ATOMIC FUSION

Joining the ATOM and DISATOM

A MANUAL FOR THE DISATOM SUPER ROM

Copyright (c) - J.R. Stevenson and J.C. Rockett - 1982

INTRODUCTION AND CONTENTS

The DISATOM SUPER ROM is contained in a 4K ROM that is fitted in the utility socket (address A000).The Floating Point Chip must also be resident. It contains two major areas: Machine Level with Memory Handling, and Additons to BASIC. It is permanently resident, does not require a LINK command, and does not use any addresses (such as zero page) you are likely to use. Most words may be abbreviated, and used in BASIC programs. It is undoubtedly the most advanced chip of its kind on the market.

ADDITIONS TO BASIC:

MACHINE LEVEL FUNCTIONS:

These m/c commands make the machine absolutely TRANSPARENT, and it is worth the time of even a novice to learn how to handle them.

PART I
ADDITIONS TO BASIC

AULD

AULD

The function of this command is complementary to PAGE in that it moves the text-space pointer. However, after the pointer (at #12) is moved the command OLD is executed. AULD therefore behaves as a targetable OLD, in that you can recall programs from any area of memory where they may reside. The syntax for AULD is the same as for PAGE and one command is used in conjunction with the other. AULD XX , where XX is two hex digits,refers to the memory page number where you wish to revover text. Thus AULD 82 recovers text at#8200, AULD 28 recovers any at#2800, and so on. Be sure there is BASIC text at that location!

AUTO

AUTO

This command saves the program writer from having to type out a new line number each time while writing a program. After executing the command the computer offers a line number after which textual material may be typed in. On hitting the <CR> key the line just typed is entered into the program, and the next line number is offered.

The AUTO mode is cancelled by either pressing the <ESC> key or by hitting <CR> twice. AUTO can be followed by one or two numbers separated by a comma. The first number indicates the line number at which you wish to begin, and the second is the line number distance between each line. If AUTO is not followed by any numbers, than the first line number offered is 100, and the step size is 10. This is acceptable, since ATOM always uses two bytes to store a line number regardless of what size it is.

Some examples are:

AUTO	START AT 100, STEP SIZE 10
AUTO 200	START AT 200, STEP SIZE 10
AUTO 150,20	START AT 150, STEP SIZE 20
AUTO,5	START AT 100, STEP SIZE 5

The numbers which follow the AUTO command can be of any value in the range 32767 to 0. However, if the command causes a line number tc be generated outside the allowable range then an error will occur and the command cancelled.

COPY

COPY

The COPY command , which can be used in direct mode or in a program, is provided to rapidly copy sections of memory into new locations.The command must be followed by three parameters: COPY X,Y,Z . The first two, X and Y, are the start and end addresses of the data to be moved, and the third address ,Z, is the start address where the data is to be relocated. They are separated by commas or spaces.

-continued overleaf-

The parameters X,Y,Z may be numbers, variables, or expressions, but if they are expressions they should be enclosed in brackets. In general the format of this command is the same as that for PLOT. Remember that in moving memory you are normally dealing in HEX numbers, and so the hex sign should be placed in front of the parameter.

The function takes care of the direction in which the data is copied. Thus whole areas of memory may be moved to overlapping areas either up or down, without corruption of the data. As an example, the following program spirals the whole screen area continuously one character to the right and down:

```
100 S=#8000;E=#81FF
110 DO
120 COPY S,E,(S+1)
130 UNTIL 0
```

CURSOR

CURSOR(

This function allows the cursor to be moved rapidly to simplify and speed up printing.

The syntax of the command is CURSOR X,Y where X and Y are the column and row numbers of the new cursor position. Positio 0,0 is the top left of the screen. X and Y may be numbers,integer variables, or an expression enclosed in brackets which will yield a number. Either X or Y must be specified. Examples are:

```
CURSOR 20,10          MOVE TO COL 20 ROW 10
CURSOR ,5             STAY IN SAME COL, BUT ROW 5
CURSOR 8,.            SAME ROW, BUT MOVE TO COL 8
CURSOR RND,RND        MOVE TO A RANDOM POSITION
```

DATA

DATA

DATA is part of the DATA-READ-RESTORE function.
The READ statement is used to assign values to variables, and the values themselves are stored in a DATA statement. This means that you need not use INPUT to assign values. The computer considers that all DATA lines are in fact one giant single one, so when it reaches the end of the first DATA line, it will automaticaly carry on to the beginning of the next. The word RESTORE moves the data pointer to the beginning of the first line of DATA, but in the case of the DISATOM it is targetable, so you can RESTORE to any line you wish.

