* VZ-LINK *
NEWSLETTER

<< AUCKLAND VZ 300/200 USERS CLUB >>

ISSUE # 22 JULY 1988

Well,another month is here already,isn't the year going fast.

The workshop/meeting went really well.We had four VZ computer's running.I hope everyone had a super time,and managed to get something from the day.
Don't forget to mark the date of the next one on your calendar,and try to attend. The next workshop will be held on:

SATURDAY 27Th AUGUST 88

With Dick Smith's selling of their VZ stock's,it has now become obvious that their support is about to end.

I feel that this move won't effect the more established VZ Computer users.
However the newer and younger users will feel the pinch,
when it comes to obtaining Disk drives and Ram expansion unit's, As well as the other add-on's that maybe needed.
This is were the VZ Computer user club's are really going to be needed.
We are already lucky enough to have members who have the technical knowledge to overcome the lack of these add-on's.
And we will now have to investigate the making or obtaining these items for the members.

Ram units shouldn't be a problem,but obtaining disk drive's and drive controllers will be.

If you are able to help,by writing items to solve these problems or you can help build these units please do so and let's keep the vz up and running for years to come.

There's been a couple of hardware projects recently published.
Checkout page 2 for details.

We are now in the software marketing business.
Thank's to Scott Le Brun we how hold a limited stock of software for sale.
We are not out to make a profit,the idea is to insure that this software is available to the New Zealand VZ user to purchase without to much trouble.
Anyway,details are on the back page.

I won't tell you what else is in this issue.
So you will have to open it to find out.

SEE YOU NEXT ISSUE,CHEERS

RECENT MAGAZINE ITEMS FOR THE VZ

AEM APRIL '88 - VZ ULTRA GRAPHICS ADAPTER. This project, once built and installed enables all of the graphic modes that the 6847 (the video generator chip currently in the VZ) can produce. Not only can we use graphics which has 256x192 pixels (compared to 128*64), but we can switch over to an external character set containing upper & lower case letters aswell as *computer* type letters plus the greek alphabet for scientific formulae. Overall this project is a real eye opener! AEM estimates that it will cost about only $50! But beware, the installation is a headache - 57 interconnections!

ETI MAY '88 - VZ EPROM PROGRAMMER. This project, when completed, plugs into the memory expansion slot. It programs 2764, 27128, 27256 type eproms. The module is software controlled. I haven't seen the software yet so I cannot comment on it. Not only can it program eproms but it can copy from one to another on the same board. The module is powered from the VZ so no power supply is needed. Also the eprom programmer features an edit mode, to ***modify*** the eprom before programming or copying. It also has provision for an extra 4k of RAM to be fitted for your personal use when you are not programming eproms. This is one project that I recommend to anyone who's serious with the hardward side of the VZ. The author of the article even recommends buying a VZ just to use with this eprom programmer as the cost of both put together is well below the cost of commercial eprom programmers.

GAMERS HI SCORE LIST

This list is for New zealand VZ Game players.
Any VZ game is eligible to be entered on this list.
To register your high score all you have to do is send it in with the following details.
:Your Name
:Game Name
:Score and Date

Sign it,also get a witness to sign to verify your score.
This list rely's on your Honesty.
We will accept unverified scores,if there is no one available(at 2am)

