# NASCOM

# ROM

# BASIC

# DIS-ASSEMBLED

# PART I

BY CARL LLOYD-PARKER

A description of BASIC's usage of the memory

*** The work space RAM from 1000 to 10F8 ***

| Name | Addr | What it is used for |
|---|---|---|
| WRKSPC | 1000 | Jump to warm start BASIC |
| USR | 1003 | Jump for user defined function "USR(X)". This is initialised to give "?FC Error". |
| OUTSUB | 1006 | Skeleton for output to port "n" as the 8080 does not have the "OUT (C),r" instruction. The port "n" is loaded into 1007. |
| DIVSUP | 1009 | Skeleton subtraction routine for division. The dividend, divisor and quotient cannot all be held in the registers therefore the divisor is loaded into this routine so that there are sufficient registers for the dividend and quotient. |
| SEED | 1017 | 3 byte seed for random number generator. |
| | 101A | Table of floating point values used by RND. The seed is used to find which value of the eight to multiply the last random number by. |
| LSTRND | 103A | Where the last random number "RND(0)" is kept. |
| INPSUB | 103E | Skeleton for input from port "n" as the 8080 does not have the "IN r,(C)" instruction. The port "n" is loaded into 103F. |
| NULLS | 1041 | Number of nulls to output after carriage return. This value is set by the "NULLS n" command. |
| LWIDTH | 1042 | Width of terminal. This is set by "WIDTH n" command. |
| COMMAN | 1043 | Width of terminal for printing with commas. Why this has to be a seperate byte I don't know, however, the "WIDTH n" command sets LWIDTH but does NOT set this value, this has an irritating result when using BASIC with a printer and trying to have more than three columns using commas in "PRINT" statements, what ever you set "WIDTH" to you don't get more than three columns! This can be overcome in a simple way: "POKE 4163,252" - This makes "WIDTH" work correctly. |
| NULFLG | 1044 | Nulls after input byte flag. This is a flag that is examined and then zeroed by the teletype line input routine (TTYLIN). If the character input routine sets this flag before returning the input character then a null is output before the character. The only use I can think of for this is for VERY slow terminals which need to be "woken up" before a character is sent to them! |

CTLOFG 1045 Control "O" (Disable output) flag.
If this flag is set then no output will not appear at the terminal. This flag is flipped every time control "O" is typed from the keyboard.

LINESC 1046 "LINES" counter.
This is initially set to the value in LINESN and decremented after every line. If this value reaches zero then it is loaded with the value in LINESN and BASIC waits for a character from the keyboard.

LINESN 1048 "LINES" number.
This is set by the "LINES n" command for the number of lines to be LISTed at a time.

CHKSUM 104A Check sum for array loading and saving.
This accumulates the value of all the bytes in the array so that errors can be detected during "CLOAD*"

NMIFLG 104C Non-maskable interrupt flag.
When an NMI is received this flag is set to let BASIC know that the break was caused by an NMI.

BRKFLG 104D Break flag.
This flag is set by the input routine to let BASIC know that the break key was pressed.

RINPUT 104E Reflection for "INPUT" routine.
When an "INPUT" instruction is encountered BASIC jumps to the input routine via this jump. This value may be changed by the user as in the NASCOM BASIC manual. They go to the following:-

| | | |
|---|---|---|
| "DOKE 4175,-25" | Output CR,LF and get a line | (CRLIN) |
| "DOKE 4175,-6670" | No CRLF, get a line | (GETLIN) |
| "DOKE 4175,-6649" | Get a line by character | (TTYLIN) |

POINT 1051 Reflection for "POINT (X,Y)" function (Unused!)
This contains a jump to the "POINT (X,Y)" routine so that the user SHOULD be able to change it to be some other function. However, due to a "mis-firing" in someone's brain when the "POINT" routine was added, the function driver (FNOFST) tests for "POINT" and jumps DIRECTLY to the "POINT (X,Y)" routine!

PSET 1054 Reflection for "SET (X,Y)" routine.
This contains a jump to the "SET (X,Y)" routine in the same fashion as in "POINT". The "SET (X,Y)" routine is called via this reflection so that the user CAN change it to add more routines to BASIC.

RESET 1057 Reflection for "RESET (X,Y)" routine.
This contains a jump to the "RESET (X,Y)" as in PSET.

STRSPC 105A Start of string space pointer.
This contains a pointer to the start of string space. It is initially set to 50 bytes below the end of memory but can be changed by the "CLEAR n" statement.

LINEAT 105C Current line number.
This contains the value of the current line being executed. If a direct statement is being executed then this is -1 and if "Memory size?" is being asked it contains -2. This value is examined by the ERROR routine to see if a line number has to be printed or not. If this value is -2 then the error must have occurred in reply to "Memory size?" so BASIC is cold started again.

BASTXT 105E BASIC program text origin.
This is a pointer to where the BASIC program is stored in memory. It usually points to 10FA which is the usual start of a BASIC program. This can be changed if a program is located somewhere else in memory such as in ROM.

BUFFER 1061 Terminal input buffer.
This is a 72 character buffer where all input from the keyboard is to be stored such as commands, BASIC lines and "INPUT".

CURPOS 10AB Cursor position.
This contains the current value returned by "POS (X)" and it is incremented every time a character is printed. When a new line is started this value is zeroed, however, when a back space is printed it is NOT decremented and when "SCREEN X,Y" is done it is not set to the new "X". This can cause spurious CRLFs to be output when "SCREEN" and back spaces are used. A cure for this is to set "WIDTH 255".

LCRFLG 10AC Locate / Create variable flag.
This is used by the variable search routine to tell if it is in a "DIM" statement or not so that it knows if it has to locate or create the specified array.

TYPE 10AD Data type flag.
This flag contains the "type" of the current expression.
That is:- zero = Numeric, non-zero = string.

DATFLG 10AE Literal statement flag.
This flag is used to tell BASIC that it is pointing at a literal statement such as a string in quotes or a "REM" or "DATA" statement.

LSTRAM 10AF Last available RAM pointer.
This contains the address of the highest location in RAM that BASIC will use. It can be changed by the "CLEAR,n" statement.

TMSTPT 10B1 Temporary string pointer.
This contains a pointer into the temporary string pool.

TMSTPL 10B3 Temporary string pool.
This is a store of 4 temporary strings that were created by such things as "LEFT$","CHR$" and string concatenation.

TMPSTR 10BF Temporary string.
This string block for the current string being constructed. All string blocks are four bytes long. The first byte gives the length of the string. The second byte is not used and the third and fourth bytes form the address in memory where the string itself can be found.

STRBOT 10C3 Bottom of string space.
This contains a pointer to the bottom of the string area that is currently being used. Each time a new string is defined it is moved into the string area below this pointer and the pointer is adjusted to point to the new bottom of string area. If there is not enough room for the new string then a "garbage collection" is performed to remove all unused strings. If there still is not enough room then an "?OS Error" occurs.

CUROPR 10C5 Current operator address.
This contains the address of the current operator in EVAL so that the registers may hold other values without using the stack which is being used to hold sub-expressions.

LOOPST 10C7 Loop start address.
This contains the address of first statement in the FOR loop which is being constructed. This address is later moved into the FOR block on the stack.

DATLIN 10C9 DATA statement line number.
This contains the line number of the current position of the DATA statement pointer. It is used by DATSNR to tell the user in which line the bad DATA occurred.

FORFLG 10CB FOR / FN / array flag.
This contains a flag as to what GETVAR must find.
A value of 00H means find a variable or array element.
A value or 01H means find an array name.
A value of 64H means find a variable only.
A value of 80H means find an FN function.

LSTBIN 10CC This flag is set when ever any input is made into BUFFER.
RETURN first tests to see if the GOSUB was a direct statement and if so checks this flag, if this flag is set then an INPUT statement has occurred within the subroutine and so as far as RETURN is concerned, BUFFER now contains garbage so it jumps back to command mode.

READFG 10CD READ / INPUT flag.
This contains a flag to tell the READ/INPUT routine if it is processing a DATA statement or data for an INPUT statement. If this value is zero then INPUT is active else READ is.

BRKLIN 10CE Break line point.
This contains the address in the line where break occurred. It's value is saved so that CONT knows where to continue.

NXTOPR 10DO Next operator address.
This contains a pointer into the expression being evaluated by EVAL. It is used to keep track of where it is in the string.

ERRLIN 10D2 Line number of break.
This contains the line number of the line where the break occurred. It is saved so that CONT knows what line it is in.

CONTAD 10D4 Continue address.
This contains the address of the statement where CONT will continue.

******** Values from here on are saved during CSAVE ********

PROGND 10D6 End of program.
This contains the address of the byte after the end of the BASIC program text.

VAREND 10D8 End of variables.
This contains the address of the byte after the last variable.

ARREND 10DA End of arrays.
This contains the address of the byte after the last array.

NXTDAT 10DC Next DATA item.
This contains the address of the next item of DATA to be READ.

FNRGNM 10DE FN argument name.
This contains the name of the argument for the current FN function. If an FN function calls another FN function then this name is saved on the stack.

FNARG 10E0 FN function argument.
This is the floating point value of the the current FN function's argument. If an FN function calls another FN function then this value is saved on the stack together with the name of the FN argument.

FPREG 10E4 Floating point register.
This is a floating point number for the current value. It is a four byte store using 24 bit normalised sign and magnitude representation for the mantissa and excess 128 representation for the exponent of two.
Example of the number 35.25 in floating point:-

35.25 in binary is 100011.01
Which is the same as 100011.01 *2^ 00000000

The binary point is moved so that it precedes the first "1"
This gives .10001101

The point was moved left 6 times dividing the number by 2^6 so 6 must be added to the exponent to re-multiply by 2^6.
This gives .10001101 *2^ 00000110

As the bit after the point is ALWAYS "1" this bit can be used to store the sign of the number "0" for +ve, "1" for -ve
So +35.25 is stored as .00001101 * 2 ^ 00000110
Which in 24 bits is .000011010000000000000000 *2^ 00000110

128 is added to the exponent so that overflows and underflows can be more easily detected.
So the whole number in binary is:-
00001101 00000000 00000000 10000110
Which is 0 D 0 0 0 0 8 6 HEX
The bytes of the mantissa are stored in reverse order.
This gives the value for +35.25 as 00 00 0D 86
And -35.25 would be stored as 00 00 8D 86

SGNRES 10E8 Sign of result.
This contains the sign of the result for multiplication. Both values to be multiplied are tested and if their signs are different then the product will be negative otherwise it will be positive. The sign for the product is stored here so that it can be tested after to make the result correct.

PBUFF 10E9 Number print buffer.
When a floating point number has to be converted into ASCII for PRINT or STR$ the ASCII number is built up in this buffer by NUMASC so that it can be output or assigned to a string.

NULVAL 10F6 Multiply value.
This contains the 24 bit multiplier because there are not enough registers to hold the multiplier, multiplicand and product all at the same time.

PROGST 10F9 Program start.
This is the byte before the first line in the program. It must be zero to tell the execution driver that the next (actually the first) line is to be executed.

******** How a program is stored in memory ********

Example:- The program:-

```
  10  FOR A=1 TO 5:PRINT A,SQR(A):NEXT A
  20  END
```

is in memory, it would look like this:-

```
PROGND  111C    Pointer to byte after program

10FA    15 11   Pointer to next line (1115)
10FC    0A 00   Line number (10)
10FE    81      FOR token
10FF    20      Space
1100    41      A
1101    B4      = token
1102    31      1
1103    20      Space
1104    A6      TO token
1105    20      Space
1106    35      5
1107    3A      :
1108    9E      PRINT token
1109    20      Space
110A    41      A
110B    2C      ,
110C    BA      SQR token
110D    28      (
110E    41      A
110F    29      )
1110    3A      :
1111    82      NEXT token
1112    20      Space
1113    41      A
1114    00      End of line
1115    1A 11   Pointer to next line (111A)
1117    14 00   Line number (20)
1118    80      END token
1119    00      End of line
111A    00 00   Pointer to next line (0000 = End of program)
111C
```

******** How variables and arrays are stored ********

Variables such as AB AB$ and FN AB are all stored in the simple variable area of memory. The start address of this area is held in PROGND and the end address is held in VAREND.

If AB=10 and AB$="TEXT" and FN AB(XY) had been defined then the memory would look like this

```
42 41           Name of AB in reverse (42 41 = "B" "A")
00 00 20 84     Floating point value for 10

C2 41           Name for AB$ (C2 is "B" with bit 7 set)
04              Length of string (4 characters)
??              This byte unused for string
LL HH           Address where "TEXT" is to be found

42 C1           Name of FN AB (C1 is "A" with bit 7 set)
LL HH           Address of function (After "=")
59 58           Argument name in reverse (59 58 = "Y" "X")
```

Arrays such as AB(1,3) and AB$(3,1) are stored in the array area of memory. The start address of this area is held in VAREND and the end address is held in ARREND.

If DIM AB(1,3),AB$(3,1) had been entered then the memory would look like this:-

```
42 41           Name of array AB in reverse
25 00           Bytes used for array (0025 = 37)
02              2 dimensions
04 00           Size of second dimension including zero element
02 00           Size of first dimension including zero element
00 00 00 00     AB(0,0)
00 00 00 00     AB(1,0)
00 00 00 00     AB(0,1)
00 00 00 00     AB(1,1)
00 00 00 00     AB(0,2)
00 00 00 00     AB(1,2)
00 00 00 00     AB(0,3)
00 00 00 00     AB(1,3)

C2 41           Name of array AB$ in reverse
25 00           Bytes used for array (0025 = 37)
02              2 dimensions
02 00           Size of second dimension including zero element
04 00           Size of first dimension including zero element
00 00 00 00     AB$(0,0)
00 00 00 00     AB$(1,0)
00 00 00 00     AB$(2,0)
00 00 00 00     AB$(3,0)
00 00 00 00     AB$(0,1)
00 00 00 00     AB$(1,1)
00 00 00 00     AB$(2,1)
00 00 00 00     AB$(3,1)
```

******** Usage of the stack for GOSUB/RETURN and FOR/NEXT ********

GOSUB and RETURN usage of the stack

When a GOSUB is executed the address of where to RETURN to and the number of the line to RETURN to are PUSHed on the stack as follows:-

```
HIGH MEMORY:    XX XX   Address of where to RETURN to
                XX XX   Number of line to RETURN to
STACK POINTER:  8C      GOSUB token as marker
```

This GOSUB block remains on the stack until a RETURN is executed at which point BASIC looks back through the stack until it finds a GOSUB block and then sets the stack there and then POPs the line number and the address of the statement after the GOSUB and continues execution.

The fact that the stack is set to this GOSUB block kills all active FOR loops which were set up inside the subroutine.



FOR and NEXT usage of the stack

When a FOR is executed the address of the first statement in the loop, the line number of the loop statement, the TO value, the STEP value and the sign of the STEP are all PUSHed onto the stack as follows:-

```
HIGH MEMORY:    XX XX           Address of first statement in loop
                XX XX           Line number of loop statement
                XX XX XX XX     TO value in floating point
                XX XX XX XX     STEP value in floating point
                XX              Sign of STEP
                XX XX           Address of index variable
STACK POINTER:  81              FOR token as marker
```

This FOR block remains on the stack until a matching NEXT is executed. When next is executed BASIC looks back through the stack to find the matching FOR block. If it is found then the STEP value is added to the value of the index variable and the result is compared with the TO value. With the use of the TO value and the sign of the step BASIC knows if the loop has been completed or not. If it has not been completed then the FOR block remains on the stack until the loop has been completed. The stack is set to point to this FOR block and effectively kills all FORs nested within this loop. When the loop is completed the FOR block is removed from the stack and execution continues from after the NEXT instruction.

If the FOR block cannot be found then a "?NF Error" occurs.

******** Now for the exciting bit - The ROM ********

| Name | Addr | What the routine does |
|---|---|---|
| INIT | E019 | Copy the initial work space conditions into the work space RAM at 1000 to 105F. |
| MSIZE | E036 | Output "Memory size" prompt and get response from user. If a number is given jump to TSTMEM otherwise start at the beginning of RAM and test each successive byte until the end of RAM is found. When the end has been found jump to SETTOP. |
| TSTMEM | E05B | Get supplied address. If any bad characters output "?SN Error" and re-initialise otherwise try to write a D9H byte into that address. If D9H is not read back then there is no RAM at that address so go to MSIZE to ask again. If D9H is read back then drop through to SETTOP. |
| SETTOP | E06D | Test for minimum memory requirement and if not enough jump to MSIZE to ask again otherwise save this address as the highest available memory location, allocate 50 bytes for string space and output the sign on message and the number of bytes free. Then drop through to WARMST. |
| WARMST | E0AE | Clear run-time registers and jump to PRNTOK. |
| BAKSTK | E356 | Look back through stack for FOR or GOSUB blocks and exit with HL pointing to the block. |
| CHKSTK | E38A | Check for "C" levels of stack space and output "?OM Error" if not enough stack space. |
| DATSNR to TMERR | E3A7 to E3BF | I like this section of code because you jump into the routine at one of the entry points, this loads the "E" register with the error number and the "LD BC,nn" opcode (01H) skips all the following "LD E,n" instructions. It is very efficient. |
| ERROR | E3C1 | Output error code for error number in the E register, output the line number if needed and drop through to PRNTOK. |
| PRNTOK | E3F8 | Output "Ok" message and drop through to GETCMD. |
| GETCMD | E405 | Get a direct statement or BASIC line, CRUNCH the text and if it is a direct statement go to EXCUTE to execute it. If it is a BASIC line then move it to the program text area. |
| SRCHLN | E499 | Search for line number in register DE. |
| NEW | E4B9 | Set PROGND to start of program text area to effectively remove program from memory and execute the null program to clear all the other pointers. |
| RUNSFT | E4C5 | Execute program from fitst statement. |
| INTVAR | E4C9 | Clear out variables and string space an reset DATA pointer. |

| | | |
|---|---|---|
| CLREG | E4DF | Reset stack to top of memory, clear out temporary strings, disable CONTinue and execute program. |
| PROMPT | E4FC | Output "? " prompt for INPUT and go to get input line. |
| CRUNCH | E509 | Move text string from HL to BUFFER crunching reserved words into tokens as you go. |
| ENDBUF | E5B8 | Mark the end of BUFFER with three nulls. |
| GETLIN | E5F2 | Get an input line from monitor. |
| TTYLIN | E607 | Get an input line by character and save it in BUFFER. |
| CPDEHL | E68A | Compare the DE register with the HL register returning flags:- Z - Registers are equal, C - DE > HL and NC- DE <= HL |
| CHKSYN | E690 | Make sure the next byte in the code string is the sam as the byte following the "CALL CHKSYN". If not - output "?SN Error". |
| OUTC | E69B | Output character in the A register to the terminal and output a CRLF if the current terminal width is exceeded. Character is not output if CTLOFG is set. |
| CLOTST | E6CC | Get an input character and flip CTLOFG if it is control "O". |
| LIST | E6DD | LIST BASIC program from start (or specified line number) LINESN lines at a time. It does not take into account that graphics may appear in quotes, DATA or REM statements so the graphic character is treated as a reserved word token and expanded to the complete word. |
| FOR | E779 | Assign the initial value to the index variable and then set up a FOR block on the stack. |
| RUNCNT | E7F2 | Main execution driver loop. This gets the next statement (May be after ":" or on next line) and gives it to EXCUTE. |
| EXCUTE | E816 | Test current statement. If it is a key word then find the address of the routine from WORDTB and call it. If it is a letter then call LET to try to assign a variable. Otherwise generate "?SN Error". |
| GETCHR | E836 | Get next character in code string and return flags:- Z - End of statement (":" or null) C - Character is a digit otherwise NC. |
| RESTOR | E846 | RESTORE routine. Set DATA pointer to start of program or to specified line if a line number is given. |
| TSTBRK | E861 | Test for break key. If pressed test for control "S" (pause). If another break pressed then stop execution and output break message. |
| STOP | E870 | STOP routine. Flag "STOP" an join END. |

| | | |
|---|---|---|
| END | E872 | END routine. Flag "END".<br>Save address of break and set continue address and line.<br>If from STOP or break then output "Break" message and if not a direct statement then output line number also. |
| CONT | E89E | CONT routine. Get continue address, if it is zero then output "?CN Error" else set address and line as current and continue. |
| NULL | E8B1 | NULL routine. Set number of nulls to be output after CRLF. |
| CHKLTR | E977 | Test next character in code string and return flags:-<br>NC - Letter, C - Not a letter. |
| FPSINT | E97F | Get an integer 0 to 32767 from next character. |
| POSINT | E982 | Get an integer 0 to 32767 to DE. |
| DEPINT | E985 | Test for integer 0 to 32767 and leave it in DE. |
| DEINT | E98B | Test for integer -32768 to 32767 and leave it in DE. |
| ATOH | E9A5 | Read an ASCII integer 0 to 65529 into DE from code string. |
| CLEAR | E9CA | CLEAR routine. "CLEAR" Clear variables,<br>"CLEAR s" also reserves string space (where t is 0 to 32767),<br>"CLEAR,t" sets top of memory (where t is 0 to 32767),<br>"CLEAR s,t" reserves string space and sets RAM top. |
| RUN | EA10 | RUN routine. "RUN" run from start of program, "RUN n" run from specified line. |
| GOSUB | EA1C | GOSUB routine. Create a GOSUB block on the stack and GOTO specified line. |
| GOTO | EA2D | GOTO routine. Get ASCII line number and continue execution from that line. |
| RETURN | EA4B | RETURN routine. Look back through stack for a GOSUB block and if one is found POP line number and address in line and continue execution from there. Otherwise "?RG Error" occurs. |
| DATA | EA70 | DATA routine. Flag DATA and drop through to common code. |
| REM | EA72 | REM routine. Flag REM and look for end of statement, DATA statement ends with ":" or null, REM statement ends only with null (End of line). |
| LET | EA87 | LET routine. Get variable name, make sure "=" follows, evaluate expression and assign it to the variable. |
| ON | EAE1 | ON GOTO / ON GOSUB routine. Get integer expression (0 to 255). Look through list of ASCII line numbers until the Nth value is found. If value N exists GOTO or GOSUB that line. Otherwise drop through to next statement. |

