C O N T E N T S

DICK SMITH
VZ-200
EDITOR ASSEMBLER

INTRODUCTION

This Instruction Manual is not meant to be a training course in ASSEMBLER LANGUAGE programming, it is meant only to explain the commands invovled with this EDITOR ASSEMBLER, although the APPENDICES do provide some important information for writing MACHINE LANGUAGE programs on the VZ-200 computer.

THE TEXT EDITOR

There are 5 basic commands used for text entry and display, they are INSERT, DELETE, EDIT, LIST and FIND. These are invoked by using the first letter of each command eg. I, D, E, L and F.

There are 5 tape commands, TAPE SAVE, TAPE LOAD, TAPE VERIFY, TAPE MERGE and TAPE OBJECT. These are invoked by using the first letter of each word eg. TS, TL, TV, TM and TO.

As well as the above 10 commands there are 4 special purpose commands, ASSEMBLE, RUN PROGRAM, ORIGIN and SET PARAMETERS. These are invoked by using just their first letter. eg A, R, O and S.

INSERT COMMAND

This command is used to input source lines. It has two formats :-

I<RETURN> - this will start to insert lines directly after the line pointed to by the CURRENT LINE POINTER, this will be the line last listed, normally just above the COMMAND LINE.

Innn<RETURN> - this will start to insert lines directly after line nnn.

After entering your line press <RETURN> to store your line in the SOURCE BUFFER. A new line number will be printed and you may type in a new line. Pressing <CTRL> + <BREAK> will exit the INSERT MODE and return

you to the COMMAND MODE.

LINE FORMAT

Each line, with only 3 exceptions, must conform to the following format. Each line may contain a LABEL, must contain an OPCODE and depending on the OPCODE an OPERAND. No COMMENT is allowed. The 3 exceptions are COMMENT LINES, COMMAND LINES and MESSAGE LINES.

If a you wish to place a LABEL on a line then it must occupy the first 1 to 4 places of the line, no preceeding spaces are allowed. The first character must be alphabetical while the last 3 may be alphanumeric, no special characters are allowed (eg. !"#$%&).

At least one space must separate the LABEL and the OPCODE (if no LABEL was used the first character of the line must be a space). The OPCODE must follow the same rules as the LABEL as far as number and type of characters. Of course the OPCODE must be one of the 71 recognised OPCODES by this ASSEMBLER if it is to be accepted at assembly time. A list of these 71 opcodes is given in APPENDIX A.

If an OPERAND is required then at least one space must be between it and the OPCODE. No check is done on the OPERAND but the first space found after the commencement of the OPERAND will be taken as the end of the line.

If the LABEL or the OPCODE do not conform to the 1 to 4, alphabetical / alphanumerical format then a LINE FORMAT ERROR will be given and you will be asked to re-enter the line.

COMMENT LINE

Comments must be on a line by themselves and the line must start with a semi-colon (;).

```
eg 001 ;GREAT GAME
   002 ;WRITTEN BY G.BLOGGS
   003 ;LAST UPDATE 12/12/84
```


COMMAND LINES

Command lines are used to send special commands to the assembler during assembly. These lines must be on a line by themselves and start with a colon (:). The commands they set are described under ASSEMBLY OPTIONS.

MESSAGE LINES

The DEFM opcode is not allowed. If this opcode was used for defining message they would be limited to 18 characters - the max. length an operand can be (due to the single line entry format of the editor). While this is long enougth for any operand it does limit messages. Of course messages longer then 18 characters could be defined with 2 DEFM opcodes and it would be the same thing as far as the finished assembly would be concerned. But to make message handling a little easier a special line indicator is used for messages, it is the asterick (*). This gives a max. message length of 27 characters. If more is required then unfortunately 2 lines will be needed. The end delimiter for a message is either an asterick or the first non-space characters working back from the end of the line.

```
eg 001*THIS IS A MESSAGE
   002*  THIS IS A MESSAGE  *
   003** THIS IS A MESSAGE **
```

Line 3 will be assembled * THIS IS A MESSAGE *, only the first and last * are message delimiters, the rest are part of the message.