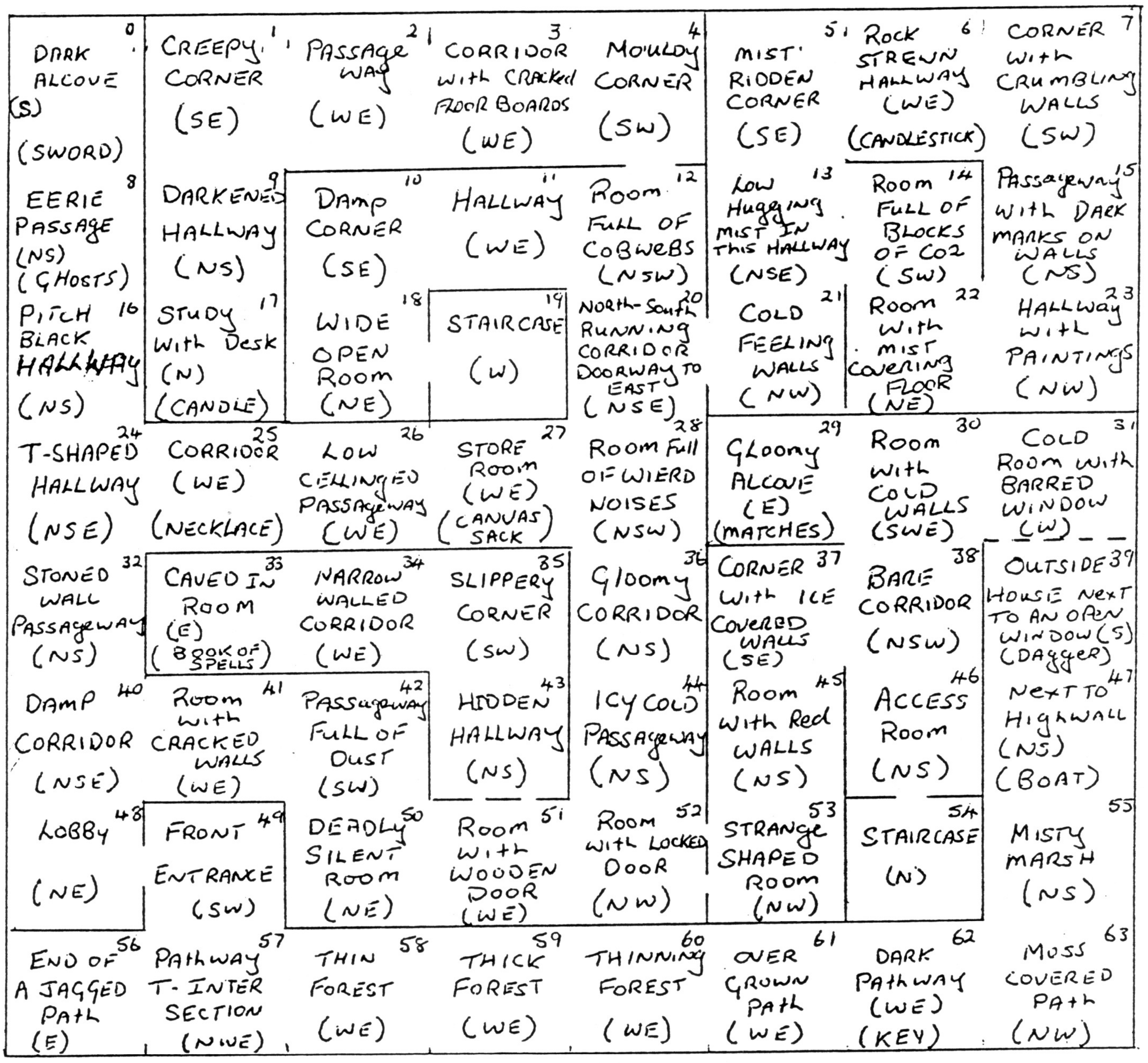
All scores submitted will be compared with Scott's list in VZ Down Under Newsletter. (see april issue)
If you do get a higher score then that shown on his list,
I will forward details to scott .

GAME	SCORE	NAME
GALAXON	12510	PETER HILL
HAMBURGER SAM	55700	RICHARD SPONG
DAWN PATROL	99900	RICHARD SPONG
GHOST HUNTER	12690	PETER HILL
MISSILE ATTACK	37490	PETER HILL
PLANET PATROL	700	BRENDA HILL

WRITING ADVENTURE GAMES

WRITTEN BY SCOTT LE BRUN

Below is an example MAP of an ADVENTURE land.The theme of this one is HAUNTED HOUSE.I have listed down all the traps and their solutions.



0 DARK ALCOVE (S) (SWORD)
1 CREEPY CORNER (SE)
2 PASSAGE WAY (WE)
3 CORRIDOR WITH CRACKED FLOOR BOARDS (WE)
4 MOULDY CORNER (SW)
5 MIST RIDDEN CORNER (SE)
6 ROCK STREWN HALLWAY (WE) (CANDLESTICK)
7 CORNER WITH CRUMBLING WALLS (SW)
8 EERIE PASSAGE (NS) (GHOSTS)
9 DARKENED HALLWAY (NS)
10 DAMP CORNER (SE)
11 HALLWAY (WE)
12 ROOM FULL OF COBWEBS (NSW)
13 LOW HUGGING MIST IN THIS HALLWAY (NSE)
14 ROOM FULL OF BLOCKS OF CO2 (SW)
15 PASSAGEWAY WITH DARK MARKS ON WALLS (NS)
16 PITCH BLACK HALLWAY (NS)
17 STUDY WITH DESK (N) (CANDLE)
18 WIDE OPEN ROOM (NE)
19 STAIRCASE (W)
20 NORTH-SOUTH RUNNING CORRIDOR DOORWAY TO EAST (NSE)
21 COLD FEELING WALLS (NW)
22 ROOM WITH MIST COVERING FLOOR (NE)
23 HALLWAY WITH PAINTINGS (NW)
24 T-SHAPED HALLWAY (NSE)
25 CORRIDOR (WE) (NECKLACE)
26 LOW CEILINGED PASSAGEWAY (WE)
27 STORE ROOM (WE) (CANVAS SACK)
28 ROOM FULL OF WIERD NOISES (NSW)
29 GLOOMY ALCOVE (E) (MATCHES)
30 ROOM WITH COLD WALLS (SWE)
31 COLD ROOM WITH BARRED WINDOW (W)
32 STONED WALL PASSAGEWAY (NS)
33 CAVED IN ROOM (E) (BOOK OF SPELLS)
34 NARROW WALLED CORRIDOR (WE)
35 SLIPPERY CORNER (SW)
36 GLOOMY CORRIDOR (NS)
37 CORNER WITH ICE COVERED WALLS (SE)
38 BARE CORRIDOR (NSW)
39 OUTSIDE HOUSE NEXT TO AN OPEN WINDOW (S) (DAGGER)
40 DAMP CORRIDOR (NSE)
41 ROOM WITH CRACKED WALLS (WE)
42 PASSAGEWAY FULL OF DUST (SW)
43 HIDDEN HALLWAY (NS)
44 ICY COLD PASSAGEWAY (NS)
45 ROOM WITH RED WALLS (NS)
46 ACCESS ROOM (NS)
47 NEXT TO HIGHWALL (NS) (BOAT)
48 LOBBY (NE)
49 FRONT ENTRANCE (SW)
50 DEADLY SILENT ROOM (NE)
51 ROOM WITH WOODEN DOOR (WE)
52 ROOM WITH LOCKED DOOR (NW)
53 STRANGE SHAPED ROOM (NW)
54 STAIRCASE (N)
55 MISTY MARSH (NS)
56 END OF A JAGGED PATH (E)
57 PATHWAY T-INTERSECTION (NWE)
58 THIN FOREST (WE)
59 THICK FOREST (WE)
60 THINNING FOREST (WE)
61 OVER GROWN PATH (WE)
62 DARK PATHWAY (WE) (KEY)
63 MOSS COVERED PATH (NW)