| | | |
|---|---|---|
| IF | EAFF | IF routine. Evaluate expression. If it is zero then dop through to the next line. Otherwise GOTO line number if specified else go to execute the statement. |
| PRINT | EB23 | PRINT routine. |
| INPUT | EBFD | INPUT routine. Test for direct mode, if in direct mode then no INPUT allowed ("?ID Error"). Otherwise output prompt string if one is given, Get an input line and join READ routine to assign values to variables. |
| READ | EC2C | READ routine. Get address of next DATA item and flag READ. Read in items an assign to variables (used for INPUT and READ). If any errors occur use READFG to see if:- "?SN Error" or "?OD Error" are needed for READ or "Redo" or "Extra ignored" are needed for INPUT. |
| FDTLP | ECD2 | Find next DATA statement. |
| NEXT | ECF6 | NEXT routine. Look back through stack for a matching FOR block. If one is found then add STEP to index variable and compare it with the TO value. If the loop has not been completed then continue execution from the first statement in that loop. Otherwise remove the FOR block from the stack. Then look for another variable (Eg "NEXT B,A"). If found then re-enter NEXT routine to process this loop. Otherwise continue execution from the statement after the NEXT. |
| GETNUM | ED41 | Get a numeric expression. |
| TSTNUM | ED44 | Make sure current value is numeric ("?TM Error" if not). |
| TSTSTR | ED45 | Make sure current value is a string ("?TM Error" if not). |
| CHKTYP | ED46 | Make sure current value is of the type selected by TYPE. |
| EVAL | ED5A | Evaluate an expression using algebraic logic (Do multiplication and division before addition and subtraction. Etc.). Leave the result in FPREG and set TYPE to string or numeric. |
| FNOFST | EE33 | Enter with offset into FNCTAB. If it is "POINT (X,Y)" then jump directly to "POINT (X,Y)" routine (This SHOULD have been a jump via the POINT reflection). If it is LEFT$, RIGHT$ or MID$ then evaluate the string expression and get the "," after it then go to the string function. If it is any other function then get its address from FNCTAB and call it. |
| SGNEXP | EE70 | Get sign of number and return in the D register:- FF - Negative, 00 Positive. |

| | | |
|---|---|---|
| DIM | EF28 | DIM routine. Flag "create" variable and join common [illegible]. |
| GETVAR | EF2D | Look for specified variable. Use FORFLG to:-<br>1. Not allow subscripted FOR loops.<br>2. Look for an FN function definition.<br>3. Look for an array name for CLOAD* and CSAVE*.<br>4. Or just look for the variable as supplied. |
| FRE | F0D0 | FRE(X) routine. If X is a numeric expression then return the amount of memory between ARREND and the stack. If X is a string expression then do a "Garbage collection" and return the number of unused bytes in the string area. |
| POS | F0FE | POS(X) routine. Return the current cursor position (CURPOS). |
| DEF | F106 | DEF FN routine. Define a user defined function.<br>This is not legal in direct mode. |
| DOFN | F133 | Call an FN function. Save any previous argument name and value on the stack, Evaluate the function an then restore the previous argument name and value. |
| IDTEST | F17B | If in direct mode then output "?ID Error" else return. |
| CHEKFN | F189 | Make sure "FN" follows and flag find FN function definition. |
| STR | F19A | STR$ routine. Convert current numeric value to ASCII and create a temporary string for it. |
| SAVSTR | F1AA | Save current string in string area. |
| MKTMST | F1BF | Make a temporary string. |
| PRNUMS | F20F | Print number string at HL. |
| PRS | F210 | Print string at HL. |
| TESTR | F229 | Test if enough string space. If insufficient then GRBAGE collect and test again. If still insufficient space then output "?OS Error". |
| GRBAGE | F253 | Do a "Garbage collection" on string area.<br>Scan through all string variables and shift all active strings to the top of the string area thus leaving the free space in one block. |
| CONCAT | F306 | Concatenate two strings (Eg A$+B$). |
| GETSTR | F350 | Get a string routine. |
| LEN | F382 | LEN(X$) routine. |
| ASC | F391 | ASC(X$) routine. |
| CHR | F3A2 | CHR$(X) routine. |

| | | |
|---|---|---|
| LEFT | F3B2 | LEFT$(X$,X) routine. |
| RIGHT | F3E2 | RIGHT$(X$,X) routine. |
| MID | F3EC | MID$(X$,P[,L]) routine.<br>MID$(X$,P) returns string from character P to the end.<br>MID$(X$,P,L) returns string from position P for L characters. |
| VAL | F41C | VAL(X$) routine. |
| INP | F441 | INP(X) routine. Set up port number in work space and call INPSUB skeleton. Return value from port. |
| POUT | F44D | OUT P,N routine. Set up port number in work space and call OUTSUB skeleton to output value to the port. |
| WAIT | F453 | WAIT P,A,X routine. Set up port number in work space. Call INPSUB to get byte, XOR value with X if X is supplied and then AND the value with A. If this value is zero then go back to WAITLP until value is non-zero. |
| FNDNUM | F481 | Get next number from code string. |
| CSAVE | F4C3 | CSAVE routine. If CSAVE* jump to array save routine ARRSV1. Evaluate string expression for name and save program. |
| CLOAD | F4F9 | CLOAD routine. If CLOAD* jump to array load routine ARRLD1. See if CLOAD? and save status. Evaluate string expression for file name (None given - any file will do) and search for that file. When found load it into memory. |
| PEEK | F5A3 | PEEK(X) routine. |
| POKE | F5AA | POKE A,V routine. |
| ROUND | F5BB | Round number up by adding 0.5 to it. |
| ADDPHL | F5BE | Add floating point number at HL to the current value in FPREG. |
| SUBPHL | F5C4 | Subtract the current value from the value at HL and leave the result in FPREG. |
| PSUB | F5C8 | Subtract FPREG from the value on the stack and leave the result in FPREG. |
| SUBCDE | F5CA | Subtract FPREG from the value in BCDE and leave the result in FPREG. |
| FPADD | F5CD | Add the value in BCDE to the value in FPREG. |
| MINCDE | F60D | Subtract value in FPREG from value in BCDE. |
| NORMAL | F638 | Normalise value in BCDE. |
| PLUCDE | F672 | Add value in FPREG to value in BCDE. |
| COMPL | F67E | Complement value in BCDE. |

| | | |
|---|---|---|
| LOG | F6C7 | LOG(X) routine. Get LOG of FPREG.<br>Scale the number to be between 0 and 1.<br>Add SQR(1/2) to value.<br>Divide into SQR(2).<br>Subtract from 1.<br>Compute the sum of series using LOGTAB for coefficients.<br>Subtract 0.5 from result.<br>Re-scale the number and then multiply by LOG(2). |
| MULT | F706 | Multiply FPREG by value on stack. |
| FPMULT | F708 | MULTIPLY FPREG by value in BCDE. |
| DIV | F767 | Divide the value on stack by FPREG and leave the result in FPREG. |
| DVBCDE | F768 | Divide the value in BCDE by FPREG and leave the result in FPREG. |
| TSTSGN | F813 | Test sign of number in FPREG. |
| SGN | F822 | SGN(X) routine. |
| ABS | F838 | ABS(X) routine. |
| STAKFP | F844 | Move value in FPREG to stack. |
| PHLTFP | F851 | Move value at HL to FPREG. |
| FPBCDE | F854 | Move value in BCDE to FPREG. |
| BCDEFP | F85F | Move FPREG to BCDE. |
| LOADFP | F862 | Move value at HL to BCDE. |
| FPTHL | F86B | Move Value in FPREG to HL. |
| SIGNS | F879 | Set sign of result depending on signs of operands. |
| CMPNUM | F88E | Compare numbers. |
| FPINT | F8BB | Get integer of FP value. |
| INT | F8E6 | INT(X) routine. |
| ASCTFP | F91A | Convert ASCII floating point number into binary. |
| NUMASC | F9B8 | Convert floating point binary into ASCII. |
| SQR | FAAC | SQR(X) routine. Uses SQR(X) = X ^ 0.5 |
| POWER | FAB5 | Raise base BCDE to the power FPREG.<br>Uses X^Y = EXP( Y*LOG(X) ). |

| | | |
|---|---|---|
| EXP | FAFA | EXP(X) routine.<br>Scale value to be between 0 and 1<br>Compute sum of series using EXPTAB for coefficients.<br>Re-scale number. |
| SUMSER | FB5B | Sum the series using table of coefficients at HL.<br>(( N1 * X^2 + N2 ) * X^2 + N3 ) * X^2 ... + Nn |
| SUMSER1 | FB6A | Sum the series using table of coefficients at HL.<br>(( N1 * X + N2 ) * X + N3 ) * X ... + Nn |
| RND | FB8B | RND(X) routine. |
| COS | FC00 | COS(X) routine. Uses COS(X) = SIN(X+PI/2). |
| SIN | FC03 | SIN(X) routine.<br>Divide angle by 2*PI.<br>Get fraction part.<br>Move other quadrants around and set what result will be.<br>Sum the series using table of coefficients at SINTAB.<br>Adjust result. |
| TAN | FC67 | TAN(X) routine. TAN(X) = SIN(X) / COS(X). |
| ATN | F67C | ATN(X) routine.<br>If value > 1 then get 1/value and set PI/2 - angle.<br>Sum the series using table of coefficients at ATNTAB.<br>Negate result if original value was > 1. |
| WIDTH | FDA5 | WIDTH routine. Set terminal width but NOT commas width. |
| LINES | FDAD | LINES routine. |
| DEEK | FDBC | DEEK(X) routine. |
| DOKE | FDC7 | DOKE A,V routine. |
| SCREEN | FDE6 | SCREEN X,Y routine. |
| SCRADR | FE11 | Get screen address from row and column in DE and BC. |
| INLINE | FEE8 | Get an input line from NAS-SYS. |
| GETXYA | FF15 | Get (X,Y) for SET,RESET and POINT.<br>Return address on screen and a bit mask. |
| SETB | FF40 | SET(X,Y) routine. |
| RESETB | FF55 | RESET(X,Y) routine. |
| POINTB | FF79 | POINT(X,Y) routine. |
| XYPOS | FF96 | Convert (X,Y) to a row and column on screen. |

