February 1995 By Leslie Milburn Issue #3

ArchiVZ

Editorial

Well, hello again. The time has certainly passed by quickly since the last issue. I have managed to get in contact with quite a few more VZ users since Issue 2 went out and that is where most of my time has been spent. I apologise if I took a while to respond to your enquires and it is a good feeling to know that I am now just about up to date.

You will notice a few changes to the format of the newsletter. These were at YOUR request, so I hope it meets your needs. The Technical Library section, once up to date, will now show new additions only. A full list will be provided upon request. The BASIC Programming section has been changed slightly as well.

Although I have had a lot of letters & requests, nobody mentioned the Quick Reference Card! Was it too Technical??

Enough of my blabbing, until next issue.....

Leslie.

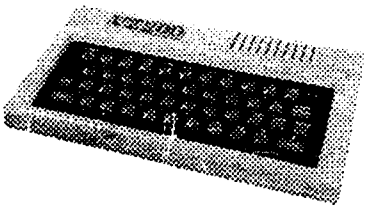
The Technical Library

Below is a list of New Additions to the Library. A Full list of all items will be provided upon request, just send a Self Addressed **Stamped** envelope.

Magazines *(Photocopies of articles only provided)* :-

Magazine	Issues
Alternative Computing	#3
Byte	Aug '87, Sept '87, Oct '87, Nov '87, Dec '87,Jan '88, Feb '88, Mar '88, Apr '88
Commodore Magazine	Aug '84, Feb '85
The Computer Journal	#66, #67
Computing with the Amstrad	March '87
Electronics Australia: Op Amps	-
Electronics in Action	May '94, July '94
Electronics Today International (UK)	May '94 - Jan '95
Elektor Electronics	#181, #221, #222, #223
ETI Circuit Techniques	#2
Everyday with Practical Electronics	April '94, August '94
The Home Computer Course (Plus Dictionary of Terms)	#1 - #4, #6 - #19, #23, #24
The Home Computer Advanced Course	#1
MicroComputer Journal	May/June '94 - Jan/Feb '95
Silicon Chip	All issues to Feb '95
SOFTEX (For Texas Instruments 99/4A)	Vol 1 #1 - Vol 1 #6
Your Computer	Apr '86, Jan '88, Oct '91, Nov '91, Feb '92, Mar '92, Oct '92, Nov '92 , Tech Tips #2

OUR TRUSTY VZ



VZ User Group Newsletters *(Photocopies only provided)* :-

NewsLetter	Issue	Notes
DSE VZ200 Interface	#1	A User in SA has informed me that there were about 7 or 8 issues of this newsletter sent out. I am investigating this claim.
J & R Software	Vol 1 #3	Does anybody know anything about this company as they produced some very good software & hardware. It was run by two people:- Jane & Ronald Rohde (hence J & R) of Walsh Street, South Yarra, Victoria 3141.
Christchurch VZ User Group	ALL issues **except** Oct '84, Nov '84, Oct '85, Nov '85, July '86	Thanks to a contact in New Zealand I have managed to get original copies of most of the newsletters for this User Group. However, we are still missing some.

INSIDE

Wanted for the Library

Below is a list of those magazines which are either missing from the **T.L.** or a damaged copy is held. If you have any of these which you don't want, let me know.

Name	Issue/Volume
	Needed
PCG	19 July 1984 Vol 1 #1
PCG	Aug 1984 Vol 1 #2
PCG	Oct 1984 Vol 1 #3
PCG	Dec 1984 Vol 1 #5
PCG	Feb 1985
AEM	Nov 1986
AEM	Dec 1987
AEM	Aug 1988
AEM	Dec 1988 onwards.
EA	Pre June 1976
EA	Jan, March 1977
EA	Aug, Sept 1979
EA	Nov 1989
EA	May, Aug 1990
EA	93 (except April) & onwards
ETI	Pre June 1976 (except November 1974)
ETI	Mar, May, Jun 1977
ETI	July, Aug, Sept 1979
ETI	April, Nov 1988
ETI	Mar, April, May, Oct 1989
ETI	Feb, Mar, May 1990