Here's the LIST of treasures and objects that our hero is going to collect during the game.

TREASURES	OTHER OBJECTS
1) GOLD COINS	5) CANDLE
2) BOOK OF MAGIC SPELLS	6) CANDLESTICK
3) DIAMOND NECKLACE	7) MATCHES
4) JEWELED DAGGER	8) CANVAS SACK
	9) SWORD
	10) KEY
	11) BOATS

We can also have "props" in our ADVENTURE. i.e-Paintings on wall,or blocks of carbon dioxide ice on the floor. These are things that the player cannot actually pick up,but can serve as a clue or just add realism to the scene.

TRAPS AND SOLUTIONS

Bars in window won't allow access through it.
-The bars are rusted through.Cut them with the sword.

Door locked-Unlock with key.

Can't open wooden door-Burn down with matches.

Giant spider in cobwebby room won't let you pass.
-Say magic word from the book of magic spells.

Ghosts won't let you pass.
-Say magic word from the book of magic spells.

Darkness-Put candle into candlestick and light with matches.

Can't carry more than 3 objects.
-Get canvas sack.It will allow you to store every object.

VERB LIST

This is a list of all the possible verbs that the player will be able to use,to do various things in his/her surroundings.

N,S,W,E - Compass points allow the player to move around the land using North,South,West,East.
GET (object) - The player picks up objects.
DROP (object)
EXAMINE (object or prop) - May give away a clue.
PUT (object) in (object) - Places one item inside another,if possible.
SAY (magic word) - May help you get out of a sticky situation.
BURN (object) - Burns the object,if player is carrying matches.
CUT (object) - Player must have sword.
UNLOCK door - Player must have key.
CLIMB stairs - Allows player to move from one set of stairs to another.
READ book - Read through book of spells,tells you some magic words.
LIGHT candle - Sheds some light onto the situation.

Now we can put all our work into the computer.
We first place all the descriptions and their exits into arrays.
(The arrays must be DIMensioned first)

```
EG:- 60000 DATA "DARK ALCOVE","CREEPY CORNER","PASSAGEWAY"
     60010 DATA "CORRIDOR WITH CRACKED FLOOR BOARDS"
        "
        "
        "
     60890 DATA "MOSS COVERED PATH"
     60900 FOR I=0 TO 63:READ D$(I) :NEXT
```

The last line above READs the DATA into array -D$( ).
Why we do this,will become clearer later when we use it.

Now we enter in their relative exits.

```
 60910 DATA "S","SE,"WE","WE"
   "
   "
 61710 DATA "NW"
 61720 FOR I=0 TO 63:READ E$(I):NEXT
```

Now we READ in the exits for each individual location.
EG:-Location # 0 - Exit South.
or Location #63 - Exits Nortn and West.
This allows us to check to see if the player is allowed to move in that direction that they wish to move.
We now enter in the VERBS and NOUNS and read them into arrays.

```
 61800 DATA "N","S","W","E","GET"
 61810 DATA "DROP"
   "
   "
 61890 FOR I=1 TO 15:READ V$(I):NEXT
 61900 DATA "GOLD COINS","BOOK","NECKLACE"
   "
   "
 62000 FOR I=1 TO 11:READ O$(I):NEXT
```