# NASCOM

# ROM

# BASIC

# DIS–ASSEMBLED

# PART 2

BY CARL LLOYD–PARKER

```
                        ; GENERAL EQUATES

0001                    UARTD   EQU     01H             ; UART data port
0002                    UARTS   EQU     02H             ; UART status port

0003                    CTRLC   EQU     03H             ; Control "C"
0007                    CTRLG   EQU     07H             ; Control "G"
0008                    BKSP    EQU     08H             ; Back space
000A                    LF      EQU     0AH             ; Line feed
000C                    CS      EQU     0CH             ; Clear screen
000D                    CR      EQU     0DH             ; Carriage return
000F                    CTRLO   EQU     0FH             ; Control "O"
0012                    CTRLR   EQU     12H             ; Control "R"
0013                    CTRLS   EQU     13H             ; Control "S"
0015                    CTRLU   EQU     15H             ; Control "U"
001A                    CTRLZ   EQU     1AH             ; Control "Z"
001B                    ESC     EQU     1BH             ; Escape
001C                    TBRK    EQU     1CH             ; "T" monitor break
001D                    TBS     EQU     1DH             ; "T" monitor back space
001E                    TCS     EQU     1EH             ; "T" monitor clear screen
001F                    TCR     EQU     1FH             ; "T" monitor carriage return
007F                    DEL     EQU     7FH             ; Delete

                        ; MONITOR LOCATIONS

0000                    MONSTT  EQU     0000H           ; Start of monitor
000D                    STMON   EQU     000DH           ; NAS-SYS initialisation
0051                    MFLP    EQU     0051H           ; Flip tape LED ("T")
008D                    MONTYP  EQU     008DH           ; Type of "T" monitor
03D1                    T2DUMP  EQU     03D1H           ; "T2" Dump routine
0400                    T4WR    EQU     0400H           ; "T4" Write routine
070C                    T4READ  EQU     070CH           ; "T4" Read routine

0800                    VDU     EQU     0800H           ; NASCOM Video RAM base

                        ; MONITOR WORK SPACE LOCATIONS

0C00                    PORTO   EQU     0C00H           ; Copy of output port 0
0C0C                    ARG1    EQU     0C0CH           ; Argument 1
0C0E                    ARG2    EQU     0C0EH           ; Argument 2
0C18                    TCUR    EQU     0C18H           ; "T" monitor cursor
0C29                    CURSOR  EQU     0C29H           ; NAS-SYS Cursor
0C2B                    ARGN    EQU     0C2BH           ; Number of ARGS
0C4A                    TOUT    EQU     0C4AH           ; "T" Output reflection
0C4D                    TIN     EQU     0C4DH           ; "T" Input reflection
0C75                    CIN     EQU     0C75H           ; NAS-SYS Input table
0C7E                    NMI     EQU     0C7EH           ; NAS-SYS NMI Jump
```

```
                ; BASIC WORK SPACE LOCATIONS

1000            WRKSPC  EQU     1000H           ; BASIC Work space
1003            USR     EQU     1003H           ; "USR (x)" jump
1006            OUTSUB  EQU     1006H           ; "OUT p,n"
1007            OTPORT  EQU     1007H           ; Port (p)
1009            DIVSUP  EQU     1009H           ; Division support routine
100A            DIV1    EQU     100AH           ; <- Values
100E            DIV2    EQU     100EH           ; <-    to
1012            DIV3    EQU     1012H           ; <-    be
1015            DIV4    EQU     1015H           ; <-inserted
1017            SEED    EQU     1017H           ; Random number seed
103A            LSTRND  EQU     103AH           ; Last random number
103E            INPSUB  EQU     103EH           ; "INP (x)" Routine
103F            INPORT  EQU     103FH           ; PORT (x)
1041            NULLS   EQU     1041H           ; Number of nulls
1042            LWIDTH  EQU     1042H           ; Terminal width
1043            COMMAN  EQU     1043H           ; Width for commas
1044            NULFLG  EQU     1044H           ; Null after input byte flag
1045            CTLOFG  EQU     1045H           ; Control "O" flag
1046            LINESC  EQU     1046H           ; Lines counter
1048            LINESN  EQU     1048H           ; Lines number
104A            CHKSUM  EQU     104AH           ; Array load/save check sum
104C            NMIFLG  EQU     104CH           ; Flag for NMI break routine
104D            BRKFLG  EQU     104DH           ; Break flag
104E            RINPUT  EQU     104EH           ; Input reflection
1051            POINT   EQU     1051H           ; "POINT" reflection (unused)
1054            PSET    EQU     1054H           ; "SET"   reflection
1057            RESET   EQU     1057H           ; "RESET" reflection
105A            STRSPC  EQU     105AH           ; Bottom of string space
105C            LINEAT  EQU     105CH           ; Current line number
105E            BASTXT  EQU     105EH           ; Pointer to start of program
1061            BUFFER  EQU     1061H           ; Input buffer
1066            STACK   EQU     1066H           ; Initial stack
10AB            CURPOS  EQU     10ABH           ; Character position on line
10AC            LCRFLG  EQU     10ACH           ; Locate/Create flag
10AD            TYPE    EQU     10ADH           ; Data type flag
10AE            DATFLG  EQU     10AEH           ; Literal statement flag
10AF            LSTRAM  EQU     10AFH           ; Last available RAM
10B1            TMSTPT  EQU     10B1H           ; Temporary string pointer
10B3            TMSTPL  EQU     10B3H           ; Temporary string pool
10BF            TMPSTR  EQU     10BFH           ; Temporary string
10C3            STRBOT  EQU     10C3H           ; Bottom of string space
10C5            CUROPR  EQU     10C5H           ; Current operator in EVAL
```

```
10C7            LOOPST  EQU     10C7H           ; First statement of loop
10C9            DATLIN  EQU     10C9H           ; Line of current DATA item
10CB            FORFLG  EQU     10CBH           ; "FOR" loop flag
10CC            LSTBIN  EQU     10CCH           ; Last byte entered
10CD            READFG  EQU     10CDH           ; Read/Input flag
10CE            BRKLIN  EQU     10CEH           ; Line of break
10D0            NXTOPR  EQU     10D0H           ; Next operator in EVAL
10D2            ERRLIN  EQU     10D2H           ; Line of error
10D4            CONTAD  EQU     10D4H           ; Where to CONTinue
10D6            PROGND  EQU     10D6H           ; End of program
10D8            VAREND  EQU     10D8H           ; End of variables
10DA            ARREND  EQU     10DAH           ; End of arrays
10DC            NXTDAT  EQU     10DCH           ; Next data item
10DE            FNRGNM  EQU     10DEH           ; Name of FN argument
10E0            FNARG   EQU     10E0H           ; FN argument value
10E4            FPREG   EQU     10E4H           ; Floating point register
10E7            FPEXP   EQU     FPREG+3         ; Floating point exponent
10E8            SGNRES  EQU     10E8H           ; Sign of result
10E9            PBUFF   EQU     10E9H           ; Number print buffer
10F6            MULVAL  EQU     10F6H           ; Multiplier
10F9            PROGST  EQU     10F9H           ; Start of program text area
115D            STLOOK  EQU     115DH           ; Start of memory test

                ; BASIC ERROR CODE VALUES

0000            NF      EQU     00H             ; NEXT without FOR
0002            SN      EQU     02H             ; Syntax error
0004            RG      EQU     04H             ; RETURN without GOSUB
0006            OD      EQU     06H             ; Out of DATA
0008            FC      EQU     08H             ; Function call error
000A            OV      EQU     0AH             ; Overflow
000C            OM      EQU     0CH             ; Out of memory
000E            UL      EQU     0EH             ; Undefined line number
0010            BS      EQU     10H             ; Bad subscript
0012            DD      EQU     12H             ; Re-DIMensioned array
0014            DZ      EQU     14H             ; Division by zero (/0)
0016            ID      EQU     16H             ; Illegal direct
0018            TM      EQU     18H             ; Type mis-match
001A            OS      EQU     1AH             ; Out of string space
001C            LS      EQU     1CH             ; String too long
001E            ST      EQU     1EH             ; String formula too complex
0020            CN      EQU     20H             ; Can't CONTinue
0022            UF      EQU     22H             ; UnDEFined FN function
0024            MO      EQU     24H             ; Missing operand
```

```
E000 C303E0     START:  JP      STARTB          ; Jump for restart jump
E003 F3         STARTB: DI                      ; No interrupts
E004 DD210000           LD      IX,0            ; Flag cold start
E008 C312E0             JP      CSTART          ; Jump to initialise

E00B 8BE9               DEFW    DEINT           ; Get integer -32768 to 32767
E00D F2F0               DEFW    ABPASS          ; Return integer in AB

E00F C33CE7             JP      LDNMI1          ; << NO REFERENCE TO HERE >>

E012 210010     CSTART: LD      HL,WRKSPC       ; Start of workspace RAM
E015 F9                 LD      SP,HL           ; Set up a temporary stack
E016 C3BBFE             JP      INITST          ; Go to initialise

E019 11DFE2     INIT:   LD      DE,INITAB       ; Initialise work space
E01C 0663               LD      B,INITBE-INITAB+3;Bytes to copy
E01E 210010             LD      HL,WRKSPC       ; Into workspace RAM
E021 1A         COPY:   LD      A,(DE)          ; Get source
E022 77                 LD      (HL),A          ; To destination
E023 23                 INC     HL              ; Next destination
E024 13                 INC     DE              ; Next source
E025 05                 DEC     B               ; Count bytes
E026 C221E0             JP      NZ,COPY         ; More to move
E029 F9                 LD      SP,HL           ; Temporary stack
E02A CDDFE4             CALL    CLREG           ; Clear registers and stack
E02D CD81EB             CALL    PRNTCR          ; Output CRLF
E030 32AA10             LD      (BUFFER+72+1),A ; Mark end of buffer
E033 32F910             LD      (PROGST),A      ; Initialise program area
E036 2103E1     MSIZE:  LD      HL,MEMMSG       ; Point to message
E039 CD10F2             CALL    PRS             ; Output "Memory size"
E03C CDFCE4             CALL    PROMPT          ; Get input with "? "
E03F CD36E8             CALL    GETCHR          ; Get next character
E042 B7                 OR      A               ; Set flags
E043 C25BE0             JP      NZ,TSTMEM       ; If number - Test if RAM there
E046 215D11             LD      HL,STLOOK       ; Point to start of RAM
E049 23         MLOOP:  INC     HL              ; Next byte
E04A 7C                 LD      A,H             ; Above address FFFF ?
E04B B5                 OR      L
E04C CA6DE0             JP      Z,SETTOP        ; Yes - 64K RAM
E04F 7E                 LD      A,(HL)          ; Get contents
E050 47                 LD      B,A             ; Save it
E051 2F                 CPL                     ; Flip all bits
E052 77                 LD      (HL),A          ; Put it back
E053 BE                 CP      (HL)            ; RAM there if same
E054 70                 LD      (HL),B          ; Restore old contents
E055 CA49E0             JP      Z,MLOOP         ; If RAM - test next byte
E058 C36DE0             JP      SETTOP          ; Top of RAM found
```

```
E05B CDA5E9     TSTMEM: CALL    ATOH            ; Get high memory into DE
E05E B7                 OR      A               ; Set flags on last byte
E05F C2ADE3             JP      NZ,SNERR        ; ?SN Error if bad character
E062 EB                 EX      DE,HL           ; Address into HL
E063 2B                 DEC     HL              ; Back one byte
E064 3ED9               LD      A,11011001B     ; Test byte
E066 46                 LD      B,(HL)          ; Get old contents
E067 77                 LD      (HL),A          ; Load test byte
E068 BE                 CP      (HL)            ; RAM there if same
E069 70                 LD      (HL),B          ; Restore old contents
E06A C236E0             JP      NZ,MSIZE        ; Ask again if no RAM

E06D 2B         SETTOP: DEC     HL              ; Back one byte
E06E 115C11             LD      DE,STLOOK-1     ; See if enough RAM
E071 CD8AE6             CALL    CPDEHL          ; Compare DE with HL
E074 DA36E0             JP      C,MSIZE         ; Ask again if not enough RAM
E077 00                 NOP
E078 00                 NOP
E079 00                 NOP
E07A 00                 NOP
E07B 00                 NOP
E07C 00                 NOP
E07D 00                 NOP
E07E 00                 NOP
E07F 00                 NOP
E080 11CEFF             LD      DE,-50          ; 50 Bytes string space
E083 22AF10             LD      (LSTRAM),HL     ; Save last available RAM
E086 19                 ADD     HL,DE           ; Allocate string space
E087 225A10             LD      (STRSPC),HL     ; Save string space
E08A CDBAE4             CALL    CLRPTR          ; Clear program area
E08D 2A5A10             LD      HL,(STRSPC)     ; Get end of memory
E090 11EFFF             LD      DE,-17          ; Offset for free bytes
E093 19                 ADD     HL,DE           ; Adjust HL
E094 11F910             LD      DE,PROGST       ; Start of program text
E097 7D                 LD      A,L             ; Get LSB
E098 93                 SUB     E               ; Adjust it
E099 6F                 LD      L,A             ; Re-save
E09A 7C                 LD      A,H             ; Get MSB
E09B 9A                 SBC     A,D             ; Adjust it
E09C 67                 LD      H,A             ; Re-save
E09D E5                 PUSH    HL              ; Save bytes free
E09E 21C5E0             LD      HL,SIGNON       ; Sign-on message
E0A1 CD10F2             CALL    PRS             ; Output string
E0A4 E1                 POP     HL              ; Get bytes free back
E0A5 CDADF9             CALL    PRNTHL          ; Output amount of free memory
E0A8 21B7E0             LD      HL,BFREE        ; " Bytes free" message
E0AB CD10F2             CALL    PRS             ; Output string

E0AE 316610     WARMST: LD      SP,STACK        ; Temporary stack
E0B1 CDDFE4     BRKRET: CALL    CLREG           ; Clear registers and stack
E0B4 C3F8E3             JP      PRNTOK          ; Go to get command line
```

```
E0B7 20427974   BFREE:  DEFB    " Bytes free",CR,0,0

E0C5 4E415343   SIGNON: DEFB    "NASCOM ROM BASIC Ver 4.7   ",CR
E0E1 436F7079          DEFB    "Copyright (C) 1978 by Microsoft",CR,0,0

E103 4D656D6F   MEMMSG: DEFB    "Memory size",0

                ; FUNCTION ADDRESS TABLE

E10F 22F8       FNCTAB: DEFW    SGN
E111 E6F8               DEFW    INT
E113 38F8               DEFW    ABS
E115 0310               DEFW    USR
E117 D0F0               DEFW    FRE
E119 41F4               DEFW    INP
E11B FEF0               DEFW    POS
E11D ACFA               DEFW    SQR
E11F 8BFB               DEFW    RND
E121 C7F6               DEFW    LOG
E123 FAFA               DEFW    EXP
E125 00FC               DEFW    COS
E127 06FC               DEFW    SIN
E129 67FC               DEFW    TAN
E12B 7CFC               DEFW    ATN
E12D A3F5               DEFW    PEEK
E12F BCFD               DEFW    DEEK
E131 5110               DEFW    POINT
E133 82F3               DEFW    LEN
E135 9AF1               DEFW    STR
E137 1CF4               DEFW    VAL
E139 91F3               DEFW    ASC
E13B A2F3               DEFW    CHR
E13D B2F3               DEFW    LEFT
E13F E2F3               DEFW    RIGHT
E141 ECF3               DEFW    MID
```

```
                ; RESERVED WORD LIST

E143 C54E44     WORDS:  DEFB    80H+"END"
E146 C64F52             DEFB    80H+"FOR"
E149 CE455854           DEFB    80H+"NEXT"
E14D C4415441           DEFB    80H+"DATA"
E151 C94E5055           DEFB    80H+"INPUT"
E156 C4494D             DEFB    80H+"DIM"
E159 D2454144           DEFB    80H+"READ"
E15D CC4554             DEFB    80H+"LET"
E160 C74F544F           DEFB    80H+"GOTO"
E164 D2554E             DEFB    80H+"RUN"
E167 C946               DEFB    80H+"IF"
E169 D2455354           DEFB    80H+"RESTORE"
E170 C74F5355           DEFB    80H+"GOSUB"
E175 D2455455           DEFB    80H+"RETURN"
E17B D2454D             DEFB    80H+"REM"
E17E D3544F50           DEFB    80H+"STOP"
E182 CF5554             DEFB    80H+"OUT"
E185 CF4E               DEFB    80H+"ON"
E187 CE554C4C           DEFB    80H+"NULL"
E18B D7414954           DEFB    80H+"WAIT"
E18F C44546             DEFB    80H+"DEF"
E192 D04F4B45           DEFB    80H+"POKE"
E196 C44F4B45           DEFB    80H+"DOKE"
E19A D3435245           DEFB    80H+"SCREEN"
E1A0 CC494E45           DEFB    80H+"LINES"
E1A5 C34C53             DEFB    80H+"CLS"
E1A8 D7494454           DEFB    80H+"WIDTH"
E1AD CD4F4E49           DEFB    80H+"MONITOR"
E1B4 D34554             DEFB    80H+"SET"
E1B7 D2455345           DEFB    80H+"RESET"
E1BC D052494E           DEFB    80H+"PRINT"
E1C1 C34F4E54           DEFB    80H+"CONT"
E1C5 CC495354           DEFB    80H+"LIST"
E1C9 C34C4541           DEFB    80H+"CLEAR"
E1CE C34C4F41           DEFB    80H+"CLOAD"
E1D3 C3534156           DEFB    80H+"CSAVE"
E1D8 CE4557             DEFB    80H+"NEW"
```

```
E1DB D4414228           DEFB    80H+"TAB("
E1DF D44F               DEFB    80H+"TO"
E1E1 C64E               DEFB    80H+"FN"
E1E3 D3504328           DEFB    80H+"SPC("
E1E7 D448454E           DEFB    80H+"THEN"
E1EB CE4F54             DEFB    80H+"NOT"
E1EE D3544550           DEFB    80H+"STEP"

E1F2 AB                 DEFB    80H+"+"
E1F3 AD                 DEFB    80H+"-"
E1F4 AA                 DEFB    80H+"*"
E1F5 AF                 DEFB    80H+"/"
E1F6 DE                 DEFB    80H+"^"
E1F7 C14E44             DEFB    80H+"AND"
E1FA CF52               DEFB    80H+"OR"
E1FC BE                 DEFB    80H+">"
E1FD BD                 DEFB    80H+"="
E1FE BC                 DEFB    80H+"<"

E1FF D3474E             DEFB    80H+"SGN"
E202 C94E54             DEFB    80H+"INT"
E205 C14253             DEFB    80H+"ABS"
E208 D55352             DEFB    80H+"USR"
E20B C65245             DEFB    80H+"FRE"
E20E C94E50             DEFB    80H+"INP"
E211 D04F53             DEFB    80H+"POS"
E214 D35152             DEFB    80H+"SQR"
E217 D24E44             DEFB    80H+"RND"
E21A CC4F47             DEFB    80H+"LOG"
E21D C55850             DEFB    80H+"EXP"
E220 C34F53             DEFB    80H+"COS"
E223 D3494E             DEFB    80H+"SIN"
E226 D4414E             DEFB    80H+"TAN"
E229 C1544E             DEFB    80H+"ATN"
E22C D045454B           DEFB    80H+"PEEK"
E230 C445454B           DEFB    80H+"DEEK"
E234 D04F494E           DEFB    80H+"POINT"
E239 CC454E             DEFB    80H+"LEN"
E23C D3545224           DEFB    80H+"STR$"
E240 D6414C             DEFB    80H+"VAL"
E243 C15343             DEFB    80H+"ASC"
E246 C3485224           DEFB    80H+"CHR$"
E24A CC454654           DEFB    80H+"LEFT$"
E24F D2494748           DEFB    80H+"RIGHT$"
E255 CD494424           DEFB    80H+"MID$"
E259 80                 DEFB    80H             ; End of list marker
```

```
                ; KEYWORD ADDRESS TABLE
E25A 72E8       WORDTB: DEFW    PEND
E25C 79E7               DEFW    FOR
E25E F6EC               DEFW    NEXT
E260 70EA               DEFW    DATA
E262 FDEB               DEFW    INPUT
E264 28EF               DEFW    DIM
E266 2CEC               DEFW    READ
E268 87EA               DEFW    LET
E26A 2DEA               DEFW    GOTO
E26C 10EA               DEFW    RUN
E26E FFEA               DEFW    IF
E270 46E8               DEFW    RESTOR
E272 1CEA               DEFW    GOSUB
E274 4BEA               DEFW    RETURN
E276 72EA               DEFW    REM
E278 70E8               DEFW    STOP
E27A 4DF4               DEFW    POUT
E27C E1EA               DEFW    ON
E27E B1E8               DEFW    NULL
E280 53F4               DEFW    WAIT
E282 06F1               DEFW    DEF
E284 AAF5               DEFW    POKE
E286 C7FD               DEFW    DOKE
E288 E6FD               DEFW    SCREEN
E28A ADFD               DEFW    LINES
E28C 8BFD               DEFW    CLS
E28E A5FD               DEFW    WIDTH
E290 A2FE               DEFW    MONITR
E292 5410               DEFW    PSET
E294 5710               DEFW    RESET
E296 23EB               DEFW    PRINT
E298 9EE8               DEFW    CONT
E29A DDE6               DEFW    LIST
E29C CAE9               DEFW    CLEAR
E29E F9F4               DEFW    CLOAD
E2A0 C3F4               DEFW    CSAVE
E2A2 B9E4               DEFW    NEW
```

```
                ; RESERVED WORD TOKEN VALUES

0080            ZEND    EQU     080H            ; END
0081            ZFOR    EQU     081H            ; FOR
0083            ZDATA   EQU     083H            ; DATA
0088            ZGOTO   EQU     088H            ; GOTO
008C            ZGOSUB  EQU     08CH            ; GOSUB
008E            ZREM    EQU     08EH            ; REM
009E            ZPRINT  EQU     09EH            ; PRINT
00A4            ZNEW    EQU     0A4H            ; NEW

00A5            ZTAB    EQU     0A5H            ; TAB
00A6            ZTO     EQU     0A6H            ; TO
00A7            ZFN     EQU     0A7H            ; FN
00A8            ZSPC    EQU     0A8H            ; SPC
00A9            ZTHEN   EQU     0A9H            ; THEN
00AA            ZNOT    EQU     0AAH            ; NOT
00AB            ZSTEP   EQU     0ABH            ; STEP

00AC            ZPLUS   EQU     0ACH            ; +
00AD            ZMINUS  EQU     0ADH            ; -
00AE            ZTIMES  EQU     0AEH            ; *
00AF            ZDIV    EQU     0AFH            ; /
00B2            ZOR     EQU     0B2H            ; OR
00B3            ZGTR    EQU     0B3H            ; >
00B4            ZEQUAL  EQU     0B4H            ; =
00B5            ZLTH    EQU     0B5H            ; <
00B6            ZSGN    EQU     0B6H            ; SGN
00C7            ZPOINT  EQU     0C7H            ; POINT
00CD            ZLEFT   EQU     0CDH            ; LEFT$
```

```
                ; ARITHMETIC PRECEDENCE TABLE

E2A4 79         PRITAB: DEFB    79H             ; Precedence value
E2A5 94F9               DEFW    PADD            ; FPREG = <last> + FPREG

E2A7 79                 DEFB    79H             ; Precedence value
E2A8 C8F5               DEFW    PSUB            ; FPREG = <last> - FPREG

E2AA 7C                 DEFB    7CH             ; Precedence value
E2AB 06F7               DEFW    MULT            ; FPREG = <last> * FPREG

E2AD 7C                 DEFB    7CH             ; Precedence value
E2AE 67F7               DEFW    DIV             ; FPREG = <last> / FPREG

E2B0 7F                 DEFB    7FH             ; Precedence value
E2B1 B5FA               DEFW    POWER           ; FPREG = <last> ^ FPREG

E2B3 50                 DEFB    50H             ; Precedence value
E2B4 81EE               DEFW    PAND            ; FPREG = <last> AND FPREG

E2B6 46                 DEFB    46H             ; Precedence value
E2B7 80EE               DEFW    POR             ; FPREG = <last> OR FPREG

                ; BASIC ERROR CODE LIST

E2B9 4E46       ERRORS: DEFB    "NF"            ; NEXT without FOR
E2BB 534E               DEFB    "SN"            ; Syntax error
E2BD 5247               DEFB    "RG"            ; RETURN without GOSUB
E2BF 4F44               DEFB    "OD"            ; Out of DATA
E2C1 4643               DEFB    "FC"            ; Illegal function call
E2C3 4F56               DEFB    "OV"            ; Overflow error
E2C5 4F4D               DEFB    "OM"            ; Out of memory
E2C7 554C               DEFB    "UL"            ; Undefined line
E2C9 4253               DEFB    "BS"            ; Bad subscript
E2CB 4444               DEFB    "DD"            ; Re-DIMensioned array
E2CD 2F30               DEFB    "/0"            ; Division by zero
E2CF 4944               DEFB    "ID"            ; Illegal direct
E2D1 544D               DEFB    "TM"            ; Type mis-match
E2D3 4F53               DEFB    "OS"            ; Out of string space
E2D5 4C53               DEFB    "LS"            ; String too long
E2D7 5354               DEFB    "ST"            ; String formula too complex
E2D9 434E               DEFB    "CN"            ; Can't CONTinue
E2DB 5546               DEFB    "UF"            ; Undefined FN function
E2DD 4D4F               DEFB    "MO"            ; Missing operand
```

```
                ; INITIALISATION TABLE

E2DF C3AEE0     INITAB: JP      WARMST          ; Warm start jump
E2E2 C3A0E9             JP      FCERR           ; "USR (X)" jump (Set to Error)

E2E5 D300               OUT     (0),A           ; "OUT p,n" skeleton
E2E7 C9                 RET

E2E8 D600               SUB     0               ; Division support routine
E2EA 6F                 LD      L,A
E2EB 7C                 LD      A,H
E2EC DE00               SBC     A,0
E2EE 67                 LD      H,A
E2EF 78                 LD      A,B
E2F0 DE00               SBC     A,0
E2F2 47                 LD      B,A
E2F3 3E00               LD      A,0
E2F5 C9                 RET

E2F6 000000             DEFB    0,0,0                   ; Random number seed
                                                        ; Table used by RND
E2F9 354ACA99           DEFB    035H,04AH,0CAH,099H     ;-2.65145E+07
E2FD 391C7698           DEFB    039H,01CH,076H,098H     ; 1.61291E+07
E301 2295B398           DEFB    022H,095H,0B3H,098H     ;-1.17691E+07
E305 0ADD4798           DEFB    00AH,0DDH,047H,098H     ; 1.30983E+07
E309 53D19999           DEFB    053H,0D1H,099H,099H     ;-2.01612E+07
E30D 0A1A9F98           DEFB    00AH,01AH,09FH,098H     ;-1.04269E+07
E311 65BCCD98           DEFB    065H,0BCH,0CDH,098H     ;-1.34831E+07
E315 D6773E98           DEFB    0D6H,077H,03EH,098H     ; 1.24825E+07
E319 52C74F80           DEFB    052H,0C7H,04FH,080H     ; Last random number

E31D DB00               IN      A,(0)           ; INP (x) skeleton
E31F C9                 RET

E320 01                 DEFB    1               ; POS (x) number (1)
E321 2F                 DEFB    47              ; Terminal width (47)
E322 1C                 DEFB    28              ; Width for commas (3 columns)
E323 00                 DEFB    0               ; No nulls after input bytes
E324 00                 DEFB    0               ; Output enabled (^O off)

E325 0500               DEFW    5               ; Initial lines counter
E327 0500               DEFW    5               ; Initial lines number
E329 0000               DEFW    0               ; Array load/save check sum

E32B 00                 DEFB    0               ; Break not by NMI
E32C 00                 DEFB    0               ; Break flag

E32D C307E6             JP      TTYLIN          ; Input reflection (set to TTY)
E330 C379FF             JP      POINTB          ; POINT reflection unused
E333 C340FF             JP      SETB            ; SET   reflection
E336 C355FF             JP      RESETB          ; RESET reflection

E339 5D11               DEFW    STLOOK          ; Temp string space
E33B FEFF               DEFW    -2              ; Current line number (cold)
E33D FA10               DEFW    PROGST+1        ; Start of program text
E33F            INITBE:                         ; END OF INITIALISATION TABLE
```

```
E33F 20457272   ERRMSG: DEFB    " Error",0
E346 20696E20   INMSG:  DEFB    " in ",0
E34A            ZERBYT  EQU     $-1             ; A zero byte
E34B 4F6B0D00   OKMSG:  DEFB    "Ok",CR,0,0
E350 42726561   BRKMSG: DEFB    "Break",0

E356 210400     BAKSTK: LD      HL,4            ; Look for "FOR" block with
E359 39                 ADD     HL,SP           ; same index as specified
E35A 7E         LOKFOR: LD      A,(HL)          ; Get block ID
E35B 23                 INC     HL              ; Point to index address
E35C FE81               CP      ZFOR            ; Is it a "FOR" token
E35E C0                 RET     NZ              ; No - exit
E35F 4E                 LD      C,(HL)          ; BC = Address of "FOR" index