If a label is required for a message then you must use the EQU $ (OPCODE OPERAND)

```
eg 001 MES1 EQU  $
   002 *THIS IS A MESSAGE
   003      LD   HL,MES1
```

In the above example the register set HL will be loaded with the address of the memory location that contains the first letter of the message.

DELETE COMMAND

There are 3 forms of the DELETE command. They are :-

D<RETURN> - Delete the line pointed to by the CURRENT LINE POINTER. The CURRENT LINE POINTER will now point to the line below the one just deleted except in the case of the line deleted being the last line in the source buffer, in which case it will point to the one above.

Dnnn<RETURN> - Delete line nnn

Dnnn:mmm<RETURN> - Delete lines nnn and mmm and all lines inbetween.

*** WARNING ***

Whenever a single line is deleted an automatic line renumber is done, so all the lines below the line you deleted will now be one number less, so if you wish to delete say lines 15, 24, 38 and 67 then start with the highest number and work down eg. delete 67 first and 15 last. If 15 is deleted first then line 24 will now be line 23 and a D24<RETURN> will delete the wrong line.

EDIT COMMAND

There are 3 forms of the EDIT MODE. They are :-

E<RETURN> - edit line currently pointed to by the CURRENT LINE POINTER.

Ennn<RETURN> - edit line nnn.

Ennn:mmm<RETURN> - edit line nnn and mmm and all lines in between. In this mode, after you have edited a line press <RETURN> and the next line to be edited will be displayed.

In all the EDIT MODES pressing <CTRL> + <BREAK> will end the EDIT MODE for that line and cancel any changes that may have been made.

LIST COMMAND

There are 5 types of LIST COMMANDS. They are :-

L<RETURN> - list all lines

Lnnn<RETURN> - list line nnn.

Lnnn:-<RETURN> - list from line nnn to the end of the source.

L-:nnn<RETURN> - list all line from the beginning to and including line nnn

Lnnn:mmm<RETURN> - list lines nnn and mmm and all lines in between.

In all the multiple line listings the following keys will have the following functions :-

<S> - slow listing down

<F> - speed listing up

<H> - hold listing

<C> - continue listing after a hold <H>.

<CTRL> + <BREAK> - exit list mode.

FIND COMMAND

Fstring<RETURN> - will search through the source file and find the first occurence of the string. It will list the line and then ask NEXT (Y=N). If you wish to find the next string press Y if not press N and you will be returned to the command mode. Using F<RETURN> with no string will find the first occurance of the string last asked for starting the search from the line pointed to by the CURRENT LINE POINTER. The string can be a max. of 8 characters.

TAPE SAVE

TS:name<RETURM> - this command is used to save your source file to tape for later retrieval. The name can be a max. of 8 characters long and must start with an alphabetical character.

TAPE LOAD

TL<RETURN> - this will load a source file from tape into memory. No name is allowed as the program will not search for a file but will load the first file it comes to. The name of the file being loaded will be displayed on the bottom line.

TAPE VERIFY

TV<RETURN> - this command is used to verify that a source file has been saved correctly.

As in the TL command no name is allowed or searched for. It is used only to verify tapes made with the TS command, not the TO command.

TAPE MERGE

TM<RETURN> - this command will merge a source file on tape with the one that is currently in memory. As in TL and TV no name is allowed or searched for. This command allows a library of useful routines to be built up and then merged into a source file as required.

TAPE OBJECT

TO:name<RETURN> - This command will write an object tape (the actual machine language program). The program will auto execute when loaded back into the VZ-200 either with the CRUN or CLOAD command. This command can be used only after the source program has been assembled and provided that no source lines have been edited, deleted or inserted. If the source has been modified in any way a REASSEMBLY REQUIRED error will be given.