We are now ready to write the MAIN BODY of our ADVENTURE program.
We first must CLS and print the location where the player is.
We will store the room number in "R".
IE:- If we are in room number 11 which is the "HALLWAY",then R=11:
For example:

```
 100 CLS:PRINT:PRINT
 110 PRINT D$(R) :PRINT
 120 PRINT "THE ONLY POSSIBLE EXITS ARE -";E$(R)
```

This will print up the description and possible exits from the location.

```
 130 PRINT "WHAT DO YOU WANT TO DO NOW?";
 140 INPUT Q$
```

Gets the next instruction from the player.
The instructions form a two or three word sentence telling the computer what to do.
The first word is a VERB while the second and third are usually NOUNS.
The computer must pull apart your instuction into a VERB and NOUN.
The routine below does just that.It also compensates for extra spaces between the words.

```
 150 V$="":W$="":VB=0:OB=0
 160 IF MID$(Q$,I,1)=" " AND V$=""THEN V$=LEFT$(Q$,I,-1)
 170 If MID$(Q$,I+1,1)<>"    " AND V$<>"" THEN
W$=MID$(Q$,I+1,LEN(Q$)-1):I=LEN (Q$)
 180 NEXT I
```

V$ contains the VERB and W$ contains the NOUN.
Now we must check that the VERB and NOUN are part of the VERB and NOUN list.

```
 190 IF W$="" THEN V$=Q$
 200 FOR I=1 TO 15
 210 IF V$=V$(I) THEN VB=I
 220 NEXT
 230 FOR I=1 TO 11
 240 IF W$=O$(I) THEN OB=I
 250 NEXT
```

VZ ASSEMBLY

Last month we looked at how the Operating System classifies variables, and at some useful routines in ROM for carrying out mathematical operations, namely addition, subtraction, multiplication and division. This month we move on to more complicated functions. The operations we're considering now would be far too involved to be included in the average assembly language program, unlike the previous routines which could have been written out in full within the main program.

The math routines supported by Microsoft are SINE, COSINE, ARCTANGENT, TANGENT, SQUARE ROOT, EXPONENTIAL (base e) and NATURAL LOG. You may wonder what use these are to the average programmer, but if one considers such games as golf, tennis etc (where angles of shots, distances etc must be calculated), and many scientific applications, it soon becomes obvious that the Assembly Language programmer needs to be able to handle such functions efficiently.

Since none of these functions are linear, the system must either look up a table, which contains corresponding functions, or use some method of approximation. Since using tables would chew up a lot of memory, and be very limited in the number of values that could be handled, the later system is used.

Mathematicians will notice that three of these functions can be obtained from special identities involving the other four. These are COS (SIN+PI/2), TAN (SIN/COS) and sqare root (e^(ln(x)/2)). The fundamental functions, namely SIN, EXPONENTIAL, ARCTANGENT and NATURAL LOG are calculated by the use of mathematical series. These are beyond the scope of this article - if you want to know more, consult a good book on maths or Microsoft. All that we need to understand is where the routines are located and how to use them.

All the routines listed here assume that the value for which the function is to be calculated is stored in WRA1 and that the data type (integer, single or double precision) is indicated by the value in location 78AFH (see last month). In each case, we study the operation of the routine, and give an example of its use, where appropriate.

SIN - CALL 1547H : Calculates the SIN of the value in WRA1. The value must be in radians. The result is a single precision value left in WRA1. For those unfamiliar with radians, here's a simple way to convert from degrees to radians : Divide the number of degrees by 180, then multiply the result by pi (=3.142). This gives the result 1 degree = 0.0175 rad.

Here's an example of how to calculate the SIN of 30 degrees:

```
        LD A,4          ;code for single prec.
        LD (78AFH),A    ;store to mode flag
```

```
             LD HL,ANGL      ;pt HL to table containing angle
             CALL 09B1H      ;& store in WRA1
             CALL 1547H      ;calculate sin
                             ;result in WRA1
        ANGL DEFB 18H        ;30 degrees in radians
             DEFB 04H        ;  =0.5235 rad
             DEFB 06H
             DEFB 80H        ;exponent
```

COSINE - CALL 1541H : Calculates the cos of a floating point value. Result will be returned in WRA1 as a floating point value. No example is given since this routine is the same as sin; just change the CALL.

TANGENT - CALL 15A8H : Calculates tan of a single precision value in WRA1. Result left in WRA1. Example is as for sin.

ARCTANGENT - CALL 15BDH : Calculates the arctangent of the floating point value in WRA1. Result left in WRA1. Example as for sin.