E360 23                 INC     HL
E361 46                 LD      B,(HL)
E362 23                 INC     HL              ; Point to sign of STEP
E363 E5                 PUSH    HL              ; Save pointer to sign
E364 69                 LD      L,C             ; HL = address of "FOR" index
E365 60                 LD      H,B
E366 7A                 LD      A,D             ; See if an index was specified
E367 B3                 OR      E               ; DE = 0 if no index specified
E368 EB                 EX      DE,HL           ; Specified index into HL
E369 CA70E3             JP      Z,INDFND        ; Skip if no index given
E36C EB                 EX      DE,HL           ; Index back into DE
E36D CD8AE6             CALL    CPDEHL          ; Compare index with one given
E370 010D00     INDFND: LD      BC,16-3         ; Offset to next block
E373 E1                 POP     HL              ; Restore pointer to sign
E374 C8                 RET     Z               ; Return if block found
E375 09                 ADD     HL,BC           ; Point to next block
E376 C35AE3             JP      LOKFOR          ; Keep on looking

E379 CD93E3     MOVUP:  CALL    ENFMEM          ; See if enough memory
E37C C5         MOVSTR: PUSH    BC              ; Save end of source
E37D E3                 EX      (SP),HL         ; Swap source and dest" end
E37E C1                 POP     BC              ; Get end of destination
E37F CD8AE6     MOVLP:  CALL    CPDEHL          ; See if list moved
E382 7E                 LD      A,(HL)          ; Get byte
E383 02                 LD      (BC),A          ; Move it
E384 C8                 RET     Z               ; Exit if all done
E385 0B                 DEC     BC              ; Next byte to move to
E386 2B                 DEC     HL              ; Next byte to move
E387 C37FE3             JP      MOVLP           ; Loop until all bytes moved
```

```
E38A E5          CHKSTK: PUSH    HL              ; Save code string address
E38B 2ADA10              LD      HL,(ARREND)     ; Lowest free memory
E38E 0600                LD      B,0             ; BC = Number of levels to test
E390 09                  ADD     HL,BC           ; 2 Bytes for each level
E391 09                  ADD     HL,BC
E392 3E                  DEFB    (LD A,n)        ; Skip "PUSH HL"
E393 E5          ENFMEM: PUSH    HL              ; Save code string address
E394 3ED0                LD      A,LOW -48       ; 48 Bytes minimum RAM
E396 95                  SUB     L
E397 6F                  LD      L,A
E398 3EFF                LD      A,HIGH -48      ; 48 Bytes minimum RAM
E39A 9C                  SBC     A,H
E39B DAA2E3              JP      C,OMERR         ; Not enough - ?OM Error
E39E 67                  LD      H,A
E39F 39                  ADD     HL,SP           ; Test if stack is overflowed
E3A0 E1                  POP     HL              ; Restore code string address
E3A1 D8                  RET     C               ; Return if enough mmory
E3A2 1E0C        OMERR:  LD      E,OM            ; ?OM Error
E3A4 C3C1E3              JP      ERROR

E3A7 2AC910      DATSNR: LD      HL,(DATLIN)     ; Get line of current DATA item
E3AA 225C10              LD      (LINEAT),HL     ; Save as current line
E3AD 1E02        SNERR:  LD      E,SN            ; ?SN Error
E3AF 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,DZ"
E3B0 1E14        DZERR:  LD      E,DZ            ; ?/0 Error
E3B2 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,NF"
E3B3 1E00        NFERR:  LD      E,NF            ; ?NF Error
E3B5 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,DD"
E3B6 1E12        DDERR:  LD      E,DD            ; ?DD Error
E3B8 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,UF"
E3B9 1E22        UFERR:  LD      E,UF            ; ?UF Error
E3BB 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,OV"
E3BC 1E0A        OVERR:  LD      E,OV            ; ?OV Error
E3BE 01                  DEFB    (LD  BC,nn)     ; Skip "LD E,TM"
E3BF 1E18        TMERR:  LD      E,TM            ; ?TM Error
```

```
E3C1 CDDFE4     ERROR:  CALL    CLREG           ; Clear registers and stack
E3C4 324510             LD      (CTLOFG),A      ; Enable output (A is 0)
E3C7 CD74EB             CALL    STTLIN          ; Start new line
E3CA 21B9E2             LD      HL,ERRORS       ; Point to error codes
E3CD 57                 LD      D,A             ; D = 0  (A is 0)
E3CE 3E3F               LD      A,"?"
E3D0 CD9BE6             CALL    OUTC            ; Output "?"
E3D3 19                 ADD     HL,DE           ; Offset to correct error code
E3D4 7E                 LD      A,(HL)          ; First character
E3D5 CD9BE6             CALL    OUTC            ; Output it
E3D8 CD36E8             CALL    GETCHR          ; Get next character
E3DB CD9BE6             CALL    OUTC            ; Output it
E3DE 213FE3             LD      HL,ERRMSG       ; "Error" message
E3E1 CD10F2     ERRIN:  CALL    PRS             ; Output message
E3E4 2A5C10             LD      HL,(LINEAT)     ; Get line of error
E3E7 11FEFF             LD      DE,-2           ; Cold start error if -2
E3EA CD8AE6             CALL    CPDEHL          ; See if cold start error
E3ED CA12E0             JP      Z,CSTART        ; Cold start error - Restart
E3F0 7C                 LD      A,H             ; Was it a direct error?
E3F1 A5                 AND     L               ; Line = -1 if direct error
E3F2 3C                 INC     A
E3F3 C4A5F9             CALL    NZ,LINEIN       ; No - output line of error
E3F6 3E                 DEFB    (LD A,n)        ; Skip "POP BC"
E3F7 C1         POPNOK: POP     BC              ; Drop address in input buffer
```

```
E3F8 AF          PRNTOK: XOR     A               ; Output "Ok" and get command
E3F9 324510              LD      (CTLOFG),A      ; Enable output
E3FC CD74EB              CALL    STTLIN          ; Start new line
E3FF 214BE3              LD      HL,OKMSG        ; "Ok" message
E402 CD10F2              CALL    PRS             ; Output "Ok"
E405 21FFFF      GETCMD: LD      HL,-1           ; Flag direct mode
E408 225C10              LD      (LINEAT),HL     ; Save as current line
E40B CDF2E5              CALL    GETLIN          ; Get an input line
E40E DA05E4              JP      C,GETCMD        ; Get line again if break
E411 CD36E8              CALL    GETCHR          ; Get first character
E414 3C                  INC     A               ; Test if end of line
E415 3D                  DEC     A               ; Without affecting Carry
E416 CA05E4              JP      Z,GETCMD        ; Nothing entered - Get another
E419 F5                  PUSH    AF              ; Save Carry status
E41A CDA5E9              CALL    ATOH            ; Get line number into DE
E41D D5                  PUSH    DE              ; Save line number
E41E CD09E5              CALL    CRUNCH          ; Tokenise rest of line
E421 47                  LD      B,A             ; Length of tokenised line
E422 D1                  POP     DE              ; Restore line number
E423 F1                  POP     AF              ; Restore Carry
E424 D216E8              JP      NC,EXCUTE       ; No line number - Direct mode
E427 D5                  PUSH    DE              ; Save line number
E428 C5                  PUSH    BC              ; Save length of tokenised line
E429 AF                  XOR     A
E42A 32CC10              LD      (LSTBIN),A      ; Clear last byte input
E42D CD36E8              CALL    GETCHR          ; Get next character
E430 B7                  OR      A               ; Set flags
E431 F5                  PUSH    AF              ; And save them
E432 CD99E4              CALL    SRCHLN          ; Search for line number in DE
E435 DA3EE4              JP      C,LINFND        ; Jump if line found
E438 F1                  POP     AF              ; Get status
E439 F5                  PUSH    AF              ; And re-save
E43A CA46EA              JP      Z,ULERR         ; Nothing after number - Error
E43D B7                  OR      A               ; Clear Carry
E43E C5          LINFND: PUSH    BC              ; Save address of line in prog
E43F D255E4              JP      NC,INEWLN       ; Line not found - Insert new
E442 EB                  EX      DE,HL           ; Next line address in DE
E443 2AD610              LD      HL,(PROGND)     ; End of program
E446 1A          SFTPRG: LD      A,(DE)          ; Shift rest of program down
E447 02                  LD      (BC),A
E448 03                  INC     BC              ; Next destination
E449 13                  INC     DE              ; Next source
E44A CD8AE6              CALL    CPDEHL          ; All done?
E44D C246E4              JP      NZ,SFTPRG       ; More to do
E450 60                  LD      H,B             ; HL = New end of program
E451 69                  LD      L,C
E452 22D610              LD      (PROGND),HL     ; Update end of program
```

```
E455 D1          INEWLN: POP     DE              ; Get address of line
E456 F1                  POP     AF              ; Get status
E457 CA7CE4              JP      Z,SETPTR        ; No text - Set up pointers
E45A 2AD610              LD      HL,(PROGND)     ; Get end of program
E45D E3                  EX      (SP),HL         ; Get length of input line
E45E C1                  POP     BC              ; End of program to BC
E45F 09                  ADD     HL,BC           ; Find new end
E460 E5                  PUSH    HL              ; Save new end
E461 CD79E3              CALL    MOVUP           ; Make space for line
E464 E1                  POP     HL              ; Restore new end
E465 22D610              LD      (PROGND),HL     ; Update end of program pointer
E468 EB                  EX      DE,HL           ; Get line to move up in HL
E469 74                  LD      (HL),H          ; Save MSB
E46A D1                  POP     DE              ; Get new line number
E46B 23                  INC     HL              ; Skip pointer
E46C 23                  INC     HL
E46D 73                  LD      (HL),E          ; Save LSB of line number
E46E 23                  INC     HL
E46F 72                  LD      (HL),D          ; Save MSB of line number
E470 23                  INC     HL              ; To first byte in line
E471 116110              LD      DE,BUFFER       ; Copy buffer to program
E474 1A          MOVBUF: LD      A,(DE)          ; Get source
E475 77                  LD      (HL),A          ; Save destinations
E476 23                  INC     HL              ; Next source
E477 13                  INC     DE              ; Next destination
E478 B7                  OR      A               ; Done?
E479 C274E4              JP      NZ,MOVBUF       ; No - Repeat
E47C CDC5E4      SETPTR: CALL    RUNFST          ; Set line pointers
E47F 23                  INC     HL              ; To LSB of pointer
E480 EB                  EX      DE,HL           ; Address to DE
E481 62          PTRLP:  LD      H,D             ; Address to HL
E482 6B                  LD      L,E
E483 7E                  LD      A,(HL)          ; Get LSB of pointer
E484 23                  INC     HL              ; To MSB of pointer
E485 B6                  OR      (HL)            ; Compare with MSB pointer
E486 CA05E4              JP      Z,GETCMD        ; Get command line if end
E489 23                  INC     HL              ; To LSB of line number
E48A 23                  INC     HL              ; Skip line number
E48B 23                  INC     HL              ; Point to first byte in line
E48C AF                  XOR     A               ; Looking for 00 byte
E48D BE          FNDEND: CP      (HL)            ; Found end of line?
E48E 23                  INC     HL              ; Move to next byte
E48F C28DE4              JP      NZ,FNDEND       ; No - Keep looking
E492 EB                  EX      DE,HL           ; Next line address to HL
E493 73                  LD      (HL),E          ; Save LSB of pointer
E494 23                  INC     HL
E495 72                  LD      (HL),D          ; Save MSB of pointer
E496 C381E4              JP      PTRLP           ; Do next line
```

```
E499 2A5E10     SRCHLN: LD      HL,(BASTXT)     ; Start of program text
E49C 44         SRCHLP: LD      B,H             ; BC = Address to look at
E49D 4D                 LD      C,L
E49E 7E                 LD      A,(HL)          ; Get address of next line
E49F 23                 INC     HL
E4A0 B6                 OR      (HL)            ; End of program found?
E4A1 2B                 DEC     HL
E4A2 C8                 RET     Z               ; Yes - Line not found
E4A3 23                 INC     HL
E4A4 23                 INC     HL
E4A5 7E                 LD      A,(HL)          ; Get LSB of line number
E4A6 23                 INC     HL
E4A7 66                 LD      H,(HL)          ; Get MSB of line number
E4A8 6F                 LD      L,A
E4A9 CD8AE6             CALL    CPDEHL          ; Compare with line in DE
E4AC 60                 LD      H,B             ; HL = Start of this line
E4AD 69                 LD      L,C
E4AE 7E                 LD      A,(HL)          ; Get LSB of next line address
E4AF 23                 INC     HL
E4B0 66                 LD      H,(HL)          ; Get MSB of next line address
E4B1 6F                 LD      L,A             ; Next line to HL
E4B2 3F                 CCF
E4B3 C8                 RET     Z               ; Lines found - Exit
E4B4 3F                 CCF
E4B5 D0                 RET     NC              ; Line not found,at line after
E4B6 C39CE4             JP      SRCHLP          ; Keep looking
```

```
E4B9 C0          NEW:     RET     NZ              ; Return if any more on line
E4BA 2A5E10      CLRPTR:  LD      HL,(BASTXT)     ; Point to start of program
E4BD AF                   XOR     A               ; Set program area to empty
E4BE 77                   LD      (HL),A          ; Save LSB = 00
E4BF 23                   INC     HL
E4C0 77                   LD      (HL),A          ; Save MSB = 00
E4C1 23                   INC     HL
E4C2 22D610               LD      (PROGND),HL     ; Set program end

E4C5 2A5E10      RUNFST:  LD      HL,(BASTXT)     ; Clear all variables
E4C8 2B                   DEC     HL

E4C9 22CE10      INTVAR:  LD      (BRKLIN),HL     ; Initialise RUN variables
E4CC 2AAF10               LD      HL,(LSTRAM)     ; Get end of RAM
E4CF 22C310               LD      (STRBOT),HL     ; Clear string space
E4D2 AF                   XOR     A
E4D3 CD46E8               CALL    RESTOR          ; Reset DATA pointers
E4D6 2AD610               LD      HL,(PROGND)     ; Get end of program
E4D9 22D810               LD      (VAREND),HL     ; Clear variables
E4DC 22DA10               LD      (ARREND),HL     ; Clear arrays

E4DF C1          CLREG:   POP     BC              ; Save return address
E4E0 2A5A10               LD      HL,(STRSPC)     ; Get end of working RAM
E4E3 F9                   LD      SP,HL           ; Set stack
E4E4 21B310               LD      HL,TMSTPL       ; Temporary string pool
E4E7 22B110               LD      (TMSTPT),HL     ; Reset temporary string ptr
E4EA AF                   XOR     A               ; A = 00
E4EB 6F                   LD      L,A             ; HL = 0000
E4EC 67                   LD      H,A
E4ED 22D410               LD      (CONTAD),HL     ; No CONTinue
E4F0 32CB10               LD      (FORFLG),A      ; Clear FOR flag
E4F3 22DE10               LD      (FNRGNM),HL     ; Clear FN argument
E4F6 E5                   PUSH    HL              ; HL = 0000
E4F7 C5                   PUSH    BC              ; Put back return
E4F8 2ACE10      DOAGN:   LD      HL,(BRKLIN)     ; Get address of code to RUN
E4FB C9                   RET                     ; Return to execution driver

E4FC 3E3F        PROMPT:  LD      A,"?"           ; "?"
E4FE CD9BE6               CALL    OUTC            ; Output character
E501 3E20                 LD      A," "           ; Space
E503 CD9BE6               CALL    OUTC            ; Output character
E506 C34E10               JP      RINPUT          ; Get input line
```

```
E509 AF          CRUNCH: XOR     A               ; Tokenise line % HL to BUFFER
E50A 32AE10              LD      (DATFLG),A      ; Reset literal flag
E50D 0E05                LD      C,2+3           ; 2 byte number and 3 nulls
E50F 116110              LD      DE,BUFFER       ; Start of input buffer
E512 7E          CRNCLP: LD      A,(HL)          ; Get byte
E513 FE20                CP      " "             ; Is it a space?
E515 CA91E5              JP      Z,MOVDIR        ; Yes - Copy direct
E518 47                  LD      B,A             ; Save character
E519 FE22                CP      '"'             ; Is it a quote?
E51B CAB1E5              JP      Z,CPYLIT        ; Yes - Copy literal string
E51E B7                  OR      A               ; Is it end of buffer?
E51F CAB8E5              JP      Z,ENDBUF        ; Yes - End buffer
E522 3AAE10              LD      A,(DATFLG)      ; Get data type
E525 B7                  OR      A               ; Literal?
E526 7E                  LD      A,(HL)          ; Get byte to copy
E527 C291E5              JP      NZ,MOVDIR       ; Literal - Copy direct
E52A FE3F                CP      "?"             ; Is it "?" short for PRINT
E52C 3E9E                LD      A,ZPRINT        ; "PRINT" token
E52E CA91E5              JP      Z,MOVDIR        ; Yes - replace it
E531 7E                  LD      A,(HL)          ; Get byte again
E532 FE30                CP      "0"             ; Is it less than "0"
E534 DA3CE5              JP      C,FNDWRD        ; Yes - Look for reserved words
E537 FE3C                CP      ";"+1           ; Is it "0123456789:;" ?
E539 DA91E5              JP      C,MOVDIR        ; Yes - copy it direct
E53C D5          FNDWRD: PUSH    DE              ; Look for reserved words
E53D 1142E1              LD      DE,WORDS-1      ; Point to table
E540 C5                  PUSH    BC              ; Save count
E541 018DE5              LD      BC,RETNAD       ; Where to return to
E544 C5                  PUSH    BC              ; Save return address
E545 067F                LD      B,ZEND-1        ; First token value -1
E547 7E                  LD      A,(HL)          ; Get byte
E548 FE61                CP      "a"             ; Less than "a" ?
E54A DA55E5              JP      C,SEARCH        ; Yes - search for words
E54D FE7B                CP      "z"+1           ; Greater than "z" ?
E54F D255E5              JP      NC,SEARCH       ; Yes - search for words
E552 E65F                AND     01011111B       ; Force upper case
E554 77                  LD      (HL),A          ; Replace byte
E555 4E          SEARCH: LD      C,(HL)          ; Search for a word
E556 EB                  EX      DE,HL
E557 23          GETNXT: INC     HL              ; Get next reserved word
E558 B6                  OR      (HL)            ; Start of word?
E559 F257E5              JP      P,GETNXT        ; No - move on
E55C 04                  INC     B               ; Increment token value
E55D 7E                  LD      A,(HL)          ; Get byte from table
E55E E67F                AND     01111111B       ; Strip bit 7
E560 C8                  RET     Z               ; Return if end of list
E561 B9                  CP      C               ; Same character as in buffer?
E562 C257E5              JP      NZ,GETNXT       ; No - get next word
E565 EB                  EX      DE,HL
E566 E5                  PUSH    HL              ; Save start of word
```

```
E567 13          NXTBYT: INC     DE          ; Look through rest of word
E568 1A                  LD      A,(DE)      ; Get byte from table
E569 B7                  OR      A           ; End of word ?
E56A FA89E5              JP      M,MATCH     ; Yes - Match found
E56D 4F                  LD      C,A         ; Save it
E56E 78                  LD      A,B         ; Get token value
E56F FE88                CP      ZGOTO       ; Is it "GOTO" token ?
E571 C278E5              JP      NZ,NOSPC    ; No - Don't allow spaces
E574 CD36E8              CALL    GETCHR      ; Get next character
E577 2B                  DEC     HL          ; Cancel increment from GETCHR
E578 23          NOSPC:  INC     HL          ; Next byte
E579 7E                  LD      A,(HL)      ; Get byte
E57A FE61                CP      "a"         ; Less than "a" ?
E57C DA81E5              JP      C,NOCHNG    ; Yes - don't change
E57F E65F                AND     01011111B   ; Make upper case
E581 B9          NOCHNG: CP      C           ; Same as in buffer ?
E582 CA67E5              JP      Z,NXTBYT    ; Yes - keep testing
E585 E1                  POP     HL          ; Get back start of word
E586 C355E5              JP      SEARCH      ; Look at next word

E589 48          MATCH:  LD      C,B         ; Word found - Save token value
E58A F1                  POP     AF          ; Throw away return
E58B EB                  EX      DE,HL
E58C C9                  RET                 ; Return to "RETNAD"
E58D EB          RETNAD: EX      DE,HL       ; Get address in string
E58E 79                  LD      A,C         ; Get token value
E58F C1                  POP     BC          ; Restore buffer length
E590 D1                  POP     DE          ; Get destination address
E591 23          MOVDIR: INC     HL          ; Next source in buffer
E592 12                  LD      (DE),A      ; Put byte in buffer
E593 13                  INC     DE          ; Move up buffer
E594 0C                  INC     C           ; Increment length of buffer
E595 D63A                SUB     ":"         ; End of statement?
E597 CA9FE5              JP      Z,SETLIT    ; Jump if multi-statement line
E59A FE49                CP      ZDATA-":"   ; Is it DATA statement ?
E59C C2A2E5              JP      NZ,TSTREM   ; No - see if REM
E59F 32AE10      SETLIT: LD      (DATFLG),A  ; Set literal flag
E5A2 D654        TSTREM: SUB     ZREM-":"    ; Is it REM?
E5A4 C212E5              JP      NZ,CRNCLP   ; No - Leave flag
E5A7 47                  LD      B,A         ; Copy rest of buffer
E5A8 7E          NXTCHR: LD      A,(HL)      ; Get byte
E5A9 B7                  OR      A           ; End of line ?
E5AA CAB8E5              JP      Z,ENDBUF    ; Yes - Terminate buffer
E5AD B8                  CP      B           ; End of statement ?
E5AE CA91E5              JP      Z,MOVDIR    ; Yes - Get next one
E5B1 23          CPYLIT: INC     HL          ; Move up source string
E5B2 12                  LD      (DE),A      ; Save in destination
E5B3 0C                  INC     C           ; Increment length
E5B4 13                  INC     DE          ; Move up destination
E5B5 C3A8E5              JP      NXTCHR      ; Repeat
```

```
E5B8 216010     ENDBUF: LD      HL,BUFFER-1     ; Point to start of buffer
E5BB 12                 LD      (DE),A          ; Mark end of buffer (A = 00)
E5BC 13                 INC     DE
E5BD 12                 LD      (DE),A          ; A = 00
E5BE 13                 INC     DE
E5BF 12                 LD      (DE),A          ; A = 00
E5C0 C9                 RET

E5C1 3A4410     DODEL:  LD      A,(NULFLG)      ; Get null flag status
E5C4 B7                 OR      A               ; Is it zero?
E5C5 3E00               LD      A,0             ; Zero A - Leave flags
E5C7 324410             LD      (NULFLG),A      ; Zero null flag
E5CA C2D5E5             JP      NZ,ECHDEL       ; Set - Echo it
E5CD 05                 DEC     B               ; Decrement length
E5CE CAF2E5             JP      Z,GETLIN        ; Get line again if empty
E5D1 CD9BE6             CALL    OUTC            ; Output null character
E5D4 3E                 DEFB    (LD A,n)        ; Skip "DEC B"
E5D5 05         ECHDEL: DEC     B               ; Count bytes in buffer
E5D6 2B                 DEC     HL              ; Back space buffer
E5D7 CAE9E5             JP      Z,OTKLN         ; No buffer - Try again
E5DA 7E                 LD      A,(HL)          ; Get deleted byte
E5DB CD9BE6             CALL    OUTC            ; Echo it
E5DE C310E6             JP      MORINP          ; Get more input

E5E1 05         DELCHR: DEC     B               ; Count bytes in buffer
E5E2 2B                 DEC     HL              ; Back space buffer
E5E3 CD9BE6             CALL    OUTC            ; Output character in A
E5E6 C210E6             JP      NZ,MORINP       ; Not end - Get more
E5E9 CD9BE6     OTKLN:  CALL    OUTC            ; Output character in A
E5EC CD81EB     KILIN:  CALL    PRNTCR          ; Output CRLF
E5EF C307E6             JP      TTYLIN          ; Get line again

E5F2 CD6DFE     GETLIN: CALL    MONTST          ; Is it NAS-SYS?
E5F5 CA07E6             JP      Z,TTYLIN        ; No - Character input
E5F8 2A750C             LD      HL,(CIN)        ; Point to NAS-SYS input table
E5FB 7E                 LD      A,(HL)          ; Get input mode
E5FC FE74               CP      74H             ; Is it "X" mode?
E5FE CA07E6             JP      Z,TTYLIN        ; Yes - Teletype line input
E601 CDE8FE             CALL    INLINE          ; Get a line from NAS-SYS
E604 C386EB             JP      DONULL          ; POS(X)=0 and do nulls
```

```
E607 216110     TTYLIN: LD      HL,BUFFER       ; Get a line by character
E60A 0601               LD      B,1             ; Set buffer as empty
E60C AF                 XOR     A
E60D 324410             LD      (NULFLG),A      ; Clear null flag
E610 CDCCE6     MORINP: CALL    CLOTST          ; Get character and test ^O
E613 4F                 LD      C,A             ; Save character in C
E614 FE7F               CP      DEL             ; Delete character?
E616 CAC1E5             JP      Z,DODEL         ; Yes - Process it
E619 3A4410             LD      A,(NULFLG)      ; Get null flag
E61C B7                 OR      A               ; Test null flag status
E61D CA29E6             JP      Z,PROCES        ; Reset - Process character
E620 3E00               LD      A,0             ; Set a null
E622 CD9BE6             CALL    OUTC            ; Output null
E625 AF                 XOR     A               ; Clear A
E626 324410             LD      (NULFLG),A      ; Reset null flag
E629 79         PROCES: LD      A,C             ; Get character
E62A FE07               CP      CTRLG           ; Bell?
E62C CA6DE6             JP      Z,PUTCTL        ; Yes - Save it
E62F FE03               CP      CTRLC           ; Is it control "C"?
E631 CC81EB             CALL    Z,PRNTCR        ; Yes - Output CRLF
E634 37                 SCF                     ; Flag break
E635 C8                 RET     Z               ; Return if control "C"
E636 FE0D               CP      CR              ; Is it enter?
E638 CA7CEB             JP      Z,ENDINP        ; Yes - Terminate input
E63B FE15               CP      CTRLU           ; Is it control "U"?
E63D CAECE5             JP      Z,KILIN         ; Yes - Get another line
E640 FE40               CP      "@"             ; Is it "kill line"?
E642 CAE9E5             JP      Z,OTKLN         ; Yes - Kill line
E645 FE5F               CP      "_"             ; Is it delete?
E647 CAE1E5             JP      Z,DELCHR        ; Yes - delete character
E64A FE08               CP      BKSP            ; Is it back space?
E64C CAE1E5             JP      Z,DELCHR        ; Yes - Delete character
E64F FE12               CP      CTRLR           ; Is it control "R"?
E651 C268E6             JP      NZ,PUTBUF       ; No - put in buffer
E654 C5                 PUSH    BC              ; Save buffer length
E655 D5                 PUSH    DE              ; Save DE
E656 E5                 PUSH    HL              ; Save buffer address
E657 3600               LD      (HL),0          ; Mark end of buffer
E659 CDF4FF             CALL    OUTNCR          ; Output and do CRLF
E65C 216110             LD      HL,BUFFER       ; Point to buffer start
E65F CD10F2             CALL    PRS             ; Output buffer
E662 E1                 POP     HL              ; Restore buffer address
E663 D1                 POP     DE              ; Restore DE
E664 C1                 POP     BC              ; Restore buffer length
E665 C310E6             JP      MORINP          ; Get another character
```