ASSEMBLE

A<RETURN> - this command will assemble the program. The program will be assembled starting from the first free byte after the source file but it will be assembled in referance to the value set by the ORGIN COMMAND, this is the address that the program will load at and excute at when it is loaded back into the VZ-200 as an object tape.

ORIGIN COMMAND

Onnnn<RETURN> - set the origin for the assembled program to nnnn.
O<RETURN> - this will set the origin to the first free byte passed the source file. If a program is assembled here then it can be run with the EDITOR ASSEMBLER still in memory.

Provided you do not touch any memory below that set by the ORIGIN COMMAND then you may jump back to the EDITOR ASSEMBLER at 31488 (7B00H) and your source file will still be intact.

RUN COMMAND

R<RETURN> - This command will run the assembled program provided 2 requirements are met. If any lines have been edit, deleted or inserted since assembly you will recieve a REASSEMBLY REQUIRED error. If the ORIGIN is not set to the first free byte after the source code using the O<RETURN> command then you will get a WRONG ORIGIN FOR RUN error. After running your program you may re-enter the EDITOR ASSEMBLER by doing a jump to 31488 (7B00H). Remember the first free byte after the source code is constantly changing as you enter source lines so if you wish to test run your programs using the R command use the O command before every assembly. You will be prompted SURE (Y-N) if everything is correct to run the program.

** W A R N I N G **

Always save your source file before test running a program in memory as often the slightest mistake will cause a program crash which will result in one of two things, one the computer will become locked up, and you will have to turn it off and on again to gain control or the computer will reset itself. In either case you will have to load the EDITOR ASSEMBLER again along with your saved source file.

SET PARAMETERS COMMAND

S<RETURN> - set parameters to default value
SNABC<RETURN> - Set parameter N to the number given (1-3) and turn A, B, and C on. Not all four parameters need be given when this command is used but if they are not given then they will be reset to there default value

The default value for the parameters are
N = 1
A = OFF
B = OFF
C = OFF

The parameters above are used to set certain assembly and listing options.

ASSEMBLY AND LISTING OPTIONS

Four options can be set for assembly and listing with the SET command from the editor or the colon (command line) during the assembly of a program. The parameters and their meanings are

PARAMETER A. If this is set ON then the assembler will list all lines and their corresponding assembled code. If this is set OFF then only lines with errors will be listed.

PARAMETER B. If this is set ON then the assembler will halt on an error and wait for you to press the <C> key before continuing.

PARAMETER C. If this is set ON then any lines listed during a LIST COMMAND or an ASSEMBLE COMMAND will also be sent to the printer.

PARAMETER N. N can be either 1, 2 or 3 and will decide how the assembler handles the assembly of messages.

N=1 - all messages will be assembled in ASCII format eg 0=48, A=65

N=2 - all messages will be assembled with bit 6 set off. eg 0=48, A=1. This will give you white characters on a black background if they are written directly to the screen.

N=3 - all messages will be assembled with but 6 set on. eg. 0=112, A=65. This will give you black characters on a white background if they are written directly to the screen.

NOTE :- If you are going to use the VZ-200 rom calls listed in APPENDIX B to send characters to the screen then assemble characters in ASCII format (N=1).

A fifth option can be used with the colon (command line) during an assembly. This is the H option which will simple cause a forced halt in assembly until you press the <C> key. A command line may be used to set parameters or force a halt but it may not do both.

EXAMPLE USE OF COMMAND LINES

```
001 TEST LD   HL,7000H
002      LD   DE,7001H
003:AC
004      LD   BC,1FFH
005      LD   (HL),32
006:A
007      LDIR
008      XOR  A
009:H
010      LD   (26624),A
011      JP   7B00H
```

Line 3 will turn parameters A and C on. This means that lines from now on will be listed to the screen (A) and also sent to the printer (C). Line 6 will turn option C off so lines will still be sent to the screen but not the printer. Line 9 causes a halt in assembly until the <C> key is pressed.