The data in the DATA statement can be: numeric,string enclosed in quotes,or an expression enclosed in brackets which will yield a value. The pieces of data must be separated by commas.A comma may be used in a string, since the string data is enclosed in quotes.

-continued overleaf-

The DATA lines may be placed anywhere in the program, treated by the interpreter as REM statements. DATA must be the first command on the line, but may optionally be preceeded by a label. Spaces leading to the word DATA are ignored. ALWAYS remember to RESTORE before executing the first READ to set the data pointer. Examples:

```
100DATA "john","ARTHUR",A+B
110   DATA"Morecambe, Lancashire", #20
120 DATA 471,326,18,-177
```

A typical use might be to assign values to the varaibles in an array as follows:

```
10 DIM AA(5); RESTORE
20 FOR I=1 TO 5
30 READ AA(I)
40 NEXT I
50 DATA 54,67,555,-312,0
60F.I=1TO5;P.AAI';N.;E.                    (prints out array)
```

DELETE

DELETE

This function provides a facility to delete lines or whole sections of a BASIC program. The syntax is the same as for LIST, that is DELETE X,Y where X is the line where deletion will begin, and Y is the last line to be deleted. Either X or Y must be given or the function will not operate. Examples:

```
DELETE 200          REMOVE LINE 200 FROM THE PROGRAM
DELETE 100,200      REMOVE ALL 100 TO 200 INCLUSIVE
DELETE 100,         REMOVE FROM 100 TO END OF PROG
DELETE ,150         REMOVE ALL FROM START TO 150
```

DIR

DIR

Directory, lists all the reserved word in the DIASATOM. There are so many it is usually best to put ATOM into page mode first.

DUMP

DUMP

This command searches through the resident BASIC program for assigments to simple variables. If any have been used by the program, it prints out the variable and its value.

It can be a useful de-bugging tool as well as reminding the writer about which variables he has already used. DUMP always returns to direct mode, and so should be the last command if used in a program. Examples:

```
100 X=10;B=30;P=#20
110 DUMP
```

Executing RUN will display:

```
X=10
B=30
P=32
```

ERUN

ERUN

This command executes RUN in the normal way, except that in the event of an error occurring, the complete offending line is displayed with the cursor sitting on the character of the statement where the interpreter believed the error occurred. This command is most useful in picking out and correcting typographical errors that occured on program entry.

When there is a logic error (e.g. GOTO 700 , but there is no line 700) then the cursor will sit at the end of the offending line. It is normally unwise to mix ERUN and ON ERROR, but it is permissible.

EXEC$

EXEC$

Execute the BASIC statement in the specified string.

The parameter passed with the command specifies the string location. Its main use is to allow the user to enter complete equations, assignments, or commands while running a BASIC program.

```
10 DIM A(6),B(3),C(7);X=50
20 $A="GOTO90";$B="X=3"
30 INPUT"TYPE IN AN EQUATION"$C  (type in Y=3*X+1)
40 P.X';EXEC$B;P.X'
50 EXEC$C;P.$C,Y'
60 EXEC$A
70 END
90 P."JUMPED OVER END SO ERROR HERE"
```

This command can only be used in a BASIC program, but can be used in Direct Mode by pressing <ESC> afterward.

FIND

FIND

A) To find matching strings of characters in a BASIC program and list those lines containing them. In this mode the parameter after the command is enclosed in quotes.

B) to find matching strings of data or machine code anywhere in the memory map , and to print the hexadecimal address of the location where a match occurred. The parameter after the command may be machine code, assembly code enclosed in square brackets, or ASCII characters preceeded by a full stop, or a mix of these.

-continued overleaf-

EXAMPLES:

FIND "REM" - will list all lines in a BASIC program which contain REM statements.

FIND ":" - useful when writing assembler programs, as it provides a list of all labels used.

FIND "X=" -prints all lines where X was assigned a value.