# NASCOM

# ROM

# BASIC

# DIS–ASSEMBLED

# PART 3

# BY CARL LLOYD–PARKER

```
E668 FE20       PUTBUF: CP      " "             ; Is it a control code?
E66A DA10E6             JP      C,MORINP        ; Yes - Ignore
E66D 78         PUTCTL: LD      A,B             ; Get number of bytes in buffer
E66E FE49               CP      72+1            ; Test for line overflow
E670 3E07               LD      A,CTRLG         ; Set a bell
E672 D282E6             JP      NC,OUTNBS       ; Ring bell if buffer full
E675 79                 LD      A,C             ; Get character
E676 71                 LD      (HL),C          ; Save in buffer
E677 32CC10             LD      (LSTBIN),A      ; Save last input byte
E67A 23                 INC     HL              ; Move up buffer
E67B 04                 INC     B               ; Increment length
E67C CD9BE6     OUTIT:  CALL    OUTC            ; Output the character entered
E67F C310E6             JP      MORINP          ; Get another character

E682 CD9BE6     OUTNBS: CALL    OUTC            ; Output bell and back over it
E685 3E08               LD      A,BKSP          ; Set back space
E687 C37CE6             JP      OUTIT           ; Output it and get more

E68A 7C         CPDEHL: LD      A,H             ; Get H
E68B 92                 SUB     D               ; Compare with D
E68C C0                 RET     NZ              ; Different - Exit
E68D 7D                 LD      A,L             ; Get L
E68E 93                 SUB     E               ; Compare with E
E68F C9                 RET                     ; Return status

E690 7E         CHKSYN: LD      A,(HL)          ; Check syntax of character
E691 E3                 EX      (SP),HL         ; Address of test byte
E692 BE                 CP      (HL)            ; Same as in code string?
E693 23                 INC     HL              ; Return address
E694 E3                 EX      (SP),HL         ; Put it back
E695 CA36E8             JP      Z,GETCHR        ; Yes - Get next character
E698 C3ADE3             JP      SNERR           ; Different - ?SN Error
```

```
E69B F5          OUTC:   PUSH    AF              ; Save character
E69C 3A4510              LD      A,(CTLOFG)      ; Get control "O" flag
E69F B7                  OR      A               ; Is it set?
E6A0 C245F2              JP      NZ,POPAF        ; Yes - don't output
E6A3 F1                  POP     AF              ; Restore character
E6A4 C5                  PUSH    BC              ; Save buffer length
E6A5 F5                  PUSH    AF              ; Save character
E6A6 FE20                CP      " "             ; Is it a control code?
E6A8 DABFE6              JP      C,DINPOS        ; Yes - Don't INC POS(X)
E6AB 3A4210              LD      A,(LWIDTH)      ; Get line width
E6AE 47                  LD      B,A             ; To B
E6AF 3AAB10              LD      A,(CURPOS)      ; Get cursor position
E6B2 04                  INC     B               ; Width 255?
E6B3 CABBE6              JP      Z,INCLEN        ; Yes - No width limit
E6B6 05                  DEC     B               ; Restore width
E6B7 B8                  CP      B               ; At end of line?
E6B8 CC81EB              CALL    Z,PRNTCR        ; Yes - output CRLF
E6BB 3C          INCLEN: INC     A               ; Move on one character
E6BC 32AB10              LD      (CURPOS),A      ; Save new position
E6BF F1          DINPOS: POP     AF              ; Restore character
E6C0 C1                  POP     BC              ; Restore buffer length
E6C1 F5                  PUSH    AF              ; << This sequence >>
E6C2 F1                  POP     AF              ; << is not needed >>
E6C3 F5                  PUSH    AF              ; Save character
E6C4 C5                  PUSH    BC              ; Save buffer length
E6C5 4F                  LD      C,A             ; Character to C
E6C6 CDD9FC              CALL    CONMON          ; Send it
E6C9 C1                  POP     BC              ; Restore buffer length
E6CA F1                  POP     AF              ; Restore character
E6CB C9                  RET

E6CC CD05FD      CLOTST: CALL    GETINP          ; Get input character
E6CF E67F                AND     01111111B       ; Strip bit 7
E6D1 FE0F                CP      CTRLO           ; Is it control "O"?
E6D3 C0                  RET     NZ              ; No don't flip flag
E6D4 3A4510              LD      A,(CTLOFG)      ; Get flag
E6D7 2F                  CPL                     ; Flip it
E6D8 324510              LD      (CTLOFG),A      ; Put it back
E6DB AF                  XOR     A               ; Null character
E6DC C9                  RET
```

```
E6DD CDA5E9     LIST:    CALL    ATOH            ; ASCII number to DE
E6E0 C0                  RET     NZ              ; Return if anything extra
E6E1 C1                  POP     BC              ; Rubbish - Not needed
E6E2 CD99E4              CALL    SRCHLN          ; Search for line number in DE
E6E5 C5                  PUSH    BC              ; Save address of line
E6E6 CD33E7              CALL    SETLIN          ; Set up lines counter
E6E9 E1         LISTLP:  POP     HL              ; Restore address of line
E6EA 4E                  LD      C,(HL)          ; Get LSB of next line
E6EB 23                  INC     HL
E6EC 46                  LD      B,(HL)          ; Get MSB of next line
E6ED 23                  INC     HL
E6EE 78                  LD      A,B             ; BC = 0 (End of program)?
E6EF B1                  OR      C
E6F0 CAF8E3              JP      Z,PRNTOK        ; Yes - Go to command mode
E6F3 CD46E7              CALL    COUNT           ; Count lines
E6F6 CD61E8              CALL    TSTBRK          ; Test for break key
E6F9 C5                  PUSH    BC              ; Save address of next line
E6FA CD81EB              CALL    PRNTCR          ; Output CRLF
E6FD 5E                  LD      E,(HL)          ; Get LSB of line number
E6FE 23                  INC     HL
E6FF 56                  LD      D,(HL)          ; Get MSB of line number
E700 23                  INC     HL
E701 E5                  PUSH    HL              ; Save address of line start
E702 EB                  EX      DE,HL           ; Line number to HL
E703 CDADF9              CALL    PRNTHL          ; Output line number in decimal
E706 3E20                LD      A," "           ; Space after line number
E708 E1                  POP     HL              ; Restore start of line address
E709 CD9BE6     LSTLP2:  CALL    OUTC            ; Output character in A
E70C 7E         LSTLP3:  LD      A,(HL)          ; Get next byte in line
E70D B7                  OR      A               ; End of line?
E70E 23                  INC     HL              ; To next byte in line
E70F CAE9E6              JP      Z,LISTLP        ; Yes - get next line
E712 F209E7              JP      P,LSTLP2        ; No token - output it
E715 D67F                SUB     ZEND-1          ; Find and output word
E717 4F                  LD      C,A             ; Token offset+1 to C
E718 1143E1              LD      DE,WORDS        ; Reserved word list
E71B 1A         FNDTOK:  LD      A,(DE)          ; Get character in list
E71C 13                  INC     DE              ; Move on to next
E71D B7                  OR      A               ; Is it start of word?
E71E F21BE7              JP      P,FNDTOK        ; No - Keep looking for word
E721 0D                  DEC     C               ; Count words
E722 C21BE7              JP      NZ,FNDTOK       ; Not there - keep looking
E725 E67F       OUTWRD:  AND     01111111B       ; Strip bit 7
E727 CD9BE6              CALL    OUTC            ; Output first character
E72A 1A                  LD      A,(DE)          ; Get next character
E72B 13                  INC     DE              ; Move on to next
E72C B7                  OR      A               ; Is it end of word?
E72D F225E7              JP      P,OUTWRD        ; No - output the rest
E730 C30CE7              JP      LSTLP3          ; Next byte in line
```

```
E733 E5          SETLIN: PUSH    HL              ; Set up LINES counter
E734 2A4810              LD      HL,(LINESN)     ; Get LINES number
E737 224610              LD      (LINESC),HL     ; Save in LINES counter
E73A E1                  POP     HL
E73B C9                  RET

E73C 21DEFE      LDNMI1: LD      HL,BREAK        ; Break routine
E73F 227E0C              LD      (NMI),HL        ; NMI forces break
E742 C3F8E3              JP      PRNTOK          ; Go to command mode

E745 FE                  DEFB    (CP n)          ; <<< NO REFERENCE TO HERE >>>

E746 E5          COUNT:  PUSH    HL              ; Save code string address
E747 D5                  PUSH    DE
E748 2A4610              LD      HL,(LINESC)     ; Get LINES counter
E74B 11FFFF              LD      DE,-1
E74E ED5A                ADC     HL,DE           ; Decrement
E750 224610              LD      (LINESC),HL     ; Put it back
E753 D1                  POP     DE
E754 E1                  POP     HL              ; Restore code string address
E755 F0                  RET     P               ; Return if more lines to go
E756 E5                  PUSH    HL              ; Save code string address
E757 2A4810              LD      HL,(LINESN)     ; Get LINES number
E75A 224610              LD      (LINESC),HL     ; Reset LINES counter
E75D 3A4C10              LD      A,(NMIFLG)      ; Break by NMI?
E760 B7                  OR      A
E761 C2E5FE              JP      NZ,ARETN        ; Yes - "RETN"
E764 CD05FD              CALL    GETINP          ; Get input character
E767 FE03                CP      CTRLC           ; Is it control "C"?
E769 CA70E7              JP      Z,RSLNBK        ; Yes - Reset LINES an break
E76C E1                  POP     HL              ; Restore code string address
E76D C346E7              JP      COUNT           ; Keep on counting

E770 2A4810      RSLNBK: LD      HL,(LINESN)     ; Get LINES number
E773 224610              LD      (LINESC),HL     ; Reset LINES countr
E776 C3B1E0              JP      BRKRET          ; Go and output "Break"
```

```
E779 3E64      FOR:    LD      A,64H           ; Flag "FOR" assignment
E77B 32CB10            LD      (FORFLG),A      ; Save "FOR" flag
E77E CD87EA            CALL    LET             ; Set up initial index
E781 C1                POP     BC              ; Drop RETurn address
E782 E5                PUSH    HL              ; Save code string address
E783 CD70EA            CALL    DATA            ; Get next statement address
E786 22C710            LD      (LOOPST),HL     ; Save it for start of loop
E789 210200            LD      HL,2            ; Offset for "FOR" block
E78C 39                ADD     HL,SP           ; Point to it
E78D CD5AE3    FORSLP: CALL    LOKFOR          ; Look for existing "FOR" block
E790 D1                POP     DE              ; Get code string address
E791 C2A9E7            JP      NZ,FORFND       ; No nesting found
E794 09                ADD     HL,BC           ; Move into "FOR" block
E795 D5                PUSH    DE              ; Save code string address
E796 2B                DEC     HL
E797 56                LD      D,(HL)          ; Get MSB of loop statement
E798 2B                DEC     HL
E799 5E                LD      E,(HL)          ; Get LSB of loop statement
E79A 23                INC     HL
E79B 23                INC     HL
E79C E5                PUSH    HL              ; Save block address
E79D 2AC710            LD      HL,(LOOPST)     ; Get address of loop statement
E7A0 CD8AE6            CALL    CPDEHL          ; Compare the FOR loops
E7A3 E1                POP     HL              ; Restore block address
E7A4 C28DE7            JP      NZ,FORSLP       ; Different FORs - Find another
E7A7 D1                POP     DE              ; Restore code string address
E7A8 F9                LD      SP,HL           ; Remove all nested loops
```

```
E7A9 EB          FORFND: EX      DE,HL           ; Code string address to HL
E7AA 0E08                LD      C,8
E7AC CD8AE3              CALL    CHKSTK          ; Check for 8 levels of stack
E7AF E5                  PUSH    HL              ; Save code string address
E7B0 2AC710              LD      HL,(LOOPST)     ; Get first statement of loop
E7B3 E3                  EX      (SP),HL         ; Save and restore code string
E7B4 E5                  PUSH    HL              ; Re-save code string address
E7B5 2A5C10              LD      HL,(LINEAT)     ; Get current line number
E7B8 E3                  EX      (SP),HL         ; Save and restore code string
E7B9 CD44ED              CALL    TSTNUM          ; Make sure it's a number
E7BC CD90E6              CALL    CHKSYN          ; Make sure "TO" is next
E7BF A6                  DEFB    ZTO             ; "TO" token
E7C0 CD41ED              CALL    GETNUM          ; Get "TO" expression value
E7C3 E5                  PUSH    HL              ; Save code string address
E7C4 CD5FF8              CALL    BCDEFP          ; Move "TO" value to BCDE
E7C7 E1                  POP     HL              ; Restore code string address
E7C8 C5                  PUSH    BC              ; Save "TO" value in block
E7C9 D5                  PUSH    DE
E7CA 010081              LD      BC,8100H        ; BCDE = 1 (default STEP)
E7CD 51                  LD      D,C             ; C=0
E7CE 5A                  LD      E,D             ; D=0
E7CF 7E                  LD      A,(HL)          ; Get next byte in code string
E7D0 FEAB                CP      ZSTEP           ; See if "STEP" is stated
E7D2 3E01                LD      A,1             ; Sign of step = 1
E7D4 C2E5E7              JP      NZ,SAVSTP       ; No STEP given - Default to 1
E7D7 CD36E8              CALL    GETCHR          ; Jump over "STEP" token
E7DA CD41ED              CALL    GETNUM          ; Get step value
E7DD E5                  PUSH    HL              ; Save code string address
E7DE CD5FF8              CALL    BCDEFP          ; Move STEP to BCDE
E7E1 CD13F8              CALL    TSTSGN          ; Test sign of FPREG
E7E4 E1                  POP     HL              ; Restore code string address
E7E5 C5          SAVSTP: PUSH    BC              ; Save the STEP value in block
E7E6 D5                  PUSH    DE
E7E7 F5                  PUSH    AF              ; Save sign of STEP
E7E8 33                  INC     SP              ; Don't save flags
E7E9 E5                  PUSH    HL              ; Save code string address
E7EA 2ACE10              LD      HL,(BRKLIN)     ; Get address of index variable
E7ED E3                  EX      (SP),HL         ; Save and restore code string
E7EE 0681        PUTFID: LD      B,ZFOR          ; "FOR" block marker
E7F0 C5                  PUSH    BC              ; Save it
E7F1 33                  INC     SP              ; Don't save C
```

```
E7F2 CD40FD     RUNCNT: CALL    CHKBRK          ; Execution driver - Test break
E7F5 B7                 OR      A               ; Break key hit?
E7F6 C466E8             CALL    NZ,STALL        ; Yes - Pause for a key
E7F9 22CE10             LD      (BRKLIN),HL     ; Save code address for break
E7FC 7E                 LD      A,(HL)          ; Get next byte in code string
E7FD FE3A               CP      ":"             ; Multi statement line?
E7FF CA16E8             JP      Z,EXCUTE        ; Yes - Execute it
E802 B7                 OR      A               ; End of line?
E803 C2ADE3             JP      NZ,SNERR        ; No - Syntax error
E806 23                 INC     HL              ; Point to address of next line
E807 7E                 LD      A,(HL)          ; Get LSB of line pointer
E808 23                 INC     HL
E809 B6                 OR      (HL)            ; Is it zero (End of prog)?
E80A CA7AE8             JP      Z,ENDPRG        ; Yes - Terminate execution
E80D 23                 INC     HL              ; Point to line number
E80E 5E                 LD      E,(HL)          ; Get LSB of line number
E80F 23                 INC     HL
E810 56                 LD      D,(HL)          ; Get MSB of line number
E811 EB                 EX      DE,HL           ; Line number to HL
E812 225C10             LD      (LINEAT),HL     ; Save as current line number
E815 EB                 EX      DE,HL           ; Line number back to DE
E816 CD36E8     EXCUTE: CALL    GETCHR          ; Get key word
E819 11F2E7             LD      DE,RUNCNT       ; Where to RETurn to
E81C D5                 PUSH    DE              ; Save for RETurn
E81D C8         IFJMP:  RET     Z               ; Go to RUNCNT if end of STMT
E81E D680       ONJMP:  SUB     ZEND            ; Is it a token?
E820 DA87EA             JP      C,LET           ; No - try to assign it
E823 FE25               CP      ZNEW+1-ZEND     ; END to NEW ?
E825 D2ADE3             JP      NC,SNERR        ; Not a key word - ?SN Error
E828 07                 RLCA                    ; Double it
E829 4F                 LD      C,A             ; BC = Offset into table
E82A 0600               LD      B,0
E82C EB                 EX      DE,HL           ; Save code string address
E82D 215AE2             LD      HL,WORDTB       ; Keyword address table
E830 09                 ADD     HL,BC           ; Point to routine address
E831 4E                 LD      C,(HL)          ; Get LSB of routine address
E832 23                 INC     HL
E833 46                 LD      B,(HL)          ; Get MSB of routine address
E834 C5                 PUSH    BC              ; Save routine address
E835 EB                 EX      DE,HL           ; Restore code string address

E836 23         GETCHR: INC     HL              ; Point to next character
E837 7E                 LD      A,(HL)          ; Get next code string byte
E838 FE3A               CP      ":"             ; Z  if ":"
E83A D0                 RET     NC              ; NC if > "9"
E83B FE20               CP      " "
E83D CA36E8             JP      Z,GETCHR        ; Skip over spaces
E840 FE30               CP      "0"
E842 3F                 CCF                     ; NC if < "0"
E843 3C                 INC     A               ; Test for zero - Leave carry
E844 3D                 DEC     A               ; Z  if Null
E845 C9                 RET
```

```
E846 EB          RESTOR: EX      DE,HL           ; Save code string address
E847 2A5E10              LD      HL,(BASTXT)     ; Point to start of program
E84A CA5BE8              JP      Z,RESTNL        ; Just RESTORE - reset pointer
E84D EB                  EX      DE,HL           ; Restore code string address
E84E CDA5E9              CALL    ATOH            ; Get line number to DE
E851 E5                  PUSH    HL              ; Save code string address
E852 CD99E4              CALL    SRCHLN          ; Search for line number in DE
E855 60                  LD      H,B             ; HL = Address of line
E856 69                  LD      L,C
E857 D1                  POP     DE              ; Restore code string address
E858 D246EA              JP      NC,ULERR        ; ?UL Error if not found
E85B 2B          RESTNL: DEC     HL              ; Byte before DATA statement
E85C 22DC10      UPDATA: LD      (NXTDAT),HL     ; Update DATA pointer
E85F EB                  EX      DE,HL           ; Restore code string address
E860 C9                  RET

E861 CD40FD      TSTBRK: CALL    CHKBRK          ; Test for interrupts
E864 B7                  OR      A
E865 C8                  RET     Z               ; Return if no key pressed
E866 CDCCE6      STALL:  CALL    CLOTST          ; Get input and test for ^O
E869 FE13                CP      CTRLS           ; Is it control "S"
E86B CCCCE6              CALL    Z,CLOTST        ; Yes - Get another character
E86E FE03                CP      CTRLC           ; Return if not control "C"
E870 C0          STOP:   RET     NZ              ; Exit if anything else
E871 F6                  DEFB    (OR n)          ; Flag "STOP"
E872 C0          PEND:   RET     NZ              ; Exit if anything else
E873 22CE10              LD      (BRKLIN),HL     ; Save point of break
E876 21                  DEFB    (LD HL,nn)      ; Skip "OR 11111111B"
E877 F6FF        INPBRK: OR      11111111B       ; Flag "Break" wanted
E879 C1                  POP     BC              ; Return not needed and more
E87A 2A5C10      ENDPRG: LD      HL,(LINEAT)     ; Get current line number
E87D F5                  PUSH    AF              ; Save STOP / END status
E87E 7D                  LD      A,L             ; Is it direct break?
E87F A4                  AND     H
E880 3C                  INC     A               ; Line is -1 if direct break
E881 CA8DE8              JP      Z,NOLIN         ; Yes - No line number
E884 22D210              LD      (ERRLIN),HL     ; Save line of break
E887 2ACE10              LD      HL,(BRKLIN)     ; Get point of break
E88A 22D410              LD      (CONTAD),HL     ; Save point to CONTinue
E88D AF          NOLIN:  XOR     A
E88E 324510              LD      (CTLOFG),A      ; Enable output
E891 CD74EB              CALL    STTLIN          ; Start a new line
E894 F1                  POP     AF              ; Restore STOP / END status
E895 2150E3              LD      HL,BRKMSG       ; "Break" message
E898 C2E1E3              JP      NZ,ERRIN        ; "in line" wanted?
E89B C3F8E3              JP      PRNTOK          ; Go to command mode

E89E 2AD410      CONT:   LD      HL,(CONTAD)     ; Get CONTinue address
E8A1 7C                  LD      A,H             ; Is it zero?
E8A2 B5                  OR      L
E8A3 1E20                LD      E,CN            ; ?CN Error
E8A5 CAC1E3              JP      Z,ERROR         ; Yes - output "?CN Error"
E8A8 EB                  EX      DE,HL           ; Save code string address
E8A9 2AD210              LD      HL,(ERRLIN)     ; Get line of last break