Name	Issue/Volume
	Needed
APC	Vol 1 issue 1 - Vol 1 issue 7 (inclusive)
APC	Dec 1989
APC	Jan 1990
APC	Nov 1993 and onwards.
YC	Sept 1983
YC	Jun, Aug - Oct 1984
YC	Jan, July, Nov, Dec 1985
YC	Mar - July, Sept - Nov 1986
YC	Jan - Mar, May - Oct 1987
YC	Feb, Apr - May, Aug, Sept, Nov, Dec 1988
YC	Jan, Apr 1989
YC	Jun - Sept, Dec 1990
YC	Jan - Mar, May 1991
YC	Apr - Oct 1992
YC	All 1993 and onwards.
M80	Vol 4 #2 - Vol 4 #6.
M80	Vol 4 #8 onwards.
CFG	#1, #3 onwards
	Damaged
EA	Nov 1984
EA	Dec 1988
AEM	July, Aug, Sept, Oct, Nov 1985
AEM	Jan, Feb, Apr, May, June 1986
YC	Dec 1987

Trading Post

This section is dedicated to the VeeZedders who wish to sell their setup. I shall only list their name, contact number, hardware list and what price they want for it. In some cases some of this information may be missing.

Note:

The following list contains summary information only. If you want further details then contact the person concerned.
Also Please remember that **VZ Hardware** is extremely difficult to get nowadays so I suggest that you snap up some of these bargains before they disappear (such as the rubbish tip - yes it could get to that!!)
Please also refer to **issues #1 & #2** for other items for trading. I will print an updated list next issue.

ITEMS FOR SALE:-

Name	Contact Number	Price	Hardware List
Les Brennan	(07) 378 3402	Make an offer.	300 Baud Modem suits the VZ.
Roger Hall	(052) 79 5905	His prices include postage.	Plotter Paper - $3.80 VZ300 - $25.00 Broken Plotter (parts only) - $25.00 VZ200 Further Programming - $4.00 VZ200 First Book of Progs - $4.00 VZ200 Technical Ref - $4.00 VZ300 Technical Ref - $4.00 Some LE'VZ Mags (8) - $3.00 Various Software from $2.50

David Lancaster	(03) 748 1408 5:00pm - 8:00 pm weekdays All Day Weekends	-	Serial Interface and Cable - $20 Joysticks - $10 16K VZ200 RAM - $10 SANYO Monitor (mono) - $25 VZ300 8 Bit Port - $15
Kevin Linsell	(049) 54 8189	-	VZ200 (1), VZ300 (1) - $50 each 2 Disk Controllers - $30 each Disk Drive - $50 Printer Interface - $20 16K RAM - $30 VZ Printer Plotter - $50 Joysticks - $20 Datasette - $30 Light Pen - $20 System 80 Computer - $50 Various Software from $5 Various Books from $5
Albert Fischer	(02) 681 5912 after 7:00 pm	Best Offer for the lot New Condition	VZ200, 16K RAM, Joysticks, Tape Unit, Printer/Plotter Heaps of Software All its Books Some User Group Mags.
Tony Page	95 Gregory Street Greystanes NSW 2145	$250.00 for the lot Wants it picked up from his place.	VZ200 (1), VZ300 (2) 1 Disk Controller 1 Disk Drive 2 or 3 Printer Interfaces 2 or 3 16K RAM's VZ Printer/Plotter VZ RTTY Interface Various Software & Books
David Schouten	10 Bargoo Place Erskine Park NSW 2759	$50.00 for the lot	VZ200 (needs repair), Printer Interface, 16K RAM, VZ Printer/Plotter Datasette, Some Software
Tony Vandenbelt	(03) 725 9181	$300.00 for the lot Wants it picked up from his place.	VZ300 VZ300 (2 chips missing) Disk Drive Disk Controller TP40 Plotter plus spare pens Power Supply & Pre-Regulator 16K RAM (3) Terminal Driver (Home Made) ASCII Baudot Driver (Home Made) Printer Driver (Home Made) DSE RTTY Interface DSE Terminal Driver (Board & Memory only) ETI 1611 EPROM Programmer (Not Working) 1 VZ200 Board Only Some books and articles

ITEMS WANTED:-

Name	Contact Address	Price	Hardware List
Jason Oakley	142 BroadMeadow Road Broadmeadow 2292	Will pay $10 - $15 Will pay $15 - $20	For Cheap VZ200's For Cheap VZ300's

Buyers: If you buy any items from the above people could you please let me know so that I can update the list.

Sellers: If you do sell any of your equipment can you pass their name and address on to me so that I can keep in contact with as many VZ Users as possible.

BASIC Programming

As I mentioned in the Editorial, I have decided to change the direction of this section slightly. While I will still present programs which will be able to run on an Unexpanded VZ300 (VZ200 + 16K RAM), I will not be presenting a "solution". This proved to be too time consuming. Instead I will make comments where necessary. I will also provide a brief paragraph describing code changes needed to compile the program using the LASCOM BASIC Compiler (refer Issue #2).
In an attempt to keep all users interested, I will print two programs, one being a utility or more practical program, the other a game or novelty program. Read on if I've still got your attention.