FIND [JSR #FFF4] - will print out all locations where this call is made.

FIND [LDA@ #20;JSR #FFF4]- will print out all locations that contain the 5 bytes in this routine.

FIND 20 F4 FF - This example assumes that the data which follows is hexadecimal , and prints out all locations where this sequence occurs.

FIND .P .L .O .T - will print out the location where this sequence of ASCII characters is kept, such as the reserved word PLOT in the interpreter.

FIND .P .R .I .N .T - will give the hex location of any command PRINT.

FIND 41 .T .O .M - a mix of ASCII and hex, which will give the location where the interpreter keeps the word ATOM. All combinations of mode B may be mixed.

When using in mode B the spurious addresses #0100 and #0066 will sometimes be displayed. These addresses correspond to the input buffer which holds the string you have just typed in, and a scratch-pad area used by the assembler. Pressing the <ESC> key during a search cancels the search.

HEADER

HEADER

Sets the number of lines at the top of the screen which will not scroll. As many as 6 lines may be prevented from scrolling. You are permitted to write on these lines, but they will not be scrolled away when the rest of the display does. This is most useful when creating tables of results, figures, etc. , where the row containing the titles for each column can be stationary.

HEADER 0 cancels the command, as will the (unrelated) command LOW. HEADER may be used in program, or in direct mode.
EXAMPLES:

```
100 PRINT $12; HEADER 2
110 PRINT".......X.....X↑2.....X↑3"''
120 FOR X= 0 TO 50
130 Y=X*X ; Z=X*X*X
140 PRINT X,Y,Z'
150 NEXT X ; END
```

HELP

HELP

This is essentially an extra COS command, designed to help load programs from recordings of dubious quality. The word HELP is substituted for LOAD, and no * is used. If a SUM error occurs the message "REWIND TAPE A LITTLE" will appear. The volume and tone settings may then be adjusted, and the loading continued by hitting <CR>. In order to confirm that data is reaching the computer, the cursor will display each character that passes in. You may give up by pressing <ESC>, and HELP operates at either 300 or 1200 BAUD in the usual way. It is of no importance whether the program being HELPed is BASIC or m/c code.

```
HELP"TEST"
↑HELP"TANK GAME"
```

HIGH

HIGH

This command alters the COS to enable input or output to tape at 1200 BAUD. All COS commands are then carried out at this higher speed, and the cursor reflects any data either coming in or going out. Since the whistle between the blocks of a named file is still the same length, the command only doubles the time saved if named files are used, but gives a 4 times saving with un-named files, since they are not blocked.

At this speed it is important that both the recorder and the tape be of reasonable standard. You must have had trouble-free operation at 300 before you attempt 1200. The command may be used either in direct mode or in program, and is cancelled by hitting <BREAK> or executing the command LOW. The command . is a temporary version of this command, and will give high speed operation on this one event only, then revert back to 300 BAUD.

```
10 HIGH
20 *LOAD"TEST"8200
30 LOW

↑ *SAVE"INVADERS"2900 3C00 F141
```

↑

↑

This prefix can be used with any of the COS commands to induce temporary 1200 BAUD operation. As soon as the prompt sign > passes down the print channel, then operation returns to the normal 300 BAUD. The command can only be used in Direct Mode. To effect the same function in program see HIGH. Examples:

```
↑ LOAD"TEST"
↑ *CAT          (SAME AS ↑*.)
↑ HELP"DIFFICULT"
↑ *LOAD"BOMBERS"
```

INKEY

INKEY

Captures the ASCII value of the key pressed. The command waits for a specified period while sampling the keyboard to see if a key has been pressed. If a key is pressed before the time expires, then the ASCII value is returned, and can be modified by the <SHIFT> or <CTRL> keys if they are depressed. If no key is pressed within that time then a value of 255 is returned.

The first parameter after the command word is the integer variable to which the value of the keystroke will be passed, and the second parameter is the time limit. If no time limit is specified than a 0 is assumed and one keyboard scan executed. Time is in units of 1/20 th of a second, and may be any value in the range 0 to 32767. Examples:

```
100 INKEY A,20
100 T=200; INKEY X,T

10 INKEY K
20 DO; INKEY K
30 PRINT K, $K'
40 UNTIL K=13; REM 13 IS CARRIAGE RETURN
50 END
```

Key <A> on its own gives 65.
Pressed together <SHIFT> <A> gives 91.
Pressed together <CTRL> <A> gives 1.