STATUS LINE (TOP LINE)

The top line is the status line is shown in inverse to the rest of the screen. It shows free memory (FM), the currently set origin (ORG) and the state of the parameters (SPAR). The parameter N will be shown (a number from 1 to 3) and if A, B or C is on then their letter will be shown.

For example, if free memory equals 10500, the currently set orgin is 32000, N=2, A=on, B=off and C=on then the top line will be :-
FM:10500 ORG:32000 SPAR:2A C

Whenever there is a hold on a listing (H), a forced halt in an assembly (:H) or a hold on error status situation then the top line will flash to show you that the program is waiting for you to press the <C> key.

INTERPRETATION OF NUMBERS

The ASSEMBLER recognises both decicmal

and hexadecimal numbers. All numbers must start with a digit else it will be interpreted as a label. All hex. numbers must end with the prefix H. EXAMPLES :-
100 = 100 in decimal
1000H = 1000 in Hex. (equal to 4096 in decimal)
65535 = 65535 in decimal
0FFFFH = FFFF in Hex. (equal to 65535 in decimal)

ARITHMETIC OPERATIONS

The ASSEMBLER is capable of only performing addition and subtraction and then only once in any equation. EXAMPLES :-
100+10
1000H+0AF0H
TES1+1245
234-12
10-2300
1089H-TES1
TES1+TES2

All the above are legal with TES1 and TES2 being labels. The following 2 examples are not legal because there is more then one operation.
120+2300-12
TEST+123+12

A P P E N D I X A
OPCODES RECOGNISED BY THIS ASSEMBLER

ADC	DAA	IM	LDI	RES	RRCA
ADD	DEC	IN	LDIR	RET	RRD
AND	DEFB	INC	NEG	RETI	RST
BIT	DEFS	IND	NOP	RETN	SBC
CALL	DEFW	INDR	OR	RL	SCF
CCF	DI	INI	OTDR	RLA	SET
CP	DJNZ	INIR	OUIR	RLC	SLA
CPD	EI	JP	OUT	RLCA	SRA
CPDR	EQU	JR	OUTD	RLD	SRL
CPI	EX	LD	OUTI	RR	SUB
CPIR	EXX	LDD	POP	RRA	XOR
CPL	HALT	LDDR	PUSH	RRC	

A P P E N D I X B
USEFUL ROM CALLS

SCAN KEYBOARD
CALL 2EF4H

All registers are used and on return A reg. will hold the ASCII equivalent to the key pressed, 0 if no key pressed. See example program one.

OUTPUT CHARACTER TO VIDEO
CALL 33AH

On entry A reg. holds ASCII equivalent of character to display. All registers are preserved.

OUTPUT MESSAGE TO VIDEO
CALL 2B75H

On entry HL registers must point to the start of the message and the message must be terminated by a zero byte. All registers are used.

CLEAR SCREEN
CALL 1C9H

All registers are used.

GENERATE SOUND
CALL 345CH

On entry HL registers contain the pitch and BC registers contain the duration. All registers are used.

GENERATE BEEP
CALL 3450H

This routine generates the beep heard on pressing a key in normal VZ-200 BASIC. All registers except HL are used.

SEND CHARACTER TO PRINTER
CALL 58DH

On entry C register contains the ASCII equivalent of the character to be printed. All registers are used.

CHECK PRINTER STATUS
CALL 5C4H

Only the A register is used and on return if A = 1 then the printer is busy or if A = 0 then the printer is not busy.

TAPE ROUTINES

TAPE FORMAT.