We've probably all seen the National Australia Bank's advert on TV where a young couple go in to get a Home Loan and they are shown some figures on Bank's Computer. Well here are a couple of Programs which provide the same sort of information. They are easy to type in and will work with all VZ regardless of Memory Expansions etc. I hope that you find these of practical use, and show you that the VZ is indeed a useful tool, and also help you to control you Home Loan Repayments in these fairly tough times.

Program #1(a) - Mortgage Amount Projections

This program will project monthly mortgage payments for any amount, interest rate, or mortgage term.

----------Start of Program Listing----------

```
10    REM MORTGAGE COMPUTATION PROGRAM - BASIC
20    REM SOURCE: USING THE TRS80 IN YOUR HOME
25    REM BY: CHARLES D. STERNBERG
30    REM **** DATA INITIALISATION ****
40    PRINT "ENTER THE MORTGAGE AMOUNT"
50    INPUT P
60    PRINT "ENTER THE INTEREST RATE"
70    INPUT I1
80    IF I1 > 1 THEN 100
90    I1 = I1*100
100   I = (I1/100)/12
110   PRINT "ENTER THE YEARS OF THE MORTGAGE"
120   INPUT Y
130   PRINT
140   PRINT
150   PRINT
160   REM **** COMPUTATION ****
170   M = I / ( (1+I) ^ (Y*12) - 1) + I
180   M1=M * P
190   PRINT "*****************************"
200   PRINT "MORTGAGE AMOUNT $";P
210   PRINT "INTEREST RATE";I1;"%"
220   PRINT "MONTHLY PAYMENT $";M1
230   PRINT "*****************************"
240   REM **** PROGRAM TERMINATION ****
250   PRINT
260   PRINT
270   END
```

----------End Of Program Listing----------

Main Variables used in Program Above.

Variable	Description
P	Projected Mortgage Amount
I1	Interest Rate
I	Modified Interest Rate
Y	No of Years for the Mortgage
M1	Monthly Payment

Program #1(b) - Mortgage Comparisons

The second program I have provided produces a table that compares various mortgage amounts, terms and interest rates.

----------Start of Program Listing----------

```
10    REM MORTGAGE COMPARISON PROGRAM
20    REM SOURCE: USING THE TRS80 IN YOUR HOME
30    REM BY: CHARLES D. STERNBERG
40    REM NOTE: ROUNDING ERRORS MAY OCCUR IN COMPUTED NUMBERS
50    REM **** DATA INITIALISATION ****
60    PRINT "ENTER THE ITEM TO VARY-AMOUNT(A), INT RATE(I), OR YEARS(Y)"
70    S1 = 1
80    S2 = 1
90    S3 = 1
100   INPUT A$
110   REM **** ENTRY OF VARIABLE ITEMS ****
120   IF A$ <> "A" THEN 180
130   PRINT "ENTER THE BEGINNING AMOUNT, ENDING AMOUNT TO CONSIDER"
140   INPUT A0,A1
150   PRINT "ENTER THE INCREMENT EG. $1000"
160   INPUT S1
170   GOTO 340
180   IF A$ <> "I" THEN 240
190   PRINT "ENTER THE LOWEST, HIGHEST INTEREST RATE TO CONSIDER"
200   INPUT R0,R1
210   PRINT "ENTER THE INCREMENTS (0.25 FOR 1/4%)"
220   INPUT S2
230   GOTO 300
240   IF A$ <> "Y" THEN 290
250   PRINT "ENTER THE LOWEST, HIGHEST NUMBER OF YEARS TO CONSIDER"
260   INPUT Y0,Y1
270   PRINT "ENTER THE INCREMENT"
280   INPUT S3
290   REM **** ENTRY OF CONSTANT ITEMS"
300   PRINT "ENTER THE MORTGAGE AMOUNT"
310   INPUT P
320   A0 = P
330   IF A$ = "I" THEN 400
340   PRINT "ENTER THE INTEREST RATE"
350   INPUT I1
360   IF I1 >= 1 THEN 380
370   I1 = I1 * 100
380   R0 = I1
390   IF A$ = "Y" THEN 430
400   PRINT "ENTER THE YEARS OF THE MORTGAGE"
410   INPUT Y
420   Y0 = Y
430   PRINT
440   PRINT
450   PRINT
460   REM ****
470   REM **** PROCESSING LOOP ****
480   FOR Y = Y0 TO T1 STEP S3
490   PRINT "FOR A MORTGAGE OF";Y;"YEARS"
500   PRINT
510   FOR I1 = R0 TO R1 STEP S2
520   PRINT "USING INTEREST RATE";I1;"%"
530   PRINT
540   PRINT "MORTGAGE";TAB(15);"MONTHLY PI";TAB(30);" TOTAL"
550   PRINT " AMOUNT";TAB(15);" PAYMENT";TAB(30);"INTEREST"
560   PRINT "--------";TAB(15);"---------";TAB(30);"---------"
570   FOR P = A0 TO A1 STEP S1
580   REM **** COMPUTATION AND PRINT ****
590   I = ( I1 / 100 ) / 12
600   M = I / ( (1 + I) ^ (Y * 12) - 1) + I
610   M1 = M * P
620   I3 = M1 * Y * 12 - P
630   PRINT P;TAB(15);M1;TAB(30);I3
640   NEXT P
650   PRINT "**********************************"
660   PRINT
670   NEXT I1
680   NEXT Y
690   END
```