LOW

LOW

This word can be used either in Direct mode or in program, and has the effect of resetting the COS to 300 BAUD operation. Although it is unrelated, it may also be used to cancel the HEADER and OUT commands. See HIGH for more details.

NUKE

NUKE

This command stores #FF into every RAM location up to #9FFF and then exits to the <BRK> routine. You can now be sure of the contents of all memory locations. This is useful when combined with the TAPE command, and in combination with the Hex Dump command H to examine how the operating system uses RAM. No program can withstand NUKE, so OLD wil not work. Be careful about this command if you have discs or other hardware in the NUKEd area. Also, the random number seed at #8 to C is now all FF .

ON ERROR

ON ERROR

This command allows the user to decide what action is to be taken if an ERROR occurs in a BASIC program.

ON ERROR may be followed by any valid statement, or series of statements, including GOTOs and GOSUBs. Thus the user may trap errors and redirect the program execution as desired. The command is cancelled by direct mode operation, and so is suitable only in program.

-continued overleaf-

The example below refuses alphabetic characters and prompts the user for numbers:

```
100 ON ERROR GOSUB 700
110 INPUT X
120 PRINT X'
130 GOTO 110

700 PRINT"NUMBER ONLY!"';RETURN
```

The space separating the two words ON and ERROR is mandatory. It may be useful to know that when an error occurs the number of the line where it happened is stored in locations 1 and 2,and the ERROR number in location 0 .

If you want the program to print the standard error message then execute !#10=#C9E7;LINK#C9DC

OUT

OUT

This command causes all characters which pass through the print routine to also be sent to the cassette socket as the serial transmission standard RS 232. Since most industry standard printers are configured to this standard, this feature allows both RS 232 and Centronics parallel printers to be driven from the ATOM.

Full handshake is provided, and two of the normally unused control codes (16 and 17) are used to enable and disable (respectively) the output.Alternatively <CTRL> <P> and <CTRL> <Q> can be used. Two parameters are passed with the command,seprated by a comma. The first defines the BAUD rate, and the second determines how line feeds are handled.

The first number is BAUD rate, with a default of 1200 BAUD. Other rates are selectable as follows:

FIRST NUMBER	BAUD RATE	
1	2400	
2	1200	(defualt value)
4	600	
8	300	
16	150	
32	75	etc.

The second number defines the number of line feeds sent to the printer for each line feed character encountered in the code being transmitted. Some printers will supply their own line feed after a Carriage Return code, and these will require no line feeds to be sent. A value of 0 will send no line feeds; a value of 2 will add an extra line feed to the printed one. It is now possible to use auto-line feed printers, and to elect double or treble spacing of printout to inprove readibility.

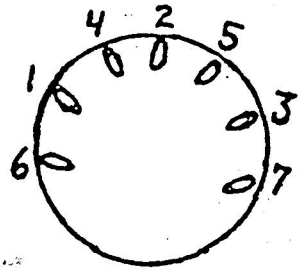
-continued overleaf-

```
OUT                gives          1200 BAUD,1LF per LF
OUT 1                             2400 BAUD,1LF per LF
OUT ,0                            1200 BAUD,NO line feeds
OUT 4,3                           600  BAUD,3 LF per LF
```

A command to list a program might be OUT;P.$16 LIST P.$17

The OUT command formats the RS 232 output stream as seven data bits with even parity, and the appropriate start and stop bits. The 'standard' voltage swing is plus and minus 15 V. However, most printers work from TTL, which operates from 0 to 5V. OUT delivers TTL voltages.

If the handshake is not used, then pin 4 should be tied to 5 Volts (such as pin 2 of edge connection PL4) through a 1K resistor.