SQUARE ROOT - CALL 13E7H : Calculates the square root of any value in WRA1, leaving single precision result there. Example:

```
             LD A,4          ;set single prec.
             LD (78AFH),A
             LD HL,VAL       ;pt HL to value in table
             CALL 09B1H      ;transfer value to WRA1
             CALL 13E7H      ;calculate square root
                             ;result in WRA1
         VAL DEFB 00         ; 4
             DEFB 00
             DEFB 00
             DEFB 83H        ;exponent of 4
```

EXPONENTIAL (raise natural base) - CALL 1439H : Raises the natural base 'e' to the single precision power in WRA1. Result is left there. Example as for square root.

NATURAL LOG - CALL 0809H : Calculates natural log of single precision value in WRA1. Single Precision result is left in WRA1. Example as for square root.

Apart from the functions above, the system supports some other mathematical operations. These are listed below:

ABSOLUTE VALUE - CALL 0977H : Converts the value in WRA1 to its positive equivalent, leaving the result in WRA1. If a negative integer greater than 2^15 is encountered, it is converted to a single precision value, and the data type indicator updated accordingly.

RETRUN INTEGER - CALL 0B37H : Returns the integer portion of a floating point number in WRA1 to WRA1. If the value is positive, the integer portion is returned. If negative, it is rounded up before truncation. The mode flag is updated.

RAISE X TO POWER Y (X^Y) - JP 13F2H : Raises the single precision value which has been saved on the stack to the power specified in WRA1, the redult being left in WRA1. For example, to raise 16 to the power 2 (16^2):

```
              LD BC,RETAD     ;return address after raisng
              PUSH BC         ;is placed on stack
              LD A,4          ;code for single precision
              LD (78AFH),A    ;set mode flag
              LD HL,X         ;pt HL to location of value to be raised
              CALL 09B1H      ;move to WRA1
              CALL 09A4H      ;move WRA1 to stack
              LD HL,Y         ;address of power
              CALL 09B1H      ;move to WRA1
              JP 13F2H        ;calculate x^y
       RETAD                  ;returns here after calculation

       X      DEFW 00         ;single prec. for 16
              DEFW 85H
       Y      DEFW 00         ;single prec for 2
              DEFW 82H
```

Notice that the routine cannot be CALLed since this would place a return address on the stack, which would be mistaken as the value of X.

RANDOM NUMBER GENERATOR - CALL 14C9H : Generates a random number between 0 and 1, or 1 and n depending on the value in WRA1. The integer result is left in WRA1 with the mode flag set. For a radom number between 0 and 1, load WRA1 with 0, for a value greater than 0, a range of random numbers is specified. For example:

```
              LD A,2          ;type code for integer
              LD (78AFH),A    ;set mode flag
              LD A,50         ;put an integer 50
              LD (31009),A    ;into WRA 1
              CALL 14C9H      ;get a random number between 1 and 50
              LD HL,(31009)   ;put it in HL
```

There are still more routines available to the Assembly Language programmer, but we've listed the main ones here. For further information, consult a good book on Microsoft BASIC, such as 'Microsoft Basic Decoded and other Mysteries', by J.L.Farvour (from which much of this information was taken). Note that the VZ uses a modified version of the Level II Operating system, and it is always wise to check that any particular routine still resides at the same address in the VZ ROM. Many other routines are listed in the VZ300 Technical Manual. However, there still remains one area of importance - displaying data on the text screen. How, for example, do we display the contents of some particular register on the screen? Level II offers three main routines to overcome this problem, by converting binary values into ASCII representation.

DISPLAY HL ON SCREEN - CALL 0FAFH : Displays the contents of HL at the current cursor position

INTEGER TO ASCII - CALL 132FH : Converts the integer in WRA1 to a string of ASCII characters and stores them in a buffer pointed to by HL. Unfortunately, due to the unusual offset incorporated into the VZ VDU, this buffer cannot be simply defined as screen memory. To display the value on the screen, the ASCII must first be sent to a buffer within the program, then dumped to the screen using the display message routine at 28A7H. Also, the B & C registers must be loaded with 5 to avoid any punctuation in the ASCII. For example, to display WRA1:

```
          LD BC,505H      ;set B=C=5
          LD HL,BUFF      ;pt HL to buffer
          CALL 132FH      ;convert WRA1 to ASCII string
          LD HL,BUFF      ;pt HL to ASCII
          CALL 28A7H      ;& display at cursor pos.
     BUFF DEFS 5          ;reserve space for ASCII here
```

FLOATING TO ASCII - CALL 0FC1H : Converts the single or double precision value in WRA1 to an ASCII string, which is stored in a buffer pointed to by HL. As the value is converted to ASCII, it is formatted as if PRINT USING was used. The possible formats are selected by loading the following values into A, B and C:

Register A=0 Do not edit. Strictly binary to ASCII.
A=x where each bit of x is set according to:

Bit:
0 - Exponential notation
1 - Reserved
2 - Sign following value
3 - Sign before value
4 - Print leading $ sign
5 - Include leading asterisks
6 - Print commas every 3rd digit
7 - 0 for no editing, 1 to enable options above.