----------End Of Program Listing----------

Main Variables used in Program Above.

Variable	Description
S1	Increment of Mortgage Amount
S2	Increment of Interest Rate
S3	Increment of Mortgage Years
A0	First Amount Considered
A1	Last Amount Considered
R0	Lowest Rate Considered
R1	Highest Rate Considered
Y0	Lowest No of Years Considered
Y1	Highest No of Years Considered
P	Single Mortgage Amount
I1	Single Interest Rate
Y	Single Year to Consider
M1	Monthly Payment Computed
I3	Total Interest Paid

Program #2 - Treasure

```
5     REM TREASURE
10    REM SOURCE: 33 CHALLENGING GAMES FOR TRS80/APPLE/PET
12    REM BY: DAVID CHANCE
15    CLS: B = 0: PRINT
30    PRINTTAB(20);"T R E A S U R E"
40    PRINT: IF R = 1 THEN 130
50    PRINT "INSTRUCTIONS: YOU ARE LOOKING FOR A LOST TREASURE."
60    PRINT "TO FIND THAT TREASURE YOU'LL HAVE TO OVERCOME SOME"
70    PRINT "OBSTACLES. TO OVERCOME THESE OBSTACLES ALL YOU HAVE"
80    PRINT "TO DO IS MOVE AROUND THEM. ENTERING A NUMBER FROM"
90    PRINT "0 TO -5 WILL MOVE YOU BACK, LEFT, RIGHT OR"
95    PRINT "STRAIGHT AHEAD SLOW."
100   PRINT "ENTERING A NUMBER FROM 0 TO 5 WILL MOVE YOU"
110   PRINT "STRAIGHT AHEAD."
130   REM SET AMOUNT OF OBSTACLES-GET CORRECT MOVE
140   A2 = 8: A3 = A2 - 7
150   GOSUB 770
155   IF R = 2 THEN FOR T = 1 TO 1000: NEXT: GOTO 190
170   PRINT: INPUT "TO PROCEED PRESS RETURN";X$
180   CLS: PRINT
190   PRINT "AFTER YOU INPUT YOUR MOVE, THE COMPUTER"
200   PRINT "WILL OUTPUT YOUR LOCATION (MAYBE)."
205   D = A + 15 * RND(0): IF D < 9 THEN 205: REM DISTANCE TO TREASURE
210   PRINT "INPUT YOUR MOVE";
220   INPUT M
230   IF M < -5 OR M > 5 THEN PRINT "DUMMY!! CAN'T YOU REMEMBER THE NUMBERS???":GOTO 220
240   B = B + 1
250   IF ABS(M - A1) > 1 AND ABS(M - A) > 1 THEN 265
260   GOSUB 280: GOTO 300: REM DECREASE DISTANCE - OUTPUT MESSAGE
265   IF A3 = 9 THEN 890
270   PRINT: PRINT "THAT DIDN'T WORK !!"
275   GOTO 320
280   IF M = 0 THEN GOTO 300
290   M1 = M + 0.5: D = D - M1: RETURN
300   IF D <= 0 THEN 830
310   IF D > 0 THEN 780
320   REM OBSTACLES
330   IF A3 > 1 THEN 380
340   PRINT: PRINT "YOU HAVE";A2;"OBSTACLES TO OVERCOME."
350   PRINT "IF YOU DON'T LOCATE THE TREASURE BEFORE THE LAST"
360   PRINT "OBSTACLE, YOU'LL HAVE TO START OVER."
380   PRINT
390   PRINT "OBSTACLE #";A3
```

To be Continued next Issue.....