CABLE side of PLUG

PIN	FUNCTION
1	Output to tape
2	GROUND
3	Input from tape
4	Handshake for RS232 through 1K resistor
5	unused
6	Serial Output for RS 232

PAGE

PAGE

This is a very convenient way of moving text areas, and is directly equivalent to executing ?18= NN followed by NEW. That is, it allows the programmer to move the current text space so that diffefernt programs or sections of the same program can be located in different areas. The syntax is PAGE NN, where NN is the page number in Hexadecimal (8200 is the start of page 82, 8C00 the start of page 8C etc.). Page 29 is the normal value in the expanded ATOM. To examine programs that are created using the PAGE command, see AULD.

PULL

PULL

Cancel the current program loop.

This command can only be used in a program, and will cancel one level of a DO/UNTIL , FOR/NEXT , or GOSUB/RETURN, so that one may jump out of these loops, leaving them 'open' and not corrupt the nest level counters. The word PULL must be followed by one letter to specify the type of loop being cancelled. Thus

```
PULL N  cancels        FOR/NEXT loop.
PULL U                 DO/UNTIL loop.
PULL R                 GOSUB/RETURN loop.
```

An attempt to pull a loop when there is no loop gives a "NO LOOP" error. Try this program:

```
100 FOR I= 1 TO 15
110 PRINT"LEVEL ",I'
120 GOTO 140
130 NEXT I
140 PRINT"JUMPED OUT OF LOOP"';GOTO 100
```

This gives an error when the number of loops left open exceeds 11. However, we can change line 140, and then it will work:

```
140 PULL N;PRINT"JUMPED OUT";GOTO 100
```

And this will spin all day long without error.

READ

READ

This enables words, numbers, or expressions which can be evaluated to be stored in the program and be available whenever the program is RUN. This is achieved by storing the numbers or words in DATA statements anywhere in the program, and when ready, READ them into numeric or string variables. See DATA and RESTORE.

The Data must contain the correct sequence of numeric and string data, such that string data is never assigned to a numeric variable. Data may be read into any variable type, including arrays. Examples:

```
100 DIM AA(9); RESTORE
110 FOR X=0 TO 9
120 READ AA(X)
130 NEXT X
140 END
150 DATA 43,8,-7,12,8,X,11,0,8,4
```

Notice the value of X when it is read into the array!

```
100 DIM AA(5)
110 RESTORE;DIM B(-1)
120 FOR X=1 TO 5
130 READ $B
140 AA(X)=B
150 B=B+LEN(B)+1
160 NEXT X
170 F.I=1TO5;P.$AA(I)';N.
180 END
190 DATA "THIS","IS","A","TEST","O.K.?"
```

Here 5 strings are READ. They are taken from the program and stored after the BASIC text area. The array AA is used to point at each string.

-continued overleaf-

Using the DATA from above, try:

```
100 DIM B(-1);RESTORE
110 DO; READ $B+LEN(B); PRINT $B'
120 UNTIL LEN(B) > 16
130 END
```

Another is:

```
100 DIM AA(5),B(4),C(10)
110 RESTORE
120 F.I=0TO5;READ AA(I);N.
130 READ %C;READ C?5;READ $B
140 F.I=0 TO 5;P.AA(I)';N.
150 F.I=0 TO 10;P. C?I';N.
160 FP.%C;P.$B;END
170 DATA 5,4,3,2,1,0,3.14159,89,"ATOM"
```

You may have noticed that any expression which can preceed an equals sign (=) may follow the READ command, such as

READ ?X or READ X or READ %AA(5) etc.

Always remember to RESTORE prior to the very first READ, in order to set the data pointer. Thereafter the READ statement will automatically sequence through the DATA (see DATA). RESTORE may be used freely to select different sets of DATA (see RESTORE). An attempt to read past the last piece of data in the last DATA statement results in an OUT OF DATA error.

REN

REN

Renumber : This allows the lines of any BASIC program to be renumbered with any desired starting value and step size. This is most usefel when tidying up a program on completion, or when you need to insert some new lines, but haven't left enough spare line numbers. REN then causes an automatic LIST for convenience, but this can be cancelled any time using the <ESC> key. The command can be followed by one or two numbers separated by a comma. The first is the starting line value, the second is the step size. If no numbers are given then default values are: start 100, renumber in steps of 10. ATOM always uses two bytes for line numbers, so no space is saved by using smaller line numbers. Examples:

REN	start at line	100, with step size	10
REN 200		200	10
REN 150,13		150	13
REN ,5		100	5

NOTE- GOTO's and GOSUB's are NOT renumbered, and the programmer is urged to use labels, which are also much faster.