Register B = No. of digits left of decimal point
Register C = No. of digits to right of decimal point

Now that we've covered most of the ROM routines, we're ready to start on a sample Assembly Language Program assignment next month.

No 91

Dick Smith Electronics

Technical Bulletin

PROGRAMMING FOR THE VZ-200 COMPUTER'S JOYSTICKS

The VZ-200's optional joysticks are interfaced and software scanned in a similar fashion to the main keyboard. The interface occupies port addresses 20H to 2FH, and the joystick switches are connected in an array whose row lines are connected to port address lines A0 to A3. This effectively places the switches at the following bit positions and addresses:

			BIT POSITION					
			4	3	2	1	0	
PORT ADDRESS	2EH	(46D)	ARM	→	←	↓	↑	RIGHT JOYSTICK
	2DH	(45D)	FIRE					
	2BH	(43D)	ARM	→	←	↓	↑	LEFT JOYSTICK
	27H	(39D)	FIRE					
	20H	(32D)	TEST OVERALL JOYSTICK STATUS					

Note that the port addresses shown above are those which cause the joystick row concerned to go low -- i.e., to logic 0. The first four addresses cause only the row concerned to go low, to test just that row, while the fifth address pulls all four rows low simultaneously, to allow a quick check of overall joystick status. In each case, if a joystick is moved from its rest position, one or more bit lines will be pulled low (logic 0). If the joystick is in its rest position, all bit lines will remain high (logic 1).

The status of the joysticks is easy to determine from within a program, both in BASIC and assembly language. In BASIC the easiest way is to use the INP command with AND 31 (or AND 16) to mask off the unused data bits, then testing for the bit(s) pulled low, as shown by the first example:

FOR SALE

ORIGINAL VZ PROGRAM TAPES
AND SMALL RANGE P.C.B,S

REASONABLE RANGE OF VZ TAPES AT AUST.$5.00 EACH
SMALL RANGE OF P.C.B'S ALSO AVAILABLE (P.O.A)

SEND SELF ADDRESSED/STAMPED ENVELOPE FOR
A LIST OF TAPES AND P.C.B'S TO:-

DAVE BOYCE
41 HEATHER DRIVE
CHRISTIE DOWNS 5146
SOUTH AUSTRALIA

PLEASE INCLUDE FULL NAME AND ADDRESS

```
5  R$="RIGHT JOYSTICK: ":L$="LEFT JOYSTICK:
10 A=INP(32)AND31:IFA=31THEN10:REM WAIT FOR SOME ACTION
20 A=INP(46)AND31:IFA=31THEN100:REM CHECK FIRST ROW
30 IFA=26THEN PRINT R$+"LEFT+UP":GOTO200
32 IFA=25THEN PRINT R$+"LEFT+DOWN":GOTO200
34 IFA=22THEN PRINT R$+"RIGHT+UP":GOTO200
36 IFA=21THEN PRINT R$+"RIGHT+DOWN":GOTO200
40 IFA=30THEN PRINT R$+"UP":GOTO200
50 IFA=29THEN PRINT R$+"DOWN":GOTO200
60 IFA=27THEN PRINT R$+"LEFT":GOTO200
70 IFA=23THEN PRINT R$+"RIGHT":GOTO200
80 IFA=15THEN PRINT R$+"ARM":GOTO200
100 A=INP(45)AND16: REM NOW CHECK SECOND ROW
110 IFA=0THEN PRINT R$+"FIRE":GOTO200
120 A=INP(43)AND31:IFA=31THEN190:REM CHECK 3RD ROW
130 IFA=26THEN PRINT L$+"LEFT+UP":GOTO200
132 IFA=25THEN PRINT L$+"LEFT+DOWN":GOTO200
134 IFA=22THEN PRINT L$+"RIGHT+UP":GOTO200
136 IFA=21THEN PRINT L$+"RIGHT+DOWN":GOTO200
140 IFA=30THEN PRINT L$+"UP":GOTO200
150 IFA=29THEN PRINT L$+"DOWN":GOTO200
160 IFA=27THEN PRINT L$+"LEFT":GOTO200
170 IFA=23THEN PRINT L$+"RIGHT":GOTO200
180 IFA=15THEN PRINT L$+"ARM":GOTO200
190 A=INP(39)AND16: REM CHECK FOURTH ROW
195 IFA=0THEN PRINT L$+"FIRE"
200 FOR I=1TO300:NEXTI:GOTO10
```

In assembly language programs it is even easier to read the joystick status. Here is a sample subroutine which reads the status of both joysticks and returns with the results in the B and C registers. Note that in each case the appropriate bit is set to logic 1 if that joystick switch is enabled, except that the two 'FIRE' switches are transferred to bit 5.