01111000001111111001110101000001011101000101111111000001111111100001101

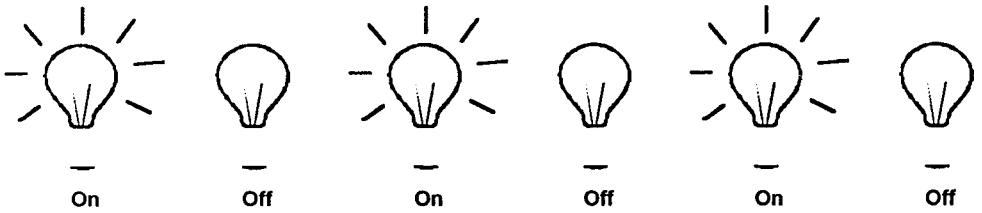
0110 # From BASIC to Machine Code 10010

10000000000111111111100101010101001111110000101111100001111000001111111

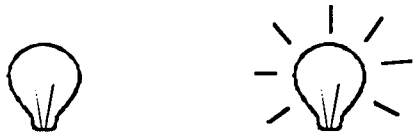
Commencing this issue, this regular section will introduce the average VZ Programmer to the world of Machine Code. My aim is to present a simple, step by step guide which takes the Programmer from BASIC to Machine Code (hence the title). If you find any of what I present this issue too difficult to understand, let me know so that I can fix things for the next issue.

What is Machine Code?

The VZ Computer is a complex piece of electronics. It is made up of many circuits containing micro-chips and other components (parts). These work by providing each component with electricity when needed. Think of each part as a light bulb, now as you know, by the flick of a switch you can switch a light bulb on and off. This is exactly what is going on inside your computer. If you flick a light switch on and off many times in a row you will end up with this:-



What you can see is "pulses" of electricity. When the bulb is on electricity flows and when the bulb is off, no or zero electricity flows. Therefore, we can now say that the number 0 represents no electricity and non-zero represents electricity flow. To keep things simple we say that the number 1 indicates electricity flow.



0

(No pulses of electricity)

1

(Pulses of electricity)

Getting back to our treasured computer, we can now imagine little "pulses" of electricity passing through different components. In the computer world these "pulses" are known by a different name (don't ask me why!) - BITS. These BITS can be either 0 or 1 (off or on). By setting these different BITS we can tell the VZ to perform a task (such as run a program). For example 101011100011001 would tell the VZ to do something (don't ask me what).
Now as you can see, a sequence of BITS means virtually nothing to us mere humans. So to help us understand things a **bit** better (excuse the pun) we have made up a couple of special words:-

A **NIBBLE** is a group of 4 BITS
A **BYTE** is a group of 8 BITS, as you can see half a byte is a nibble.

Inside the VZ, one BYTE can either be a command or a piece of information (such as the letter 'A'). A **Machine Code** Program consists of a sequence of BYTES. How? I hear you ask - I'll tell you that later.

Understanding BASIC

Before we get too carried away, we must first make sure we understand the BASIC's (I'm on a roll here). Now BASIC is itself a program which can be loaded from tape or diskette. On the VZ Computer, there is no need to do this as BASIC is contained in a special micro-chip and activates as soon as you switch the VZ on.
Just as an aside, our version of BASIC is almost identical to the Level II BASIC used on the once popular TRS80 and other computers around that time.

So, just what does happen when you type in a BASIC Program?

VZ BASIC can be thought of as a continuous loop containing an **INPUT** command. When the return key is entered the VZ checks whether the information typed in started with a number. If it doesn't start with a number then the information is compared against a list of valid commands. If it is in the list of commands then a Machine Code subroutine is executed and the command string discarded. If there is not a match then an error message is displayed on the screen.

Below are a couple of examples:-

CLS

If the input does start with a number then your input is assumed to be part of your program and is stored in memory.
Below is an explanation of two of the commands in BASIC:-

RUN

The VZ goes to where the BASIC program is being stored in memory. It then reads the program one line at a time. Because the VZ allows multiple commands on a single line, the VZ then breaks the line up into each command (eg. CLS:SOUND31,1). When it has extracted the command from the line it then goes to the machine code subroutine for that command and runs it. If all goes well with the BASIC program, the VZ will loop until it has finished the BASIC program.