RESTORE

RESTORE

The RESTORE command allows the data pointer to be moved from one set of DATA to another. Used without parameters following it, it will reset the data pointer to the first occurance of a DATA statement in the program. However, it may be followed by a line number, a label, or an expression in brackets which yields a line number (if it doesn't, then the data pointer is set to the next highest DATA statement). This command MUST be used prior to the first READ statement.The following are all valid:

RESTORE or RESTORE X or RESTORE[p]or RESTORE (3*Y+2) or RESTORE 5

An attempt to RESTORE to a line number past the last DATA statement results in an OUT OF DATA error. You are again advised to use labels rather than line numbers. See DATA and READ.

TAPE

TAPE

This is another command used to get data from tape when there is some difficulty. The command forces all data bytes to be stored in an area of memory, and also in the top half of the screen, so you can visually confirm the load. Since the data from tape also includes the titles, destinations, checksums, etc., These items can be examined and repaired if neccesary using the Hex or ASCII Dump/Edit facility.The repaired version is then COPYied to the correct area of memory.

The syntax is TAPE XXXX, where XXXX is a memory address in hexadecimal. All bytes fetched from cassette will be loaded to the area of RAM starting at this address. Hold down <ESC> to abort the function (it sometimes takes a few seconds).

Eg. TAPE 8200 or TAPE 3000 or TAPE 9600

TONE

TONE

This command is provided to play musical notes. The syntax is TONE X,$Y, where X is a number representing the duration of the note, and $Y is a string defining the pitch. The duration is in steps of 1/20 th of a second, and must be in the range 0 to 127. The string used to define the pitch must consist of at least two characters. The first is a number in the range 0 to 5 to define the octave, and the next defines the actual note, such as C ,B ,D etc. .Optionally, you may specify a sharp or flat using " + " for sharp and " - " for flat. Thus "3C+" is middle C sharp. All noted are chromatically related. Middle C is about 256 Hz., and the lowest note "0C" is about 32 Hz. The highest note is 5E. "R" means rest.

As well as playing notes on the ATOM's internal loudspeaker, the same audio is passed down the standard cassette output, and may be recorded directly or sent to an amplifier for big sound. In the example below, $P is pitch, S is speed, and D is duration.

```
100 DIM P3
110 S=5
120 RESTORE
130 FOR N=1 TO 11
140 READ D; READ $P
150 TONE (S*D),$P
160 NEXT N
170 DATA 1,"3C",1,"2A",3,"R",1,"2A"
180 DATA 1,"2B-",1,"3C",1,"3A",1,"R"
190 DATA 1,"3A",1,"R",3,"3F"
200 F.I=1 TO 60;WAIT;N.
210 GOTO 120
```

ZERO

ZERO

This assigns the value 0 to all integer variables. Some versions of BASIC do this automatically, but in ATOM the simple variables retain their assigned values (or random if unassigned) at all times. Whereas this can be useful, it is sometimes convenient to reset all integer variables. It may be used in direct mode or in program.

PART II
MACHINE LEVEL FUNCTIONS

Five powerful functions, characterised by single shifted characters [D], [H], [A], [T], [X] are provided to assist those working at m/c code level.

[D] DISASSEMBLE/EDIT

[D] DISASSEMBLE/EDIT

The syntax of the command is D X,Y , where X and Y are the Hexadecimal start and finish address of the code to be disassembled. Thus, typing [D]F396,F39B will give the following display:

```
DF396  20  91  F2      JSR #F291
DF399  C9  3A          CMP@ 3A    .:
DF39B  F0  91          BEQ #F32E
```

Note the following in this display:

a)The hex address of the start of the instruction is shown first, preceeded by a [D] to denote the mode.
b)The hex values of the instruction are then displayed (up to 3 bytes, and these may be edited (see EDITING).
c)The meaning of the instruction, expressed as assembly mnemonic, and any relevant data is shown.
d)where the instruction is of immediate mode, then the ASCII character of the immediate value is displayed, preceeded by a full stop.

e)Branch mnemonics have the target address displayed.
f)The first line displayed has cleared the typed-in instruction. This is to aid editing. See EDITING.