I.e., the bit assignment becomes:

BIT	5	4	3	2	1	0
SWITCH	FIRE	ARM	→	←	↓	↑

```
JOYSTK  IN   A,(2EH)      ;read 1st row
        OR   0E0H         ;set bits 5-7 high
        CPL               ;then complement to invert
        LD   B,A          ;& save in B reg
        IN   A,(2DH)      ;read 2nd row
        BIT  4,A          ;check for FIRE pressed
        JR   NZ,JOYST1    ;skip if not
        SET  5,B          ;otherwise set bit 5
JOYST1  IN   A,(2BH)      ;read 3rd row
        OR   0E0H         ;& process as above
        CPL
        LD   C,A          ;except save in C reg
        IN   A,(27H)      ;read 4th row
        BIT  4,A          ;check for FIRE pressed
        RET  NZ           ;return if not
        SET  5,C          ;otherwise set bit 5
        RET               ;& then leave
```

I hope this information is enough to allow you to program the joysticks with confidence.

SLOT MACHINE

BY KEITH WRIGHT

```
1 CLS
2 DIMA(6):DIMR$(6)
7 PRINT@72,"$$LOT MACHINE
10 PRINT:PRINT:PRINT:PRINT:PRINT"  PAPARATA SOFTWARE CO.18/2/86
12 PRINT@480,"DO YOU HAVE A PRINTER ?(Y/N)";:GOSUB1010
13 HC=0:IF A$="Y",CLS:HC=1:GOTO15 ELSE IFA$<>"N",RUN
15 SOUNDO,9:SOUNDO,9:PRINT@448,"DO YOU NEED INSTRUCTIONS?(Y/N)"
20 GOSUB 1010
30 IF A$="Y"THEN50ELSEIFA$="N"THEN150
40 PRINT"             Y OR N ONLY!":GOTO20
50 CLS:PRINT"THE RULES ARE VERY SIMPLE,TYPE"
55 PRINT"'H' TO PULL THE HANDLE.IT WILL"
60 PRINT"COST YOU $1.00 A TIME TO PLAY."
65 PRINT:PRINT"  WINNING COMBINATIONS ARE:-"
70 PRINT"WHEEL 1 APPLE,$1.00"
75 PRINT"WHEEL 1 & 2 APPLES,$3.00"
80 PRINT"WHEEL 3 A CHERRY AND 1&2 THE"
85 PRINT"SAME (BUT NOT APPLES),$5.00"
90 PRINT"WHEELS 1,2&3 THE SAME,(BUT NOT"
95 PRINT"CHERRIES),$15.00"
100 PRINT"   THREE CHERRIES = JACKPOT
105 PRINT"      YOU GET  $90.00
107 IFHC=1,COPY ELSE 110
108 LPRINTCHR$(14);"  YOU GET  $90.00
110 GOSUB 1000
120 IFA$<>"H"THEN 110
150 CLS:PRINT"ALL READY TO PLAY ??"
160 PRINT"WHAT AMOUNT WOULD YOU LIKE AS A"
170 INPUT"BANKROLL TO START WITH";B
180 IF SGN(B)<1 THEN CLS:PRINT"SORRY,NO CREDIT,":GOTO170
190 IF B>97 THEN CLS:PRINT"THAT'S OVER THE HOUSE LIMIT":GOTO160
195 SB=B
200 DATA "APPLE","ORANGE","BANANA","LEMON","CHERRY"
210 RESTORE:FOR N=1 TO 5:READ R$(N):NEXT
220 PRINT:PRINT"YOUR BALANCE IS $";B:IF B<1THEN 900
230 GOSUB 1000
240 IF A$ ="H"OR A$="Q"THEN 250 ELSE GOTO 230
250 IF A$="Q"THEN 1100 ELSE RESTORE:B=B-1
260 FOR N = 1 TO 3 : A(N) = RND(5) : NEXT
300 CLS:PRINT @ 224,R$(A(1));TAB(10)R$(A(2));TAB(20)R$(A(3))
310 IF A(1)=A(2) AND A(2)=A(3) AND A(3)=5 THEN B=B+90:GOTO 850
320 IF A(1)=A(2) AND A(2)=A(3) THEN B=B+15:GOTO 900
330 IF A(3)=5 AND A(1)=A(2) AND A(2)<>1 THEN B=B+5:GOTO 900
340 IF A(1)=A(2) AND A(2)=1 THEN B=B+3:GOTO 900
350 IF A(1)=1 THEN B=B+1
360 GOTO220
800 STOP
```