(L)LIST

The VZ goes to the location in memory where the program starts. It then reads that location as a number, and prints the character with the ASCII code of number on the screen or printer. It then goes through every character on the rest of the input line and prints them. The VZ does this until the whole BASIC program has been printed on the screen or the printer.

eg. Load any BASIC Program and type in the following line (as it stands):-

```
FOR A=31465 TO 31944: PRINT PEEK(A);: NEXT
```

You should be able to recognise some of the characters as being part of the first few lines of the BASIC Program. We are listing the program without using the translator (also called an **Interpreter**).

What are the Advantages of Machine Code?

There are two main advantages, these are:-

1. The speed at which a machine code program runs.
   As mentioned before, when you run a BASIC Program, it is translated line by line into a series of calls to Machine Code subroutines. The time taken to do the translation slows the program down whilst it is running. If speed is not essential, then it is easier to use a BASIC program to perform a task. However, if you want to write a fast action game with graphics etc., then you will require some knowledge of Machine Code.

2. VZ BASIC is a restricted language, it only has 65 reserved words. (there are 65 Machine code subroutines for each reserved word). In Machine Code, although there is a restriction on the number of instructions available, each instruction is very powerful.

Next Issue.....

In the Next issue, I will briefly introduce you to the different numbering systems and some more specific details about the innards of the VZ.

Submissions

I will accept submissions in virtually ANY format, either on Disk or Tape. Failing that I will also accept a printed copy of short items which MUST be clear. Please remember that in this case I will have to re-enter your article/program.
I also have a TRS80 and have access to an IBM. So basically if you send it I can use it!!

Software Project - A New Operating System (NOS V1.0)

As mentioned last issue, for the more Advanced Machine Code Programmers I will be doing a series of articles allowing us to enhance the VZ so that it will provide features available today on the newer PC's. So here we go.....

What are the aims of this Project?

The main aim of this project is to allow easier access to any peripheral devices attached to the VZ by any piece of software already written for the VZ with little or NO Modification. This statement already reeks of impossibility, but we will try our best - remember the best challenges are those which are said to be impossible.

The main assumptions before you commence this project are:-

1) You have **reasonable** knowledge of Assembly Language Programming.
2) You own a VZ with a 16K RAM Extension.
3) You have time to do the work required.

My Vision

I see in the near future the following VZ Setup:-

A VZ Computer with up to 4 megabytes of memory (available to all programs - even BASIC Programs), Multiple Disk Drives attached (including a 720K 3.5 Inch Drive), Enhanced Keyboard, optional Ram Disk Software, a Real Time Clock keeping track of the Date and Time allowing Alarm Features etc., a Mouse or other pointing device, an 80 Column screen for better viewing of the screen data, and to round off the set-up a Sound Blaster Card allowing you to record and play back sounds from various input devices (TV, Video, Radio, stereo etc.).

As you can see we are aiming high.

The BASICs

All components in **NOS** are designed to be replaceable, so that you may configure your system in any way you would like. The system has been designed to emulate the DOS System of the popular IBM Personal Computers. That is, it will provide similar functionality but will **NOT** be 100% compatible. **NOS** will consist of nine basic components:-

B:	**BOOT.EXE**	This is the **NOS** Initialisation program.
B:	**MEMORY**	This is the Memory Manager and will look after all memory allocation regardless of its location.
B:	**PROCMAN**	This is the Process Manager and looks after the loading/executing of all Programs.
B:	**SCRNMAN**	This is the Screen Memory Manager and looks after the different Graphics Modes etc.
B:	**DISKMAN**	This is the Disk Manager and shields all applications from the native Disk Drive Commands.
B:	**PRINTMAN**	This is the Printer Manager and looks after the translation of control codes and provides a Printer Spooler.
T:	**CONFIG**	This file contains the Peripheral Device Driver installation/configuration information.
B:	**COMMAND**	This is the Command Processor and contains the basic reserved words available at the command line.
T:	**AUTOEXEC**	This file contains any other Programs the User may wish to load when starting **NOS**.

Notes/Assumptions:

1) The files listed above **must** all live on the same diskette - which will from this point forth be referred to as the **NOS Boot Disk.**
2) The **NOS Boot Disk** is a standard VZ Diskette and as such can only be read from a VZ Disk Drive. This means that we will have two limitations: 80K Maximum Boot Disk size & also we cannot boot from a different type of Disk Drive yet. These are not a major hurdle and will be resolved in the future.