E8AC 225C10              LD      (LINEAT),HL     ; Set up current line number
E8AF EB                  EX      DE,HL           ; Restore code string addres
E8B0 C9                  RET                     ; CONTinue where left off
```

```
E8B1 CD84F4     NULL:   CALL    GETINT          ; Get integer 0-255
E8B4 C0                 RET     NZ              ; Return if bad value
E8B5 324110             LD      (NULLS),A       ; Set nulls number
E8B8 C9                 RET

E8B9 06FF       ARRLD1: LD      B,-1            ; Flag array load
E8BB CD36E8     ARRSV1: CALL    GETCHR          ; Skip "*"
E8BE 78                 LD      A,B             ; CLOAD* or CSAVE*
E8BF 32CE10             LD      (BRKLIN),A      ; Save it
E8C2 3E01               LD      A,1             ; It's an array
E8C4 32CB10             LD      (FORFLG),A      ; Flag array name
E8C7 CD2DEF             CALL    GETVAR          ; Get address of array name
E8CA E5                 PUSH    HL              ; Save code string address
E8CB 32CB10             LD      (FORFLG),A      ; Clear flag
E8CE 60                 LD      H,B             ; Address of array to HL
E8CF 69                 LD      L,C
E8D0 0B                 DEC     BC              ; Back space
E8D1 0B                 DEC     BC              ;  to point
E8D2 0B                 DEC     BC              ;  to the
E8D3 0B                 DEC     BC              ; array name
E8D4 3ACE10             LD      A,(BRKLIN)      ; CLOAD* or CSAVE* ?
E8D7 B7                 OR      A
E8D8 F5                 PUSH    AF              ; Save CLOAD* / CSAVE* status
E8D9 EB                 EX      DE,HL           ; Array data length
E8DA 19                 ADD     HL,DE           ; End of data
E8DB EB                 EX      DE,HL           ; To DE
E8DC 4E                 LD      C,(HL)          ; Get dimension bytes
E8DD 0600               LD      B,0
E8DF 09                 ADD     HL,BC           ; 2 Bytes each dimension
E8E0 09                 ADD     HL,BC
E8E1 23                 INC     HL              ; Over number of dimensions
E8E2 E5                 PUSH    HL              ; Address of array data
E8E3 D5                 PUSH    DE              ; End of array data
E8E4 C5                 PUSH    BC              ; Number of dimensions
E8E5 3ACE10             LD      A,(BRKLIN)      ; CLOAD* or CSAVE* ?
E8E8 FEFF               CP      -1
E8EA CCD5FC             CALL    Z,CASFF         ; CLOAD* - Cassette on
E8ED 3ACE10             LD      A,(BRKLIN)      ; CLOAD* or CSAVE* ?
E8F0 FEFF               CP      -1
E8F2 C4C8FC             CALL    NZ,CASFFW       ; CSAVE* - Cassette on and wait
E8F5 00                 NOP
E8F6 00                 NOP
E8F7 00                 NOP
E8F8 210000             LD      HL,0
E8FB 224A10             LD      (CHKSUM),HL     ; Zero check sum
E8FE C1                 POP     BC              ; Number of dimensions
E8FF D1                 POP     DE              ; End of array data
E900 E1                 POP     HL              ; Address of array data
E901 06D2               LD      B,11010010B     ; Header byte
E903 C3D6FF             JP      JPLDSV          ; CSAVE-SNDHDR , CLOAD-GETHDR

E906 78         SNDHDR: LD      A,B             ; Get header byte
E907 CDB7F4             CALL    WUART2          ; Send 2 bytes to UART
E90A CDB7F4             CALL    WUART2          ; Send 2 bytes to UART
E90D C31DE9             JP      SNDARY          ; Send array data
```

```
E910 0E04       GETHDR: LD      C,4             ; 4 Bytes to check
E912 CDB4F4     HDRLP:  CALL    RUART           ; Read byte from UART
E915 B8                 CP      B               ; Same as header?
E916 C210E9             JP      NZ,GETHDR       ; No - Wait for another
E919 0D                 DEC     C               ; Count bytes
E91A C212E9             JP      NZ,HDRLP        ; More needed
E91D CD44ED     SNDARY: CALL    TSTNUM          ; Check it's a numerical array
E920 CD8AE6     ARYLP:  CALL    CPDEHL          ; All array data done
E923 CA37E9             JP      Z,SUMOFF        ; Yes - Do check sum
E926 F1                 POP     AF              ; CLOAD* or CSAVE* ?
E927 F5                 PUSH    AF              ; Re-save flags
E928 7E                 LD      A,(HL)          ; Get byte
E929 F4BAF4             CALL    P,WUART         ; CSAVE* - Write byte
E92C FCB4F4             CALL    M,RUART         ; CLOAD* - Read byte
E92F 77                 LD      (HL),A          ; Save byte in case of CLOAD*
E930 CD40E9             CALL    ACCSUM          ; Accumulate check sum
E933 23                 INC     HL              ; Next byte
E934 C320E9             JP      ARYLP           ; Repeat

E937 CD4DE9     SUMOFF: CALL    DOSUM           ; Do check sum
E93A CDD5FC             CALL    CASFF           ; Cassette off
E93D F1                 POP     AF              ; Not needed any more
E93E E1                 POP     HL              ; Restore code string address
E93F C9                 RET

E940 E5         ACCSUM: PUSH    HL              ; Save address in array
E941 2A4A10             LD      HL,(CHKSUM)     ; Get check sum
E944 0600               LD      B,0             ; BC = Value of byte
E946 4F                 LD      C,A
E947 09                 ADD     HL,BC           ; Add byte to check sum
E948 224A10             LD      (CHKSUM),HL     ; Re-save check sum
E94B E1                 POP     HL              ; Restore address in array
E94C C9                 RET

E94D 3ACE10     DOSUM:  LD      A,(BRKLIN)      ; CLOAD* or CSAVE* ?
E950 B7                 OR      A
E951 FA60E9             JP      M,CHSUMS        ; CLOAD* - Check if sums match
E954 3A4A10             LD      A,(CHKSUM)      ; Get LSB of check sum
E957 CDBAF4             CALL    WUART           ; Write to UART
E95A 3A4B10             LD      A,(CHKSUM+1)    ; Get MSB of check sum
E95D C3BAF4             JP      WUART           ; Write to UART and return

E960 CDB4F4     CHSUMS: CALL    RUART           ; Read LSB of check sum
E963 F5                 PUSH    AF              ; Save it
E964 CDB4F4             CALL    RUART           ; Read MSB of check sum
E967 C1                 POP     BC              ; LSB to B
E968 58                 LD      E,B             ; LSB to E
E969 57                 LD      D,A             ; MSB to D
E96A 2A4A10             LD      HL,(CHKSUM)     ; Get accumulated check sum
E96D CD8AE6             CALL    CPDEHL          ; Are they the same?
E970 C8                 RET     Z               ; Yes - End CLOAD*
E971 CDD5FC             CALL    CASFF           ; Cassette off
E974 C36BF5             JP      OUTBAD          ; Different - Output "Bad"
```

```
E977 7E        CHKLTR: LD      A,(HL)          ; Get byte
E978 FE41              CP      "A"             ; < "A" ?
E97A D8                RET     C               ; Carry set if not letter
E97B FE5B              CP      "Z"+1           ; > "Z" ?
E97D 3F                CCF
E97E C9                RET                     ; Carry set if not letter

E97F CD36E8    FPSINT: CALL    GETCHR          ; Get next character
E982 CD41ED    POSINT: CALL    GETNUM          ; Get integer 0 to 32767
E985 CD13F8    DEPINT: CALL    TSTSGN          ; Test sign of FPREG
E988 FAA0E9            JP      M,FCERR         ; Negative - ?FC Error
E98B 3AE710    DEINT:  LD      A,(FPEXP)       ; Get integer value to DE
E98E FE90              CP      80H+16          ; Exponent in range (16 bits)?
E990 DABBF8            JP      C,FPINT         ; Yes - convert it
E993 018090            LD      BC,9080H        ; BCDE = -32768
E996 110000            LD      DE,0000
E999 E5                PUSH    HL              ; Save code string address
E99A CD8EF8            CALL    CMPNUM          ; Compare FPREG with BCDE
E99D E1                POP     HL              ; Restore code string address
E99E 51                LD      D,C             ; MSB to D
E99F C8                RET     Z               ; Return if in range
E9A0 1E08      FCERR:  LD      E,FC            ; ?FC Error
E9A2 C3C1E3            JP      ERROR           ; Output error.

E9A5 2B        ATOH:   DEC     HL              ; ASCII number to DE binary
E9A6 110000    GETLN:  LD      DE,0            ; Get number to DE
E9A9 CD36E8    GTLNLP: CALL    GETCHR          ; Get next character
E9AC D0                RET     NC              ; Exit if not a digit
E9AD E5                PUSH    HL              ; Save code string address
E9AE F5                PUSH    AF              ; Save digit
E9AF 219819            LD      HL,65529/10     ; Largest number 65529
E9B2 CD8AE6            CALL    CPDEHL          ; Number in range?
E9B5 DAADE3            JP      C,SNERR         ; No - ?SN Error
E9B8 62                LD      H,D             ; HL = Number
E9B9 6B                LD      L,E
E9BA 19                ADD     HL,DE           ; Times 2
E9BB 29                ADD     HL,HL           ; Times 4
E9BC 19                ADD     HL,DE           ; Times 5
E9BD 29                ADD     HL,HL           ; Times 10
E9BE F1                POP     AF              ; Restore digit
E9BF D630              SUB     "0"             ; Make it 0 to 9
E9C1 5F                LD      E,A             ; DE = Value of digit
E9C2 1600              LD      D,0
E9C4 19                ADD     HL,DE           ; Add to number
E9C5 EB                EX      DE,HL           ; Number to DE
E9C6 E1                POP     HL              ; Restore code string address
E9C7 C3A9E9            JP      GTLNLP          ; Go to next character
```

```
E9CA CAC9E4     CLEAR:  JP      Z,INTVAR        ; Just "CLEAR" Keep parameters
E9CD CD82E9             CALL    POSINT          ; Get integer 0 to 32767 to DE
E9D0 2B                 DEC     HL              ; Cancel increment
E9D1 CD36E8             CALL    GETCHR          ; Get next character
E9D4 E5                 PUSH    HL              ; Save code string address
E9D5 2AAF10             LD      HL,(LSTRAM)     ; Get end of RAM
E9D8 CAEDE9             JP      Z,STORED        ; No value given - Use stored
E9DB E1                 POP     HL              ; Restore code string address
E9DC CD90E6             CALL    CHKSYN          ; Check for comma
E9DF 2C                 DEFB    ","
E9E0 D5                 PUSH    DE              ; Save number
E9E1 CD82E9             CALL    POSINT          ; Get integer 0 to 32767
E9E4 2B                 DEC     HL              ; Cancel increment
E9E5 CD36E8             CALL    GETCHR          ; Get next character
E9E8 C2ADE3             JP      NZ,SNERR        ; ?SN Error if more on line
E9EB E3                 EX      (SP),HL         ; Save code string address
E9EC EB                 EX      DE,HL           ; Number to DE
E9ED 7D         STORED: LD      A,L             ; Get LSB of new RAM top
E9EE 93                 SUB     E               ; Subtract LSB of string space
E9EF 5F                 LD      E,A             ; Save LSB
E9F0 7C                 LD      A,H             ; Get MSB of new RAM top
E9F1 9A                 SBC     A,D             ; Subtract MSB of string space
E9F2 57                 LD      D,A             ; Save MSB
E9F3 DAA2E3             JP      C,OMERR         ; ?OM Error if not enough mem
E9F6 E5                 PUSH    HL              ; Save RAM top
E9F7 2AD610             LD      HL,(PROGND)     ; Get program end
E9FA 012800             LD      BC,40           ; 40 Bytes minimum working RAM
E9FD 09                 ADD     HL,BC           ; Get lowest address
E9FE CD8AE6             CALL    CPDEHL          ; Enough memory?
EA01 D2A2E3             JP      NC,OMERR        ; No - ?OM Error
EA04 EB                 EX      DE,HL           ; RAM top to HL
EA05 225A10             LD      (STRSPC),HL     ; Set new string space
EA08 E1                 POP     HL              ; End of memory to use
EA09 22AF10             LD      (LSTRAM),HL     ; Set new top of RAM
EA0C E1                 POP     HL              ; Restore code string address
EA0D C3C9E4             JP      INTVAR          ; Initialise variables
```

```
EA10 CAC5E4     RUN:    JP      Z,RUNFST        ; RUN from start if just RUN
EA13 CDC9E4             CALL    INTVAR          ; Initialise variables
EA16 01F2E7             LD      BC,RUNCNT       ; Execution driver loop
EA19 C32CEA             JP      RUNLIN          ; RUN from line number

EA1C 0E03       GOSUB:  LD      C,3             ; 3 Levels of stack needed
EA1E CD8AE3             CALL    CHKSTK          ; Check for 3 levels of stack
EA21 C1                 POP     BC              ; Get return address
EA22 E5                 PUSH    HL              ; Save code string for RETURN
EA23 E5                 PUSH    HL              ; And for GOSUB routine
EA24 2A5C10             LD      HL,(LINEAT)     ; Get current line
EA27 E3                 EX      (SP),HL         ; Into stack - Code string out
EA28 3E8C               LD      A,ZGOSUB        ; "GOSUB" token
EA2A F5                 PUSH    AF              ; Save token
EA2B 33                 INC     SP              ; Don't save flags

EA2C C5         RUNLIN: PUSH    BC              ; Save return address
EA2D CDA5E9     GOTO:   CALL    ATOH            ; ASCII number to DE binary
EA30 CD72EA             CALL    REM             ; Get end of line
EA33 E5                 PUSH    HL              ; Save end of line
EA34 2A5C10             LD      HL,(LINEAT)     ; Get current line
EA37 CD8AE6             CALL    CPDEHL          ; Line after current?
EA3A E1                 POP     HL              ; Restore end of line
EA3B 23                 INC     HL              ; Start of next line
EA3C DC9CE4             CALL    C,SRCHLP        ; Line is after current line
EA3F D499E4             CALL    NC,SRCHLN       ; Line is before current line
EA42 60                 LD      H,B             ; Set up code string address
EA43 69                 LD      L,C
EA44 2B                 DEC     HL              ; Incremented after
EA45 D8                 RET     C               ; Line found
EA46 1E0E       ULERR:  LD      E,UL            ; ?UL Error
EA48 C3C1E3             JP      ERROR           ; Output error message

EA4B C0         RETURN: RET     NZ              ; Return if not just RETURN
EA4C 16FF               LD      D,-1            ; Flag "GOSUB" search
EA4E CD56E3             CALL    BAKSTK          ; Look "GOSUB" block
EA51 F9                 LD      SP,HL           ; Kill all FORs in subroutine
EA52 FE8C               CP      ZGOSUB          ; Test for "GOSUB" token
EA54 1E04               LD      E,RG            ; ?RG Error
EA56 C2C1E3             JP      NZ,ERROR        ; Error if no "GOSUB" found
EA59 E1                 POP     HL              ; Get RETURN line number
EA5A 225C10             LD      (LINEAT),HL     ; Save as current
EA5D 23                 INC     HL              ; Was it from direct statement?
EA5E 7C                 LD      A,H
EA5F B5                 OR      L
EA60 C26AEA             JP      NZ,RETLIN       ; No - Return to line
EA63 3ACC10             LD      A,(LSTBIN)      ; Any INPUT in subroutine?
EA66 B7                 OR      A               ; If so buffer is corrupted
EA67 C2F7E3             JP      NZ,POPNOK       ; Yes - Go to command mode
EA6A 21F2E7     RETLIN: LD      HL,RUNCNT       ; Execution driver loop
EA6D E3                 EX      (SP),HL         ; Into stack - Code string out
EA6E 3E                 DEFB    (LD A,n)        ; Skip "POP HL"
EA6F E1         NXTDTA: POP     HL              ; Restore code string address
```

```
EA70 013A       DATA:   DEFB    (LD BC,":")     ; ":" End of statement
EA72 0E00       REM:    LD      C,0             ; 00  End of statement
EA74 0600               LD      B,0
EA76 79         NXTSTL: LD      A,C             ; Statement end byte
EA77 48                 LD      C,B
EA78 47                 LD      B,A             ; Statement end byte
EA79 7E         NXTSTT: LD      A,(HL)          ; Get byte
EA7A B7                 OR      A               ; End of line?
EA7B C8                 RET     Z               ; Yes - Exit
EA7C B8                 CP      B               ; End of statement?
EA7D C8                 RET     Z               ; Yes - Exit
EA7E 23                 INC     HL              ; Next byte
EA7F FE22               CP      '"'             ; Literal string?
EA81 CA76EA             JP      Z,NXTSTL        ; Yes - Look for another '"'
EA84 C379EA             JP      NXTSTT          ; Keep looking

EA87 CD2DEF     LET:    CALL    GETVAR          ; Get variable name
EA8A CD90E6             CALL    CHKSYN          ; Make sure "=" follows
EA8D B4                 DEFB    ZEQUAL          ; "=" token
EA8E D5                 PUSH    DE              ; Save address of variable
EA8F 3AAD10             LD      A,(TYPE)        ; Get data type
EA92 F5                 PUSH    AF              ; Save type
EA93 CD5AED             CALL    EVAL            ; Evaluate expression
EA96 F1                 POP     AF              ; Restore type
EA97 E3                 EX      (SP),HL         ; Save code - Get var addr
EA98 22CE10             LD      (BRKLIN),HL     ; Save address of variable
EA9B 1F                 RRA                     ; Adjust type
EA9C CD46ED             CALL    CHKTYP          ; Check types are the same
EA9F CADAEA             JP      Z,LETNUM        ; Numeric - Move value
EAA2 E5         LETSTR: PUSH    HL              ; Save address of string var
EAA3 2AE410             LD      HL,(FPREG)      ; Pointer to string entry
EAA6 E5                 PUSH    HL              ; Save it on stack
EAA7 23                 INC     HL              ; Skip over length
EAA8 23                 INC     HL
EAA9 5E                 LD      E,(HL)          ; LSB of string address
EAAA 23                 INC     HL
EAAB 56                 LD      D,(HL)          ; MSB of string address
EAAC 2A5E10             LD      HL,(BASTXT)     ; Point to start of program
EAAF CD8AE6             CALL    CPDEHL          ; Is string before program?
EAB2 D2C9EA             JP      NC,CRESTR       ; Yes - Create string entry
EAB5 2A5A10             LD      HL,(STRSPC)     ; Point to string space
EAB8 CD8AE6             CALL    CPDEHL          ; Is string literal in program?
EABB D1                 POP     DE              ; Restore address of string
EABC D2D1EA             JP      NC,MVSTPT       ; Yes - Set up pointer
EABF 21BF10             LD      HL,TMPSTR       ; Temporary string pool
EAC2 CD8AE6             CALL    CPDEHL          ; Is string in temporary pool?
EAC5 D2D1EA             JP      NC,MVSTPT       ; No - Set up pointer
EAC8 3E                 DEFB    (LD A,n)        ; Skip "POP DE"
EAC9 D1         CRESTR: POP     DE              ; Restore address of string
EACA CD71F3             CALL    BAKTMP          ; Back to last tmp-str entry
EACD EB                 EX      DE,HL           ; Address of string entry
EACE CDAAF1             CALL    SAVSTR          ; Save string in string area
EAD1 CD71F3     MVSTPT: CALL    BAKTMP          ; Back to last tmp-str entry
EAD4 E1                 POP     HL              ; Get string pointer
EAD5 CD6EF8             CALL    DETHL4          ; Move string pointer to var
EAD8 E1                 POP     HL              ; Restore code string address
EAD9 C9                 RET
```

```
EADA E5          LETNUM: PUSH    HL              ; Save address of variable
EADB CD6BF8              CALL    FPTHL           ; Move value to variable
EADE D1                  POP     DE              ; Restore address of variable
EADF E1                  POP     HL              ; Restore code string address
EAE0 C9                  RET

EAE1 CD84F4      ON:     CALL    GETINT          ; Get integer 0-255
EAE4 7E                  LD      A,(HL)          ; Get "GOTO" or "GOSUB" token
EAE5 47                  LD      B,A             ; Save in B
EAE6 FE8C                CP      ZGOSUB          ; "GOSUB" token?
EAE8 CAF0EA              JP      Z,ONGO          ; Yes - Find line number
EAEB CD90E6              CALL    CHKSYN          ; Make sure it's "GOTO"
EAEE 88                  DEFB    ZGOTO           ; "GOTO" token
EAEF 2B                  DEC     HL              ; Cancel increment
EAF0 4B          ONGO:   LD      C,E             ; Integer of branch value
EAF1 0D          ONGOLP: DEC     C               ; Count branches
EAF2 78                  LD      A,B             ; Get "GOTO" or "GOSUB" token
EAF3 CA1EE8              JP      Z,ONJMP         ; Go to that line if right one
EAF6 CDA6E9              CALL    GETLN           ; Get line number to DE
EAF9 FE2C                CP      ","             ; Another line number?
EAFB C0                  RET     NZ              ; No - Drop through
EAFC C3F1EA              JP      ONGOLP          ; Yes - loop

EAFF CD5AED      IF:     CALL    EVAL            ; Evaluate expression
EB02 7E                  LD      A,(HL)          ; Get token
EB03 FE88                CP      ZGOTO           ; "GOTO" token?
EB05 CA0DEB              JP      Z,IFGO          ; Yes - Get line
EB08 CD90E6              CALL    CHKSYN          ; Make sure it's "THEN"
EB0B A9                  DEFB    ZTHEN           ; "THEN" token
EB0C 2B                  DEC     HL              ; Cancel increment
EB0D CD44ED      IFGO:   CALL    TSTNUM          ; Make sure it's numeric
EB10 CD13F8              CALL    TSTSGN          ; Test state of expression
EB13 CA72EA              JP      Z,REM           ; False - Drop through
EB16 CD36E8              CALL    GETCHR          ; Get next character
EB19 DA2DEA              JP      C,GOTO          ; Number - GOTO that line
EB1C C31DE8              JP      IFJMP           ; Otherwise do statement
```