LEADER - this is a block of 255 80H bytes followed by 5 0FEH bytes.
COMMAND BYTE - this byte tells the VZ-200 what to do after loading the program. All basic programs are saved with a 0F0H byte. A 0F1H byte will make the program auto execute starting at the first address of the loaded block. This is the command byte that the EDITOR ASSEMBLER places on its OBJECT tapes.
NAME OF PROGRAM - This will be up 16 characters long and terminated with a zero byte.
DELAY - After the name is placed on the tape there is a short delay. This is placed here to allow the computer to check the name on the tape against the name of the program you wish to load. The delay is obtained with the following short program.

```
001       LD   BC,019AH
002 RPT   DEC  BC
003       LD   A,B
004       OR   C
005       JP   NZ,RPT
```

START ADDRESS - This is the beginning address of the block to load.
END ADDRESS - This is the end address of the block to load.
DATA - This is the block of data to load.
CHECKSUM BYTES - This is the next 2 bytes after the data and must be the same as the checksum of the data just loaded. The checksum starts with the first byte of the START ADDRESS and finishes with the last byte of the DATA. As the tape is being loaded it is stored in a 2 byte buffer pointed to by the IX regs. The buffer used

by BASIC is at 7823/4H.
TRAILER - this is 14 zero bytes which follow the checksum bytes.

SAVE PROGRAM
CALL 34A9H

On entry the start of the block of memory to save must be loaded into 78A4/5H (start of basic pointer) and the end of block to be saved must be placed in 78F9/AH (end of basic pointer). The HL regs. must point to the name to be given to the saved program. The name must start and finish with a quote ("). Only the first 16 characters will be used. The program will be saved with the command byte of 0F0H. If you want this to be differant than load C register with your new command byte and do a DI instruction and then CALL 34ACH

LOAD PROGRAM
CALL 365FH

Before calling the HL regs. must point to the name of the program to load with the name starting with a quote (") and ending with a quote ("). Only the first 16 characters will be used and they will bs transferred into a buffer starting at 7A9DH with the length being stored in location 7AD6H. If you wish the first program to be loaded regardless of name then the HL regs. must point to either a 3AH or 0 byte.

LOAD NAME
CALL 358CH

This routine will load a name pointed to by the HL regsinto a 16 byte buffer at loaction 7A9DH and store the length of the name at location 7AD2H. The name must start and finish with a quote ("). Only the first 16 characters will be used.

WRITE LEADER
CALL 3558H

This routine will write a leader, a command byte and the name to tape. The name

must be loaded as in the LOAD NAME ROUTINE. C reg must contain the command byte.

FIND LEADER
CALL 35E1H

This routine will search for a leader, and specific name. The name to find must be loaded as in the LOAD NAME ROUTINE. The command byte found after the leader will be stored in location 7AD2H. WARNING - it will not return till the name is found.

FIND START/END
CALL 3868H

This routine is called after finding the leader and name and will return with the start address of the block to load in the DE regs and the end address of the block to load in the HL regs. Before calling the IX regs must be loaded with address of a 2 byte buffer which will be used for the checksum test at the end of the program load. Basic uses locations 7823/4H ie load IX with 7823H.

READ BYTE
CALL 3775H

This routine will return with the next byte found on a tape in the A reg. If no byte is found within a specific time period then the routine will return with the Carry Flag set. All registers are preserved.

WRITE BYTE
CALL 3511H

This routine will write the byte in the A reg to tape. All other registers are preserved.

CHECKSUM ADD
CALL 388EH

This routine adds the A reg to the 16 bit value pointed to by the IX regs.

JUMP TO BASIC
JP 1A19H

The OBJECT TAPES written by this EDITOR ASSEMBLER will auto execute when loaded back into the computer. This is fine for programs like this EDITOR ASSEMBLER but there may be times when you wish to load sub-routines into high memory which will then be called from a BASIC program. To get your object tape to return to basic after it has loaded just make the first instruction a JP 1A19H.

EXTRA INFORMATION

The VZ-200 TECHNICAL REFERENCE MANUAL is another essential tool for the budding assembly language programmer. It contains information on the memory map, interfacing with the keyboard and video display, as well as other useful ROM routines.

Another good book for general Z-80 programming is HOW TO PROGRAM THE Z-80 by RODNAY ZAKS. This book describes all the Z-80 opcodes with a good description of what they do and how they effect the FLAG register. This book, or one similar, is another essential requisition.