The Initialisation Process (BOOT.EXE)

As mentioned above, the entry point into **NOS** will be via **BOOT.EXE** which, as with standard VZ disk software, will be started with the usual BRUN"BOOT.EXE". Once loaded, the Boot program will perform some minor tasks such as setting up internal pointers and reserving permanent memory areas for itself. It will then commence to load the other programs from the Boot disk in the order listed above. The loading order is essential as some of the processes rely upon the previous components being there to operate correctly.
Below is the pseudo code for the Boot Process. We will create the true executable version later.

```
BeginProgram 'Boot'
Origin: 40960                                          ; ie. A000H

        Disable Interrupts                             ; Prevent the System from having problems
                                                       ; whilst initialisation occurs.

        Reserve 25 Bytes for the System Information Frame (SIF)   ; See Below for Details
        at locations 31465 - 31489

        Initialise SIF:                                ; Values are for NOS 1.0 ONLY.

                Store NOS Version Description Pointer  ; = 31490
                Store High Order NOS Version           ; = 1
                Store Low Order NOS Version            ; = 0
                Initialise the System Flags            ; = 0
                Initialise the Max Process ID          ; = 255
                Set the Base System Stack Pointer      ; = 31597 + 40 (Default Stack Size)
                Set the Top System Stack Pointer       ; = 31597
                Set the Current System Stack Pointer   ; = Base System Stack Pointer.
                Store IM1 Table Pointer                ; = 31507
                Reset IM1 Function Count               ; = 0
                Store the Critical Error Handler Address. ; = 31636

        ; All following addresses are known only by the Boot Program only.
        ; Any Process wishing to use the following information should go via the System Information Frame so that they will remain
        ; compatible with future versions of NOS.

        Store Version Description String at location 31490 (16 Bytes)   ; "NOS Version 1.0"

        Initialise Interrupt Device Table at location 31507             ; 31507 - 96 = 0
        (max 30 Functions - 3 Bytes each = 90 Bytes)

        Reserve 40 Bytes for the NOS System Stack at location 31597

        Install the Critical Error Handler at location 31636   ; See Below for Details.

        Install the Interrupt Processing Subroutine for IM1    ; See Below for Details
        executed via an RST38H instruction.

        ; Load the remaining NOS System Files.
        EXECUTE [MEMORY]                               ; Install the Memory Manager
        EXECUTE [PROCMAN]                              ; Install the Process Manager
        EXECUTE [SCRNMAN]                              ; Install the Screen Manager
        EXECUTE [DISKMAN]                              ; Install the Disk Manager
        EXECUTE [PRINTMAN]                             ; Install the Printer Manager

        Call SYS Interpreter [CONFIG]                  ; Read the CONFIG file line by line and perform
                                                       ; the specified commands

        EXECUTE [COMMAND]                              ; Install the Command Line Processor

        Call Batch Interpreter [AUTOEXEC]              ; Read the AUTOEXEC file line by line and perform
                                                       ; the specified commands

        Enable Interrupts

EndProgram 'Boot'                                      ; Return control to the User (NOS Version 1.0 Installed)
```

What is the System Information Frame (SIF)?

This is an area of memory which contains all the important variables needed by the System to handle the multi-processing and the communication between processes and **NOS**. The definition of **SIF** is still not yet complete and as such the 25 bytes has some reserved space to allow us to add information as we go. As **NOS** expands so to will **SIF**. At this stage, once added to **SIF**, entries will never be removed. This way backwards compatibility will be retained so that software written for **NOS** V1.0 will still run under a future version (3.0 say?) of **NOS**. Following is the current definition of **SIF**:-

SIF: Location 31465 decimal, **Size** 25 Bytes

Offset	# of Bytes	Description
0	2	NOS Version Description Pointer.
2	1	High Order NOS Version.
3	1	Low Order NOS Version.
4	2	Base System Stack Pointer.
6	2	Top of System Stack Pointer.
8	2	Current System Stack Pointer.
10	2	Reserved.
12	1	Reserved.
13	2	IM1 Interrupt Device Table.
15	1	No. of entries in IM1 Interrupt Device Table.
16	2	Reserved.
18	1	Reserved.
19	1	System Flags: Bit 0 = Stack In Use Indicator 00 - SP is Current Process Stack Pointer. 01 - SP is System Stack Pointer. 10 - Reserved. 11 - Reserved. Bits 1 & 2 = Interrupt Processing is currently being done. 00 - No Interrupt is being Processed. 01 - Reserved. 10 - IM1 Interrupt is being Processed. 11 - Reserved. Bits 3 & 4 = Request System/Process Stack Pointer after Interrupt. 00 - No Request. 01 - Request System Stack . 10 - Request Current Process Stack. 11 - Reserved. Bits 5 - 8 = Reserved.
20	1	Max Process ID. (Total No of Processes currently loaded).
21	2	Address of the Critical Error Handler.
23	2	Reserved.