Alternatively, only the start address need be supplied, and then disassembly is then continued by holding down the <REPT> key. This mode is cancelled by the <ESC> key.

[H] HEXADECIMAL DUMP/EDIT

[H] HEXADECIMAL DUMP/EDIT

This function displays the contents of RAM or ROM in hexadecimal, and can be used to modify RAM contents. The syntax for the command is [H]X,Y , where X and Y are the start and finish addresses of the memory contents to be displayed. Alternatively, only the start address can be supplied, then pressing the <REPT> key continues the dump. <ESC> exits the mode. This function may be used to input values to memory. See EDITING.

[A] ASCII DUMP/EDIT

[A] ASCII DUMP/EDIT

This command is essentially the same as HEX DUMP, but instead of displaying the Hex value of the memory, the equivalent ASCII character is shown preceeded by a full-stop. If the value in memory is outside the range 32-127 then it is not an ASCII character, and the hex code will be shown. See EDITING.

EDITING

EDITING

All the above modes will display memory contents as either a two-digit hex number (one byte), or its ASCII equivilent, in which case it will appear with a full stop in front (e.g. 41 will appear as .A in an ASCII Dump). To change the memory contents, hit ESC, and the prompt > will return. Move the cursor over the line you want to edit, then COPY to the point on the line where you want to make the change. You may then type in EITHER the ASCII equivalent with a dot in front OR the two digit hex number, and this may be done as many times as you wish along the line. At the end of the line hit RETURN and ESC. DO NOT edit more than one line at a time without hitting RETURN and ESC. You need not go to the end of the line before hitting RETURN-the rest of the line will copy automatically.This method of editing is used in all three of the above modes.

[T] M/C CODE TRACE

[T] M/C CODE TRACE

This allows effective single stepping through a machine code program, and displays the contents of all registers. The syntax is T P, where P is the hex address of a machine code program. A,X,Y,Sp,S can be set before entry.A=Accumulator,X and Y=X and Y registers,Sp=stack pointer(always FF), S=status register. Thus [T]C2CA 00 36 47 FF 30 will commence at #C2CA with the following conditions:

-continued overleaf-

```
Accumulator=00
X Register =#36
Y Register =#47
Stack Pointer =#FF (Sp is always FF regardless of input)
status Register=#30
```

Default is all zeros except Sp=FF. Type in the command and hit <CR>, then <SHIFT> executes the next instruction. However, a JSR will be treated as a single command, and the actions in that subroutine will not be seen. Pressing <REPT> shows the actions in the subroutine (! these may be tortuous !).The top of the screen displays the contents of all the registers and all the flags, plus the ASCII equivilent of Accumulator contents.

[X] EXPANSION [X] EXPANSION

Once defined, the command [X] provides the possibility of the user creating a new reserved word and function.

The programmer first provides a machine code routine to handle the new function at any suitable RAM location, then

```
! #180=#ABCD
```

is executed. This effectively tells the interpreter to hand over control to the m/c code program located at ABCD whenever a command word begins with [X]. The example below defines [X] as a command to fill the screen area with the value which follows the command:

```
100 DIM JJ1
110 FOR X+0 TO 1                    TWO PASSES
120 P= #3B00                        SET ASSEMBLY AREA
130 ! #180=P                        POINT [X] AT IT
140 [                               START ASSEMBLY
150 JSR #C8BC                       READ VALUE AFTER [X] TO
                                    WORKSPACE STACK AT #16
160 JSR #C4E4                       CHECK NO NONSENSE AFTER
                                    COMMAND-ONLY ; OR <CR>
170 LDA@ 0; STA 4                   RESET W/S STACK POINTER
180 LDA #16;LDX@ 0                  GET VALUE FROM W/S
190:JJ0
200 STA #8000,X                     STORE IN ALL SCREEN
210 STA #8100,X                     MEMORY
220 INX; BNE JJ0
230 JMP #C55B                       BACK TO INTERPRETER
240];NEXT;END
```

To use this in a BASIC program:

```
100 FOR A=0 TO 255
110 [X]A
120 F.X0TO60;WAIT;N.
130 NEXT A
140END
```

Be sure to space the [X] command away from the line number, so it is not be mistaken by the interpreter as a label.