# NASCOM

# ROM

# BASIC

# DIS–ASSEMBLED

## PART 4

BY CARL LLOYD–PARKER

```
EB1F 2B          MRPRNT: DEC     HL              ; DEC 'cos GETCHR INCs
EB20 CD36E8              CALL    GETCHR          ; Get next character
EB23 CA81EB      PRINT:  JP      Z,PRNTCR        ; CRLF if just PRINT
EB26 C8          PRNTLP: RET     Z               ; End of list - Exit
EB27 FEA5                CP      ZTAB            ; "TAB(" token?
EB29 CAAFEB              JP      Z,DOTAB         ; Yes - Do TAB routine
EB2C FEA8                CP      ZSPC            ; "SPC(" token?
EB2E CAAFEB              JP      Z,DOTAB         ; Yes - Do SPC routine
EB31 E5                  PUSH    HL              ; Save code string address
EB32 FE2C                CP      ","             ; Comma?
EB34 CA98EB              JP      Z,DOCOM         ; Yes - Move to next zone
EB37 FE3B                CP      ";"             ; Semi-colon?
EB39 CAD2EB              JP      Z,NEXITM        ; Do semi-colon routine
EB3C C1                  POP     BC              ; Code string address to BC
EB3D CD5AED              CALL    EVAL            ; Evaluate expression
EB40 E5                  PUSH    HL              ; Save code string address
EB41 3AAD10              LD      A,(TYPE)        ; Get variable type
EB44 B7                  OR      A               ; Is it a string variable?
EB45 C26DEB              JP      NZ,PRNTST       ; Yes - Output string contents
EB48 CDB8F9              CALL    NUMASC          ; Convert number to text
EB4B CDCEF1              CALL    CRTST           ; Create temporary string
EB4E 3620                LD      (HL)," "        ; Followed by a space
EB50 2AE410              LD      HL,(FPREG)      ; Get length of output
EB53 34                  INC     (HL)            ; Plus 1 for the space
EB54 2AE410              LD      HL,(FPREG)      ; < Not needed >
EB57 3A4210              LD      A,(LWIDTH)      ; Get width of line
EB5A 47                  LD      B,A             ; To B
EB5B 04                  INC     B               ; Width 255 (No limit)?
EB5C CA69EB              JP      Z,PRNTNB        ; Yes - Output number string
EB5F 04                  INC     B               ; Adjust it
EB60 3AAB10              LD      A,(CURPOS)      ; Get cursor position
EB63 86                  ADD     A,(HL)          ; Add length of string
EB64 3D                  DEC     A               ; Adjust it
EB65 B8                  CP      B               ; Will output fit on this line?
EB66 D481EB              CALL    NC,PRNTCR       ; No - CRLF first
EB69 CD13F2      PRNTNB: CALL    PRS1            ; Output string at (HL)
EB6C AF                  XOR     A               ; Skip CALL by setting "Z" flag
EB6D C413F2      PRNTST: CALL    NZ,PRS1         ; Output string at (HL)
EB70 E1                  POP     HL              ; Restore code string address
EB71 C31FEB              JP      MRPRNT          ; See if more to PRINT

EB74 3AAB10      STTLIN: LD      A,(CURPOS)      ; Make sure on new line
EB77 B7                  OR      A               ; Already at start?
EB78 C8                  RET     Z               ; Yes - Do nothing
EB79 C381EB              JP      PRNTCR          ; Start a new line
```

```
EB7C 3600.      ENDINP: LD      (HL),0          ; Mark end of buffer
EB7E 216010             LD      HL,BUFFER-1     ; Point to buffer
EB81 3E0D       PRNTCR: LD      A,CR            ; Load a CR
EB83 CD9BE6             CALL    OUTC            ; Output character
EB86 AF         DONULL: XOR     A               ; Set to position 0
EB87 32AB10             LD      (CURPOS),A      ; Store it
EB8A 3A4110             LD      A,(NULLS)       ; Get number of nulls
EB8D 3D         NULLP:  DEC     A               ; Count them
EB8E C8                 RET     Z               ; Return if done
EB8F F5                 PUSH    AF              ; Save count
EB90 AF                 XOR     A               ; Load a null
EB91 CD9BE6             CALL    OUTC            ; Output it
EB94 F1                 POP     AF              ; Restore count
EB95 C38DEB             JP      NULLP           ; Keep counting

EB98 3A4310     DOCOM:  LD      A,(COMMAN)      ; Get comma width
EB9B 47                 LD      B,A             ; Save in B
EB9C 3AAB10             LD      A,(CURPOS)      ; Get current position
EB9F B8                 CP      B               ; Within the limit?
EBA0 D481EB             CALL    NC,PRNTCR       ; No - output CRLF
EBA3 D2D2EB             JP      NC,NEXITM       ; Get next item
EBA6 D60E       ZONELP: SUB     14              ; Next zone of 14 characters
EBA8 D2A6EB             JP      NC,ZONELP       ; Repeat if more zones
EBAB 2F                 CPL                     ; Number of spaces to output
EBAC C3C7EB             JP      ASPCS           ; Output them

EBAF F5         DOTAB:  PUSH    AF              ; Save token
EBB0 CD81F4             CALL    FNDNUM          ; Evaluate expression
EBB3 CD90E6             CALL    CHKSYN          ; Make sure ")" follows
EBB6 29                 DEFB    ")"
EBB7 2B                 DEC     HL              ; Back space on to ")"
EBB8 F1                 POP     AF              ; Restore token
EBB9 D6A8               SUB     ZSPC            ; Was it "SPC(" ?
EBBB E5                 PUSH    HL              ; Save code string address
EBBC CAC2EB             JP      Z,DOSPC         ; Yes - Do "E" spaces
EBBF 3AAB10             LD      A,(CURPOS)      ; Get current position
EBC2 2F         DOSPC:  CPL                     ; Number of spaces to print to
EBC3 83                 ADD     A,E             ; Total number to print
EBC4 D2D2EB             JP      NC,NEXITM       ; TAB < Current POS(X)
EBC7 3C         ASPCS:  INC     A               ; Output A spaces
EBC8 47                 LD      B,A             ; Save number to print
EBC9 3E20               LD      A," "           ; Space
EBCB CD9BE6     SPCLP:  CALL    OUTC            ; Output character in A
EBCE 05                 DEC     B               ; Count them
EBCF C2CBEB             JP      NZ,SPCLP        ; Repeat if more
EBD2 E1         NEXITM: POP     HL              ; Restore code string address
EBD3 CD36E8             CALL    GETCHR          ; Get next character
EBD6 C326EB             JP      PRNTLP          ; More to print
```

```
EBD9 3F526564   REDO:   DEFB    "?Redo from start",CR,LF,0

EBEC 3ACD10     BADINP: LD      A,(READFG)      ; READ or INPUT?
EBEF B7                 OR      A
EBF0 C2A7E3             JP      NZ,DATSNR       ; READ - ?SN Error
EBF3 C1                 POP     BC              ; Throw away code string addr
EBF4 21D9EB             LD      HL,REDO         ; "Redo from start" message
EBF7 CD10F2             CALL    PRS             ; Output string
EBFA C3F8E4             JP      DOAGN           ; Do last INPUT again

EBFD CD7BF1     INPUT:  CALL    IDTEST          ; Test for illegal direct
EC00 7E                 LD      A,(HL)          ; Get character after "INPUT"
EC01 FE22               CP      '"'             ; Is there a prompt string?
EC03 3E00               LD      A,0             ; Clear A and leave flags
EC05 324510             LD      (CTLOFG),A      ; Enable output
EC08 C217EC             JP      NZ,NOPMPT       ; No prompt - get input
EC0B CDCFF1             CALL    QTSTR           ; Get string terminated by '"'
EC0E CD90E6             CALL    CHKSYN          ; Check for ";" after prompt
EC11 3B                 DEFB    ";"
EC12 E5                 PUSH    HL              ; Save code string address
EC13 CD13F2             CALL    PRS1            ; Output prompt string
EC16 3E                 DEFB    (LD A,n)        ; Skip "PUSH HL"
EC17 E5         NOPMPT: PUSH    HL              ; Save code string address
EC18 CDFCE4             CALL    PROMPT          ; Get input with "? " prompt
EC1B C1                 POP     BC              ; Restore code string address
EC1C DA77E8             JP      C,INPBRK        ; Break pressed - Exit
EC1F 23                 INC     HL              ; Next byte
EC20 7E                 LD      A,(HL)          ; Get it
EC21 B7                 OR      A               ; End of line?
EC22 2B                 DEC     HL              ; Back again
EC23 C5                 PUSH    BC              ; Re-save code string address
EC24 CA6FEA             JP      Z,NXTDTA        ; Yes - Find next DATA stmt
EC27 362C               LD      (HL),","        ; Store comma as separator
EC29 C331EC             JP      NXTITM          ; Get next item

EC2C E5         READ:   PUSH    HL              ; Save code string address
EC2D 2ADC10             LD      HL,(NXTDAT)     ; Next DATA statement
EC30 F6                 DEFB    (OR n)          ; Flag "READ"
EC31 AF         NXTITM: XOR     A               ; Flag "INPUT"
EC32 32CD10             LD      (READFG),A      ; Save "READ"/"INPUT" flag
EC35 E3                 EX      (SP),HL         ; Get code str' , Save pointer
EC36 C33DEC             JP      GTVLUS          ; Get values
```

```
EC39 CD90E6     NEDMOR: CALL    CHKSYN          ; Check for comma between items
EC3C 2C                 DEFB    ","
EC3D CD2DEF     GTVLUS: CALL    GETVAR          ; Get variable name
EC40 E3                 EX      (SP),HL         ; Save code str" , Get pointer
EC41 D5                 PUSH    DE              ; Save variable address
EC42 7E                 LD      A,(HL)          ; Get next "INPUT"/"DATA" byte
EC43 FE2C               CP      ","             ; Comma?
EC45 CA65EC             JP      Z,ANTVLU        ; Yes - Get another value
EC48 3ACD10             LD      A,(READFG)      ; Is it READ?
EC4B B7                 OR      A
EC4C C2D2EC             JP      NZ,FDTLP        ; Yes - Find next DATA stmt
EC4F 3E3F               LD      A,"?"           ; More INPUT needed
EC51 CD9BE6             CALL    OUTC            ; Output character
EC54 CDFCE4             CALL    PROMPT          ; Get INPUT with prompt
EC57 D1                 POP     DE              ; Variable address
EC58 C1                 POP     BC              ; Code string address
EC59 DA77E8             JP      C,INPBRK        ; Break pressed
EC5C 23                 INC     HL              ; Point to next DATA byte
EC5D 7E                 LD      A,(HL)          ; Get byte
EC5E B7                 OR      A               ; Is it zero (No input) ?
EC5F 2B                 DEC     HL              ; Back space INPUT pointer
EC60 C5                 PUSH    BC              ; Save code string address
EC61 CA6FEA             JP      Z,NXTDTA        ; Find end of buffer
EC64 D5                 PUSH    DE              ; Save variable address
EC65 3AAD10     ANTVLU: LD      A,(TYPE)        ; Check data type
EC68 B7                 OR      A               ; Is it numeric?
EC69 CA8FEC             JP      Z,INPBIN        ; Yes - Convert to binary
EC6C CD36E8             CALL    GETCHR          ; Get next character
EC6F 57                 LD      D,A             ; Save input character
EC70 47                 LD      B,A             ; Again
EC71 FE22               CP      '"'             ; Start of literal sting?
EC73 CA83EC             JP      Z,STRENT        ; Yes - Create string entry
EC76 3ACD10             LD      A,(READFG)      ; "READ" or "INPUT" ?
EC79 B7                 OR      A
EC7A 57                 LD      D,A             ; Save 00 if "INPUT"
EC7B CA80EC             JP      Z,ITMSEP        ; "INPUT" - End with 00
EC7E 163A               LD      D,":"           ; "DATA"  - End with 00 or ":"
EC80 062C       ITMSEP: LD      B,","           ; Item separator
EC82 2B                 DEC     HL              ; Back space for DTSTR
EC83 CDD2F1     STRENT: CALL    DTSTR           ; Get string terminated by D
EC86 EB                 EX      DE,HL           ; String address to DE
EC87 219AEC             LD      HL,LTSTND       ; Where to go after LETSTR
EC8A E3                 EX      (SP),HL         ; Save HL , get input pointer
EC8B D5                 PUSH    DE              ; Save address of string
EC8C C3A2EA             JP      LETSTR          ; Assign string to variable
```

```
EC8F CD36E8     INPBIN: CALL    GETCHR          ; Get next character
EC92 CD1AF9             CALL    ASCTFP          ; Convert ASCII to FP number
EC95 E3                 EX      (SP),HL         ; Save input ptr,Get var addr
EC96 CD6BF8             CALL    FPTHL           ; Move FPREG to variable
EC99 E1                 POP     HL              ; Restore input pointer
EC9A 2B         LTSTND: DEC     HL              ; DEC 'cos GETCHR INCs
EC9B CD36E8             CALL    GETCHR          ; Get next character
EC9E CAA6EC             JP      Z,MORDT         ; End of line - More needed?
ECA1 FE2C               CP      ","             ; Another value?
ECA3 C2ECEB             JP      NZ,BADINP       ; No - Bad input
ECA6 E3         MORDT:  EX      (SP),HL         ; Get code string address
ECA7 2B                 DEC     HL              ; DEC 'cos GETCHR INCs
ECA8 CD36E8             CALL    GETCHR          ; Get next character
ECAB C239EC             JP      NZ,NEDMOR       ; More needed - Get it
ECAE D1                 POP     DE              ; Restore DATA pointer
ECAF 3ACD10             LD      A,(READFG)      ; "READ" or "INPUT" ?
ECB2 B7                 OR      A
ECB3 EB                 EX      DE,HL           ; DATA pointer to HL
ECB4 C25CE8             JP      NZ,UPDATA       ; Update DATA pointer if "READ"
ECB7 D5                 PUSH    DE              ; Save code string address
ECB8 B6                 OR      (HL)            ; More input given?
ECB9 21C1EC             LD      HL,EXTIG        ; "?Extra ignored" message
ECBC C410F2             CALL    NZ,PRS          ; Output string if extra given
ECBF E1                 POP     HL              ; Restore code string address
ECC0 C9                 RET

ECC1 3F457874   EXTIG:  DEFB    "?Extra ignored",CR,LF,0

ECD2 CD70EA     FDTLP:  CALL    DATA            ; Get next statement
ECD5 B7                 OR      A               ; End of line?
ECD6 C2EBEC             JP      NZ,FANDT        ; No - See if DATA statement
ECD9 23                 INC     HL
ECDA 7E                 LD      A,(HL)          ; End of program?
ECDB 23                 INC     HL
ECDC B6                 OR      (HL)            ; 00 00 Ends program
ECDD 1E06               LD      E,0D            ; ?0D Error
ECDF CAC1E3             JP      Z,ERROR         ; Yes - Out of DATA
ECE2 23                 INC     HL
ECE3 5E                 LD      E,(HL)          ; LSB of line number
ECE4 23                 INC     HL
ECE5 56                 LD      D,(HL)          ; MSB of line number
ECE6 EB                 EX      DE,HL
ECE7 22C910             LD      (DATLIN),HL     ; Set line of current DATA item
ECEA EB                 EX      DE,HL
ECEB CD36E8     FANDT:  CALL    GETCHR          ; Get next character
ECEE FE83               CP      ZDATA           ; "DATA" token
ECF0 C2D2EC             JP      NZ,FDTLP        ; No "DATA" - Keep looking
ECF3 C365EC             JP      ANTVLU          ; Found - Convert input
```

```
ECF6 110000     NEXT:   LD      DE,0            ; In case no index given
ECF9 C42DEF     NEXT1:  CALL    NZ,GETVAR       ; Get index address
ECFC 22CE10             LD      (BRKLIN),HL     ; Save code string address
ECFF CD56E3             CALL    BAKSTK          ; Look for "FOR" block
ED02 C2B3E3             JP      NZ,NFERR        ; No "FOR" - ?NF Error
ED05 F9                 LD      SP,HL           ; Clear nested loops
ED06 D5                 PUSH    DE              ; Save index address
ED07 7E                 LD      A,(HL)          ; Get sign of STEP
ED08 23                 INC     HL
ED09 F5                 PUSH    AF              ; Save sign of STEP
ED0A D5                 PUSH    DE              ; Save index address
ED0B CD51F8             CALL    PHLTFP          ; Move index value to FPREG
ED0E E3                 EX      (SP),HL         ; Save address of TO value
ED0F E5                 PUSH    HL              ; Save address of index
ED10 CDBEF5             CALL    ADDPHL          ; Add STEP to index value
ED13 E1                 POP     HL              ; Restore address of index
ED14 CD6BF8             CALL    FPTHL           ; Move value to index variable
ED17 E1                 POP     HL              ; Restore address of TO value
ED18 CD62F8             CALL    LOADFP          ; Move TO value to BCDE
ED1B E5                 PUSH    HL              ; Save address of line of FOR
ED1C CD8EF8             CALL    CMPNUM          ; Compare index with TO value
ED1F E1                 POP     HL              ; Restore address of line num
ED20 C1                 POP     BC              ; Address of sign of STEP
ED21 90                 SUB     B               ; Compare with expected sign
ED22 CD62F8             CALL    LOADFP          ; BC = Loop stmt,DE = Line num
ED25 CA31ED             JP      Z,KILFOR        ; Loop finished - Terminate it
ED28 EB                 EX      DE,HL           ; Loop statement line number
ED29 225C10             LD      (LINEAT),HL     ; Set loop line number
ED2C 69                 LD      L,C             ; Set code string to loop
ED2D 60                 LD      H,B
ED2E C3EEE7             JP      PUTFID          ; Put back "FOR" and continue

ED31 F9         KILFOR: LD      SP,HL           ; Remove "FOR" block
ED32 2ACE10             LD      HL,(BRKLIN)     ; Code string after "NEXT"
ED35 7E                 LD      A,(HL)          ; Get next byte in code string
ED36 FE2C               CP      ","             ; More NEXTs ?
ED38 C2F2E7             JP      NZ,RUNCNT       ; No - Do next statement
ED3B CD36E8             CALL    GETCHR          ; Position to index name
ED3E CDF9EC             CALL    NEXT1           ; Re-enter NEXT routine
                        ; < will not RETurn to here , Exit to RUNCNT or Loop >
```

```
ED41 CD5AED     GETNUM: CALL    EVAL            ; Get a numeric expression
ED44 F6         TSTNUM: DEFB    (OR n)          ; Clear carry (numeric)
ED45 37         TSTSTR: SCF                     ; Set carry (string)
ED46 3AAD10     CHKTYP: LD      A,(TYPE)        ; Check types match
ED49 8F                 ADC     A,A             ; Expected + actual
ED4A B7                 OR      A               ; Clear carry , set parity
ED4B E8                 RET     PE              ; Even parity - Types match
ED4C C3BFE3             JP      TMERR           ; Different types - Error

                        ; <<< NO REFERENCE TO HERE >>>

ED4F CD90E6             CALL    CHKSYN          ; Make sure "=" follows
ED52 B4                 DEFB    ZEQUAL          ; "="
ED53 C35AED             JP      EVAL            ; Evaluate expression

ED56 CD90E6     OPNPAR: CALL    CHKSYN          ; Make sure "(" follows
ED59 28                 DEFB    "("
ED5A 2B         EVAL:   DEC     HL              ; Evaluate expression & save
ED5B 1600               LD      D,0             ; Precedence value
ED5D D5         EVAL1:  PUSH    DE              ; Save precedence
ED5E 0E01               LD      C,1
ED60 CD8AE3             CALL    CHKSTK          ; Check for 1 level of stack
ED63 CDD1ED             CALL    OPRND           ; Get next expression value
ED66 22D010     EVAL2:  LD      (NXTOPR),HL     ; Save address of next operator
ED69 2AD010     EVAL3:  LD      HL,(NXTOPR)     ; Restore address of next opr
ED6C C1                 POP     BC              ; Precedence value and operator
ED6D 78                 LD      A,B             ; Get precedence value
ED6E FE78               CP      78H             ; "AND" or "OR" ?
ED70 D444ED             CALL    NC,TSTNUM       ; No - Make sure it's a number
ED73 7E                 LD      A,(HL)          ; Get next operator / function
ED74 1600               LD      D,0             ; Clear Last relation
ED76 D6B3       RLTLP:  SUB     ZGTR            ; ">" Token
ED78 DA92ED             JP      C,FOPRND        ; + - * / ^ AND OR - Test it
ED7B FE03               CP      ZLTH+1-ZGTR     ; < = >
ED7D D292ED             JP      NC,FOPRND       ; Function - Call it
ED80 FE01               CP      ZEQUAL-ZGTR     ; "="
ED82 17                 RLA                     ; <-  Test for legal
ED83 AA                 XOR     D               ; <-  combinations of < = >
ED84 BA                 CP      D               ; <-  by combining last token
ED85 57                 LD      D,A             ; <-  with current one
ED86 DAADE3             JP      C,SNERR         ; Error if "<<" "==" or ">>"
ED89 22C510             LD      (CUROPR),HL     ; Save address of current token
ED8C CD36E8             CALL    GETCHR          ; Get next character
ED8F C376ED             JP      RLTLP           ; Treat the two as one
```

```
EDD1 AF          OPRND:  XOR     A               ; Get operand routine
EDD2 32AD10              LD      (TYPE),A        ; Set numeric expected
EDD5 CD36E8              CALL    GETCHR          ; Get next character
EDD8 1E24                LD      E,MO            ; ?MO Error
EDDA CAC1E3              JP      Z,ERROR         ; No operand - Error
EDDD DA1AF9              JP      C,ASCTFP        ; Number - Get value
EDE0 CD77E9              CALL    CHKLTR          ; See if a letter
EDE3 D222EE              JP      NC,CONVAR       ; Letter - Find variable
EDE6 FEAC                CP      ZPLUS           ; "+" Token ?
EDE9 CAD1ED              JP      Z,OPRND         ; Yes - Look for operand
EDEB FE2E                CP      "."             ; "." ?
EDED CA1AF9              JP      Z,ASCTFP        ; Yes - Create FP number
EDF0 FEAD                CP      ZMINUS          ; "-" Token ?
EDF2 CA11EE              JP      Z,MINUS         ; Yes - Do minus
EDF5 FE22                CP      '"'             ; Literal string ?
EDF7 CACFF1              JP      Z,QTSTR         ; Get string terminated by '"'
EDFA FEAA                CP      ZNOT            ; "NOT" Token ?
EDFC CA08EF              JP      Z,EVNOT         ; Yes - Eval NOT expression
EDFF FEA7                CP      ZFN             ; "FN" Token ?
EE01 CA33F1              JP      Z,DOFN          ; Yes - Do FN routine
EE04 D6B6                SUB     ZSGN            ; Is it a function?
EE06 D233EE              JP      NC,FNOFST       ; Yes - Evaluate function
EE09 CD56ED      EVLPAR: CALL    OPNPAR          ; Evaluate expression in "()"
EE0C CD90E6              CALL    CHKSYN          ; Make sure ")" follows
EE0F 29                  DEFB    ")"
EE10 C9                  RET

EE11 167D        MINUS:  LD      D,7DH           ; "-" precedence
EE13 CD5DED              CALL    EVAL1           ; Evaluate until prec' break