Note: Remember that on Z80 machines the Stack grows down.

What is the Critical Error Handler?

The Critical Error Handler is just that, a centralised Error function which given an error code produces an Error Message and then either returns control to the Originating Process or terminates it if a major Error has occurred. The Error Messages defined so far are:-

Errorcode	Reason for Error.	Error Message
0**	Invalid Process ID	A Requested Process is not Available.
1**	Invalid Function Call	A Required Function is not Available.
2*	Stack Underflow Error	A Stack Error has occurred. Application is Terminating.
3*	Stack Overflow Error	A Stack Error has occurred. Application is Terminating.

* Indicates Current Process will Terminate.
** Indicates that the Error Message can be suppressed by the Originating Process.

What is the IM1 Interrupt Processing Subroutine?

Firstly, you should note that, by default, the VZ is setup to allow IM1 type interrupts. This is the most simple of the three types of interrupts permitted on a Z80 machine. What happens is that when an interrupt occurs an **RST38H** instruction is executed. Currently on the VZ, the only interrupting device is the Graphics Controller Chip which interrupts 50 times per second. Every interrupt the VZ looks after the flashing of the cursor and message updates onto the screen. Over the years, people have connected a variety of different peripherals. This has in many cases meant replacing the standard Interrupt Processor with one specific for the device being added. This means that two devices could not be used at the same time. The Interrupt Processor installed by **NOS** will attempt to overcome this problem.

For the initial version of **NOS** we will only be considering IM1 type of interrupts. It is highly probable that this will change in a future version of **NOS** but this will require hardware enhancements to the VZ.

In **NOS** there is a Table which contains a list of functions (in priority order) which are to be called when an interrupt occurs. This table is known as the **IM1 Interrupt Device Table** and its address is stored at location 13 of **SIF**. When an interrupt occurs, the Interrupt Processor traverses the list executing each function in the list depending upon its priority. For this version of **NOS** there is a limit of 30 functions which can be called when an interrupt occurs. Each function in the list looks after a specific device and whenever called it polls the device to see if any events occurred since the last interrupt. There is one major drawback to this method - that is if we don't poll a device often enough we will

loose information sent to us by that device. The only way to resolve this is via some form of hardware enhancement which is beyond the scope of the initial version - we will have to make do!

So we are now left with the question how do we handle interrupts of devices which require different polling priorities? Also, there is a limit in the amount of time we can spend in the Interrupt Processing Subroutine before the system slows down. I will leave you with these thoughts until next issue...

Other VZ Publications/User Groups

Hunter Valley Journal

Editor: Joe Leon
Address: 35 Tighes Terrace,
Tighes Hill
NSW 2297

This is an excellent publication containing a wealth of information about the VZ. It contains many Hardware and Software projects for the beginner and expert alike.
I recommend that you subscribe and support this journal. Subscription rates are approximately $18-00 for six issues. I would also advise that you order all Back Issues for inclusion in your personal VZ library.

Other VZ Groups

It appears that there are no other VZ Groups operating in Australia or New Zealand. However, I have been in contact with a User Group in the UK which does cater for the LASER Computers (this is the name of the VZ in some other countries). I am trying to find out more information.

Groups of Interest

Late last year I was contacted by a User Group which I believe no longer existed - SYDTRUG.
This User Group was formed back in 1979 and catered for the TRS80 Computer. They have changed focus slightly through the years and now support MSDOS and well as CP/M, DOSPLUS, LDOS LS-DOS, MULTIDOS, NEWDOS and TRSDOS. Their main aim now is to enhance knowledge of Hardware and Software and generally promote further debelopment in Computing. You should also note that the have members all around the world and have an extensive Software and Reference Library.

The full Address is:-

SYDTRUG Inc.
The Sydney TRS-80/MS-DOS Users' Group
P.O. Box 75
PANANIA NSW 213

They also run a Bulletin Board 'TRUG-86': The Phone Number is **(02) 790-5681** and will provide limited access to Visitors. The information I was sent indicated that this Bulletin Board is new and so do not give up if you experience some teething troubles.

End Notes

Please Note: I Have Moved

If you wish to get in touch with me, my NEW address is

138 Kingswood Road, Engadine NSW 2233 Australia.

I can be contacted during business hours on (02) 257 8059. After hours you can just leave a message. I will return your call as soon as possible.