statements

'RETURN' (ASCII 13) is CHAR 109

PRESTIDIGITATIONS (cont)

SELF MODIFYING PROGRAMS
NOTE: all good books warn that this type of program is highly undesirable.
If you make a mistake, your program can get itself into real trouble.
Use only with extreme care.
This is an extension of the previous item, illustrating how to change a program using DATA statements.

Start of program area is 220,0 followed by 5 characters for line number & space, then 4 characters for DATA: therefore the Ø is at the tenth position (220,9).

```
10   DATAØ
20   READ A
30   POKE 170,220
40   POKE 171,9
50   DATA1
60   IF A>0 THEN 1000
65   CLS
70   PRINT"I AM YOUR FRIENDLY WIZZARD"
80   PRINT
90   PRINT"COMPUTER. TRY TO GUESS MY"
100  PRINT
110  PRINT"NUMBER. IT IS BETWEEN 1 & 100 "
120  PRINT"YOUR GUESS";
130  INPUT A
140  PRINT
150  IF A<1 OR A>100 THEN 500
160  PRINT"FANTASTIC! YOU GOT IT FIRST"
170  PRINT"TIME!"
180  PRINT
190  PRINT"DO YOU WANT TO PLAY AGAIN";
200  INPUT A$
210  IF LEFT$(A$,1)<>"Y" THEN 180
220  PRINT
230  PRINT"WELL I DONT!"
240  STOP
500  PRINT"I'M NOT GOING TO PLAY WITH"
510  PRINT
520  PRINT"A CHEAT!"
550  GOTO 240
1000 PRINT
1010 PRINT"I TOLD YOU I DIDN'T WANT"
1020 PRINT
1030 PRINT"TO PLAY."
1040 GOTO 240
```

This program will only work correctly if line 10 is the first line entered and it is not changed later.

WIZZARDRY (modifications to the hardware or firmware)
I have been able to modify BASIC to make it run faster, by changing some of the program in the BASIC cartridge. This can only be done by experts with appropriate equipment (a PROM programmer). If you are interested, and know someone who can do the job, I can give you the list of locations to change.
The modification works by not waiting for a keyboard scan every instruction (see WIZZDOM issue 2). This does not give a dramatic improvement for most programs, but can speed up programs written for maximum speed. The modifications which allow the 'fast' BASIC to run faster would have no effect on the normal BASIC.
Some examples of improvements are given below.

```
        PROGRAM              'OLD' BASIC        'FAST' BASIC
10 FOR 1=1 TO 1000
20 NEXT I                    20 seconds           3.5 seconds (approx)

10 FOR 1=1 TO 1000
20 A=A+1
30 NEXT I
40 PRINT A                   40 seconds           22 seconds

10 I=1
20 FOR J=1 TO 1000
50 A=A+I
60 NEXT J
50 PRINT A                   40 seconds             9 seconds

10 I=1
20 FOR J=1 TO 1000
30 LET A=A+I
40 NEXT J
50 PRINT A                   40 seconds             8 seconds
```

The differences in execution time in the 'fast' BASIC between the very similar programs are caused by variations in what BASIC has to do to analyse or execute the instructions.

A+I is faster than A+1 bacause BASIC knows where I can be found and that it is in the right form to be added to A without modification. On the other hand, '1' must be converted from an ASCII character to a 'floating point' number before it can be added to A.
The last example is faster than the previous one because BASIC first searches the instruction line for a BASIC keyword, such as PRINT, READ, INPUT etc. As LET is the 2nd keyword out of 30, the type of instruction is found very quickly. In the third example above, the entire list of keywords is searched before BASIC assumes that line 30 is equivalent to a LET statement.

For those of you who are interested, I can supply copies of the circuit diagrams for the WIZZARD.

Postscript: the 4—colour printer—plotter can be put into graphics mode with
            LPRINT CHR$(8);CHR$(18)

## WIZZARD USER GROUP (VIC) PROGRAM LIBRARY

Program library tapes are C20 computer tapes. Each tape will contain two copies of each program, one on each side of the tape. A tape will hold up to *6* programs, depending on size.
The cost is $2 per tape plus postage and packaging. Blank tapes are available for $1 plus postage and packing. The first tape from the program library is free of copying charges. (you can supply your own tape if you wish).

ODDEVEN
This is a game similar to, but more difficult than number Mastermind.
3, 4 or 5 digit numbers.
VENTURE
A simplified adventure-type game — find the hermit's box and return it to the town for a reward.
WIZZARD SKETCH II
Based on the program in the First Book, but with colour and line drawing.
SOUND EFFECTS & SOUND GENERATOR
Based on the programs in the EA article. Demonstrate the capabilities of the WIZZARD sound generator and design your own effects.
ICBM
Defend a city from an incoming ICBM by directing your anti-ICBM by 'radar'
FRUIT MACHINE
Play the pokies with pretend money. Great graphics.
MONSTER MEAL
Avoid a monster 'from outer space' while picking up 'stars'
BREAKOUT
Knock down the wall of bricks
LOAN CALCULATOR
Calculates interest payments for a loan. WARNING results should only be taken as a guide.
POWERS & FACTORIALS
Calculate powers and factorials of numbers up to 100 digits
MUSIC COMPOSER (from W.A. User Group)
Program up to 42 notes, with chords
ELEPHANT ALPHABET (from New Zealand)
Match the lowercase letter displayed with the uppercase letter on the keyboard to draw a picture of an elephant.
Coming soon: CHARACTER GENERATOR
A 'utility' program to let you design characters and groups of characters for graphic displays