CONTINUED ON NEXT PAGE

```
850 PRINT:PRINT"***JACKPOT***":FOR N=23 TO 31 STEP 2:SOUND N,1:NEXT
900 IF B<1 THEN PRINT" B U S T E D ! !":GOTO 950
905 SOUND27,3:SOUND31,7:PRINT"YOUR TOTAL NOW =";
907 PRINT USING"$$###.##";B
910 IF B>200 THEN PRINT"THE BANK IS WORRIED."ELSE 230
920 IF B>499 THEN PRINT" V E R Y  WORRIED !"
940 GOTO230
950 PRINT "DO YOU WANT A NEW GAME(Y/N)"
960 A$=INKEY$:A$=INKEY$:IF A$=""THEN 960
970 IF INKEY$<>""THEN 970
980 IF A$="N"THEN 995
990 IF A$<>"Y"THEN 960 ELSE CLS:GOTO 150
995 RUN"MENU"
999 STOP
1000 PRINT@448,"'H' TO CONTINUE,'Q'TO QUIT-----"
1010 A$=INKEY$:A$=INKEY$:IF A$="" THEN 1010
1020 SOUND 29,1:IF INKEY$<>""THEN 1020
1030 RETURN
1100 CLS:PRINT @480,"YOUR INITIAL FUND IN THIS GAME"
1110 PRINT"WAS";:PRINTUSING"$$#.##";SB;:PRINT", IT IS NOW";
1120 PRINTUSING"$$####.##";B
1130 PRINT TAB(4);"I NOTE YOU HAVEN'T PAID ME"
1140 PRINT"THE";:PRINTUSING"$$#.##";SB;:PRINT"YET !!"
1150 IF B>SB THEN 1250
1160 A$="================================"
1170 PRINT A$:PRINT "DR. TO VZ300 & CO.,":PRINT A$
1175 IF HC=1,LPRINT A$:LPRINT "DR. TO VZ300 & CO.,":LPRINT A$
1180 PRINT:PRINT"BALANCE NOW OWING:--";
1185 IFHC=1,LPRINT:LPRINT"BALANCE NOW OWING:--";
1190 PRINTUSING"$$##.##";SB-B
1195 IFHC=1,LPRINTCHR$(19);:LPRINTUSING"$$##.##";SB-B
1200 PRINT"                     ----------",A$
1205 IFHC=1,LPRINT"                     ----------";CHR$(10);A$
1210 PRINT" SO COUGH UP P.D.Q."
1215 IFHC=1,FORN=1TO5:LPRINT:NEXT
1220 GOTO 950
1250 A$="================================"
1260 PRINT"SO HERE'S YOUR CHEQUE:-"
1270 PRINT A$:PRINT" THE MEMORY BANK OF VZ300"
1280 PRINT TAB(20);"STAMP DUTY"
1290 PRINT"PAY TO BEARER:-          PAID"
1300 B$(0)="ZERO":B$(1)="ONE":B$(2)="TWO":B$(3)="THREE"
1305 B$(4)="FOUR":B$(5)="FIVE":B$(6)="SIX":B$(7)="SEVEN"
1310 B$(8)="EIGHT":B$(9)="NINE"
1320 BA$=STR$(B-SB)
1330 FOR AM=1 TO LEN(BA$)
1340 PA=VAL(MID$(BA$,AM,1))
1350 PRINT B$(PA);" ";
1360 NEXT
1370 PRINT"DOLLARS & NO CENTS"
1380 PRINT TAB(15);:PRINTUSING"**$###.##";(B-SB)
1390 PRINT A$
1395 IFHC=1,COPY
1400 GOTO 950
```

NEWSLETTER INDEX
PART 1-ISSUE 1 TO 15

AK VZ

SOFTWARE FOR SALE

We now have the following software available for sale.
All prices shown include postage and packing anywhere in New Zealand.
Please make sure that the memory of your VZ is greater than or equal to the memory required by the software that you wish to purchase.
Memory sizes indicated do not include video ram.

ALL PRICES ARE SHOWN IN NEW ZEALAND DOLLARS.

BLACKJACK ROYALE	16K	$15.00
CASTLE GREYSTONE	32K	$15.00
DATA	16K	$18.00
FBI-2001	31K	$18.00
GALACTIC EMPIRES	16K	$15.00
GRAPHSTAR	22K	$24.00
HAUNTED MANSION	22K	$15.00
KNIGHTS QUEST	32K	$18.00
NAMESTAR/SHOPSTAR	22K	$15.00
SCOTLAND YARD	16K	$ 8.00
TRIVIAL CULT	16K	$ 8.00
VZ MONOPOLY V2.0	22K	$18.00

PLEASE MAKE ALL CHEQUES AND MONEY ORDERS PAYABLE TO-

AUCKLAND VZ 300/200 COMPUTER CLUB.

VZ DISASSEMBLER FOR SALE

This Disassembler is written by Peter Hickman.
And is also distributed as the "Laserlink Disassembler".

Peter is offering this very fine program to user group members for NZ$24.00.
All orders can be placed through this club,as peter has offered a cheaper price for bulk purchases.

The program is available on tape or disk.And comes in a 4 program version and
You choose which section you use depending on what memory you have.

ALL THESE PROGRAMS CAN BE DEMONSTRATED AT THE WORKSHOP IF REQUESTED