EE16 2AD010              LD      HL,(NXTOPR)     ; Get next operator address
EE19 E5                  PUSH    HL              ; Save next operator address
EE1A CD3CF8              CALL    INVSGN          ; Negate value
EE1D CD44ED              CALL    TSTNUM          ; Make sure it's a number
EE20 E1                  POP     HL              ; Restore next operator address
EE21 C9                  RET

EE22 CD2DEF      CONVAR: CALL    GETVAR          ; Get variable address to DE
EE25 E5                  PUSH    HL              ; Save code string address
EE26 EB                  EX      DE,HL           ; Variable address to HL
EE27 22E410              LD      (FPREG),HL      ; Save address of variable
EE2A 3AAD10              LD      A,(TYPE)        ; Get type
EE2D B7                  OR      A               ; Numeric?
EE2E CC51F8              CALL    Z,PHLTFP        ; Yes - Move contents to FPREG
EE31 E1                  POP     HL              ; Restore code string address
EE32 C9                  RET
```

```
ED92 7A          FOPRND: LD      A,D             ; < = > found ?
ED93 B7                  OR      A
ED94 C2A8EE              JP      NZ,TSTRED       ; Yes - Test for reduction
ED97 7E                  LD      A,(HL)          ; Get operator token
ED98 22C510              LD      (CUROPR),HL     ; Save operator address
ED9B D6AC                SUB     ZPLUS           ; Operator or function?
ED9D D8                  RET     C               ; Neither - Exit
ED9E FE07                CP      ZOR+1-ZPLUS     ; Is it + - * / ^ AND OR ?
EDA0 D0                  RET     NC              ; No - Exit
EDA1 5F                  LD      E,A             ; Coded operator
EDA2 3AAD10              LD      A,(TYPE)        ; Get data type
EDA5 3D                  DEC     A               ; FF = numeric , 00 = string
EDA6 B3                  OR      E               ; Combine with coded operator
EDA7 7B                  LD      A,E             ; Get coded operator
EDA8 CA06F3              JP      Z,CONCAT        ; String concatenation
EDAB 07                  RLCA                    ; Times 2
EDAC 83                  ADD     A,E             ; Times 3
EDAD 5F                  LD      E,A             ; To DE (D is 0)
EDAE 21A4E2              LD      HL,PRITAB       ; Precedence table
EDB1 19                  ADD     HL,DE           ; To the operator concerned
EDB2 78                  LD      A,B             ; Last operator precedence
EDB3 56                  LD      D,(HL)          ; Get evaluation precedence
EDB4 BA                  CP      D               ; Compare with eval precedence
EDB5 D0                  RET     NC              ; Exit if higher precedence
EDB6 23                  INC     HL              ; Point to routine address
EDB7 CD44ED              CALL    TSTNUM          ; Make sure it's a number

EDBA C5          STKTHS: PUSH    BC              ; Save last precedence & token
EDBB 0169ED              LD      BC,EVAL3        ; Where to go on prec' break
EDBE C5                  PUSH    BC              ; Save on stack for return
EDBF 43                  LD      B,E             ; Save operator
EDC0 4A                  LD      C,D             ; Save precedence
EDC1 CD44F8              CALL    STAKFP          ; Move value to stack
EDC4 5B                  LD      E,B             ; Restore operator
EDC5 51                  LD      D,C             ; Restore precedence
EDC6 4E                  LD      C,(HL)          ; Get LSB of routine address
EDC7 23                  INC     HL
EDC8 46                  LD      B,(HL)          ; Get MSB of routine address
EDC9 23                  INC     HL
EDCA C5                  PUSH    BC              ; Save routine address
EDCB 2AC510              LD      HL,(CUROPR)     ; Address of current operator
EDCE C35DED              JP      EVAL1           ; Loop until prec' break
```

```
EE33 0600       FNOFST: LD      B,0             ; Get address of function
EE35 07                 RLCA                    ; Double function offset
EE36 4F                 LD      C,A             ; BC = Offset in function table
EE37 C5                 PUSH    BC              ; Save adjusted token value
EE38 CD36E8             CALL    GETCHR          ; Get next character
EE3B 79                 LD      A,C             ; Get adjusted token value
EE3C FE22               CP      2*(ZPOINT-ZSGN) ; Adjusted "POINT" token?
EE3E CA79FF             JP      Z,POINTB        ; Yes - Do "POINT" (not POINTB)
EE41 FE2D               CP      2*(ZLEFT-ZSGN)-1; Adj' LEFT$,RIGHT$ or MID$ ?
EE43 DA5FEE             JP      C,FNVAL         ; No - Do function
EE46 CD56ED             CALL    OPNPAR          ; Evaluate expression  (X,...
EE49 CD90E6             CALL    CHKSYN          ; Make sure "," follows
EE4C 2C                 DEFB    ","
EE4D CD45ED             CALL    TSTSTR          ; Make sure it's a string
EE50 EB                 EX      DE,HL           ; Save code string address
EE51 2AE410             LD      HL,(FPREG)      ; Get address of string
EE54 E3                 EX      (SP),HL         ; Save address of string
EE55 E5                 PUSH    HL              ; Save adjusted token value
EE56 EB                 EX      DE,HL           ; Restore code string address
EE57 CD84F4             CALL    GETINT          ; Get integer 0-255
EE5A EB                 EX      DE,HL           ; Save code string address
EE5B E3                 EX      (SP),HL         ; Save integer,HL = adj' token
EE5C C367EE             JP      GOFUNC          ; Jump to string function

EE5F CD09EE     FNVAL:  CALL    EVLPAR          ; Evaluate expression
EE62 E3                 EX      (SP),HL         ; HL = Adjusted token value
EE63 111DEE             LD      DE,RETNUM       ; Return number from function
EE66 D5                 PUSH    DE              ; Save on stack
EE67 010FE1     GOFUNC: LD      BC,FNCTAB       ; Function routine addresses
EE6A 09                 ADD     HL,BC           ; Point to right address
EE6B 4E                 LD      C,(HL)          ; Get LSB of address
EE6C 23                 INC     HL
EE6D 66                 LD      H,(HL)          ; Get MSB of address
EE6E 69                 LD      L,C             ; Address to HL
EE6F E9                 JP      (HL)            ; Jump to function

EE70 15         SGNEXP: DEC     D               ; Dec to flag negative exponent
EE71 FEAD               CP      ZMINUS          ; "-" token ?
EE73 C8                 RET     Z               ; Yes - Return
EE74 FE2D               CP      "-"             ; "-" ASCII ?
EE76 C8                 RET     Z               ; Yes - Return
EE77 14                 INC     D               ; Inc to flag positive exponent
EE78 FE2B               CP      "+"             ; "+" ASCII ?
EE7A C8                 RET     Z               ; Yes - Return
EE7B FEAC               CP      ZPLUS           ; "+" token ?
EE7D C8                 RET     Z               ; Yes - Return
EE7E 2B                 DEC     HL              ; DEC 'cos GETCHR INCs
EE7F C9                 RET                     ; Return "NZ"
```

```
EE80 F6          POR:    DEFB    (OR n)          ; Flag "OR"
EE81 AF          PAND:   XOR     A               ; Flag "AND"
EE82 F5                  PUSH    AF              ; Save "AND" / "OR" flag
EE83 CD44ED              CALL    TSTNUM          ; Make sure it's a number
EE86 CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
EE89 F1                  POP     AF              ; Restore "AND" / "OR" flag
EE8A EB                  EX      DE,HL           ; <- Get last
EE8B C1                  POP     BC              ; <-  value
EE8C E3                  EX      (SP),HL         ; <-  from
EE8D EB                  EX      DE,HL           ; <-  stack
EE8E CD54F8              CALL    FPBCDE          ; Move last value to FPREG
EE91 F5                  PUSH    AF              ; Save "AND" / "OR" flag
EE92 CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
EE95 F1                  POP     AF              ; Restore "AND" / "OR" flag
EE96 C1                  POP     BC              ; Get value
EE97 79                  LD      A,C             ; Get LSB
EE98 21F1F0              LD      HL,ACPASS       ; Address of save AC as current
EE9B C2A3EE              JP      NZ,POR1         ; Jump if OR
EE9E A3                  AND     E               ; "AND" LSBs
EE9F 4F                  LD      C,A             ; Save LSB
EEA0 78                  LD      A,B             ; Get MBS
EEA1 A2                  AND     D               ; "AND" MSBs
EEA2 E9                  JP      (HL)            ; Save AC as current (ACPASS)

EEA3 B3          POR1:   OR      E               ; "OR" LSBs
EEA4 4F                  LD      C,A             ; Save LSB
EEA5 78                  LD      A,B             ; Get MSB
EEA6 B2                  OR      D               ; "OR" MSBs
EEA7 E9                  JP      (HL)            ; Save AC as current (ACPASS)

EEA8 21BAEE      TSTRED: LD      HL,CMPLOG       ; Logical compare routine
EEAB 3AAD10              LD      A,(TYPE)        ; Get data type
EEAE 1F                  RRA                     ; Carry set = string
EEAF 7A                  LD      A,D             ; Get last precedence value
EEB0 17                  RLA                     ; Times 2 plus carry
EEB1 5F                  LD      E,A             ; To E
EEB2 1664                LD      D,64H           ; Relational precedence
EEB4 78                  LD      A,B             ; Get current precedence
EEB5 BA                  CP      D               ; Compare with last
EEB6 D0                  RET     NC              ; Eval if last was rel' or log'
EEB7 C3BAED              JP      STKTHS          ; Stack this one and get next
```

```
EEBA BCEE       CMPLOG: DEFW    CMPLG1          ; Compare two values / strings
EEBC 79         CMPLG1: LD      A,C             ; Get data type
EEBD B7                 OR      A
EEBE 1F                 RRA
EEBF C1                 POP     BC              ; Get last expression to BCDE
EEC0 D1                 POP     DE
EEC1 F5                 PUSH    AF              ; Save status
EEC2 CD46ED             CALL    CHKTYP          ; Check that types match
EEC5 21FEEE             LD      HL,CMPRES       ; Result to comparison
EEC8 E5                 PUSH    HL              ; Save for RETurn
EEC9 CA8EF8             JP      Z,CMPNUM        ; Compare values if numeric
EECC AF                 XOR     A               ; Compare two strings
EECD 32AD10             LD      (TYPE),A        ; Set type to numeric
EED0 D5                 PUSH    DE              ; Save string name
EED1 CD53F3             CALL    GSTRCU          ; Get current string
EED4 7E                 LD      A,(HL)          ; Get length of string
EED5 23                 INC     HL
EED6 23                 INC     HL
EED7 4E                 LD      C,(HL)          ; Get LSB of address
EED8 23                 INC     HL
EED9 46                 LD      B,(HL)          ; Get MSB of address
EEDA D1                 POP     DE              ; Restore string name
EEDB C5                 PUSH    BC              ; Save address of string
EEDC F5                 PUSH    AF              ; Save length of string
EEDD CD57F3             CALL    GSTRDE          ; Get second string
EEE0 CD62F8             CALL    LOADFP          ; Get address of second string
EEE3 F1                 POP     AF              ; Restore length of string 1
EEE4 57                 LD      D,A             ; Length to D
EEE5 E1                 POP     HL              ; Restore address of string 1
EEE6 7B         CMPSTR: LD      A,E             ; Bytes of string 2 to do
EEE7 B2                 OR      D               ; Bytes of string 1 to do
EEE8 C8                 RET     Z               ; Exit if all bytes compared
EEE9 7A                 LD      A,D             ; Get bytes of string 1 to do
EEEA D601               SUB     1
EEEC D8                 RET     C               ; Exit if end of string 1
EEED AF                 XOR     A
EEEE BB                 CP      E               ; Bytes of string 2 to do
EEEF 3C                 INC     A
EEF0 D0                 RET     NC              ; Exit if end of string 2
EEF1 15                 DEC     D               ; Count bytes in string 1
EEF2 1D                 DEC     E               ; Count bytes in string 2
EEF3 0A                 LD      A,(BC)          ; Byte in string 2
EEF4 BE                 CP      (HL)            ; Compare to byte in string 1
EEF5 23                 INC     HL              ; Move up string 1
EEF6 03                 INC     BC              ; Move up string 2
EEF7 CAE6EE             JP      Z,CMPSTR        ; Same - Try next bytes
EEFA 3F                 CCF                     ; Flag difference (">" or "<")
EEFB C31EF8             JP      FLGDIF          ; "<" gives -1 , ">" gives +1

EEFE 3C         CMPRES: INC     A               ; Increment current value
EEFF 8F                 ADC     A,A             ; Double plus carry
EF00 C1                 POP     BC              ; Get other value
EF01 A0                 AND     B               ; Combine them
EF02 C6FF               ADD     A,-1            ; Carry set if different
EF04 9F                 SBC     A,A             ; 00 - Equal , FF - Different
EF05 C325F8             JP      FLGREL          ; Set current value & continue
```

```
EF08 165A       EVNOT:  LD      D,5AH           ; Precedence value for "NOT"
EF0A CD5DED             CALL    EVAL1           ; Eval until precedence break
EF0D CD44ED             CALL    TSTNUM          ; Make sure it's a number
EF10 CD8BE9             CALL    DEINT           ; Get integer -32768 - 32767
EF13 7B                 LD      A,E             ; Get LSB
EF14 2F                 CPL                     ; Invert LSB
EF15 4F                 LD      C,A             ; Save "NOT" of LSB
EF16 7A                 LD      A,D             ; Get MSB
EF17 2F                 CPL                     ; Invert MSB
EF18 CDF1F0             CALL    ACPASS          ; Save AC as current
EF1B C1                 POP     BC              ; Clean up stack
EF1C C369ED             JP      EVAL3           ; Continue evaluation

EF1F 2B         DIMRET: DEC     HL              ; DEC 'cos GETCHR INCs
EF20 CD36E8             CALL    GETCHR          ; Get next character
EF23 C8                 RET     Z               ; End of DIM statement
EF24 CD90E6             CALL    CHKSYN          ; Make sure "," follows
EF27 2C                 DEFB    ","
EF28 011FEF     DIM:    LD      BC,DIMRET       ; Return to "DIMRET"
EF2B C5                 PUSH    BC              ; Save on stack
EF2C F6                 DEFB    (OR n)          ; Flag "Create" variable
EF2D AF         GETVAR: XOR     A               ; Find variable address,to DE
EF2E 32AC10             LD      (LCRFLG),A      ; Set locate / create flag
EF31 46                 LD      B,(HL)          ; Get First byte of name
EF32 CD77E9     GTFNAM: CALL    CHKLTR          ; See if a letter
EF35 DAADE3             JP      C,SNERR         ; ?SN Error if not a letter
EF38 AF                 XOR     A
EF39 4F                 LD      C,A             ; Clear second byte of name
EF3A 32AD10             LD      (TYPE),A        ; Set type to numeric
EF3D CD36E8             CALL    GETCHR          ; Get next character
EF40 DA49EF             JP      C,SVNAM2        ; Numeric - Save in name
EF43 CD77E9             CALL    CHKLTR          ; See if a letter
EF46 DA56EF             JP      C,CHARTY        ; Not a letter - Check type
EF49 4F         SVNAM2: LD      C,A             ; Save second byte of name
EF4A CD36E8     ENDNAM: CALL    GETCHR          ; Get next character
EF4D DA4AEF             JP      C,ENDNAM        ; Numeric - Get another
EF50 CD77E9             CALL    CHKLTR          ; See if a letter
EF53 D24AEF             JP      NC,ENDNAM       ; Letter - Get another
EF56 D624       CHARTY: SUB     "$"             ; String variable?
EF58 C265EF             JP      NZ,NOTSTR       ; No - Numeric variable
EF5B 3C                 INC     A               ; A = 1 (string type)
EF5C 32AD10             LD      (TYPE),A        ; Set type to string
EF5F 0F                 RRCA                    ; A = 80H , Flag for string
EF60 81                 ADD     A,C             ; 2nd byte of name has bit 7 on
EF61 4F                 LD      C,A             ; Resave second byte on name
EF62 CD36E8             CALL    GETCHR          ; Get next character
EF65 3ACB10     NOTSTR: LD      A,(FORFLG)      ; Array name needed ?
EF68 3D                 DEC     A
EF69 CA12F0             JP      Z,ARLDSV        ; Yes - Get array name
EF6C F275EF             JP      P,NSCFOR        ; No array with "FOR" or "FN"
EF6F 7E                 LD      A,(HL)          ; Get byte again
EF70 D628               SUB     "("             ; Subscripted variable?
EF72 CAEAEF             JP      Z,SBSCPT        ; Yes - Sort out subscript
```

```
EF75 AF          NSCFOR: XOR     A               ; Simple variable
EF76 32CB10              LD      (FORFLG),A      ; Clear "FOR" flag
EF79 E5                  PUSH    HL              ; Save code string address
EF7A 50                  LD      D,B             ; DE = Variable name to find
EF7B 59                  LD      E,C
EF7C 2ADE10              LD      HL,(FNRGNM)     ; FN argument name
EF7F CD8AE6              CALL    CPDEHL          ; Is it the FN argument?
EF82 11E010              LD      DE,FNARG        ; Point to argument value
EF85 CA54F7              JP      Z,POPHRT        ; Yes - Return FN argument value
EF88 2AD810              LD      HL,(VAREND)     ; End of variables
EF8B EB                  EX      DE,HL           ; Address of end of search
EF8C 2AD610              LD      HL,(PROGND)     ; Start of variables address
EF8F CD8AE6      FNDVAR: CALL    CPDEHL          ; End of variable list table?
EF92 CAA8EF              JP      Z,CFEVAL        ; Yes - Called from EVAL?
EF95 79                  LD      A,C             ; Get second byte of name
EF96 96                  SUB     (HL)            ; Compare with name in list
EF97 23                  INC     HL              ; Move on to first byte
EF98 C29DEF              JP      NZ,FNTHR        ; Different - Find another
EF9B 78                  LD      A,B             ; Get first byte of name
EF9C 96                  SUB     (HL)            ; Compare with name in list
EF9D 23          FNTHR:  INC     HL              ; Move on to LSB of value
EF9E CADCEF              JP      Z,RETADR        ; Found - Return address
EFA1 23                  INC     HL              ; <- Skip
EFA2 23                  INC     HL              ; <- over
EFA3 23                  INC     HL              ; <- F.P.
EFA4 23                  INC     HL              ; <- value
EFA5 C38FEF              JP      FNDVAR          ; Keep looking
EFA8 E1          CFEVAL: POP     HL              ; Restore code string address
EFA9 E3                  EX      (SP),HL         ; Get return address
EFAA D5                  PUSH    DE              ; Save address of variable
EFAB 1125EE              LD      DE,FRMEVL       ; Return address in EVAL
EFAE CD8AE6              CALL    CPDEHL          ; Called from EVAL ?
EFB1 D1                  POP     DE              ; Restore address of variable
EFB2 CADFEF              JP      Z,RETNUL        ; Yes - Return null variable
EFB5 E3                  EX      (SP),HL         ; Put back return
EFB6 E5                  PUSH    HL              ; Save code string address
EFB7 C5                  PUSH    BC              ; Save variable name
EFB8 010600              LD      BC,6            ; 2 byte name plus 4 byte data
EFBB 2ADA10              LD      HL,(ARREND)     ; End of arrays
EFBE E5                  PUSH    HL              ; Save end of arrays
EFBF 09                  ADD     HL,BC           ; Move up 6 bytes
EFC0 C1                  POP     BC              ; Source address in BC
EFC1 E5                  PUSH    HL              ; Save new end address
EFC2 CD79E3              CALL    MOVUP           ; Move arrays up
EFC5 E1                  POP     HL              ; Restore new end address
EFC6 22DA10              LD      (ARREND),HL     ; Set new end address
EFC9 60                  LD      H,B             ; End of variables to HL
EFCA 69                  LD      L,C
EFCB 22D810              LD      (VAREND),HL     ; Set new end address
```

```
EFCE 2B          ZEROLP: DEC     HL              ; Back through to zero variable
EFCF 3600                LD      (HL),0          ; Zero byte in variable
EFD1 CD8AE6              CALL    CPDEHL          ; Done them all?
EFD4 C2CEEF              JP      NZ,ZEROLP       ; No - Keep on going
EFD7 D1                  POP     DE              ; Get variable name
EFD8 73                  LD      (HL),E          ; Store second character
EFD9 23                  INC     HL
EFDA 72                  LD      (HL),D          ; Store first character
EFDB 23                  INC     HL
EFDC EB          RETADR: EX      DE,HL           ; Address of variable in DE
EFDD E1                  POP     HL              ; Restore code string address
EFDE C9                  RET

EFDF 32E710      RETNUL: LD      (FPEXP),A       ; Set result to zero
EFE2 214AE3              LD      HL,ZERBYT       ; Also set a null string
EFE5 22E410              LD      (FPREG),HL      ; Save for EVAL
EFE8 E1                  POP     HL              ; Restore code string address
EFE9 C9                  RET

EFEA E5          SBSCPT: PUSH    HL              ; Save code string address
EFEB 2AAC10              LD      HL,(LCRFLG)     ; Locate/Create and Type
EFEE E3                  EX      (SP),HL         ; Save and get code string
EFEF 57                  LD      D,A             ; Zero number of dimensions
EFF0 D5          SCPTLP: PUSH    DE              ; Save number of dimensions
EFF1 C5                  PUSH    BC              ; Save array name
EFF2 CD7FE9              CALL    FPSINT          ; Get subscript (0-32767)
EFF5 C1                  POP     BC              ; Restore array name
EFF6 F1                  POP     AF              ; Get number of dimensions
EFF7 EB                  EX      DE,HL
EFF8 E3                  EX      (SP),HL         ; Save subscript value
EFF9 E5                  PUSH    HL              ; Save LCRFLG and TYPE
EFFA EB                  EX      DE,HL
EFFB 3C                  INC     A               ; Count dimensions
EFFC 57                  LD      D,A             ; Save in D
EFFD 7E                  LD      A,(HL)          ; Get next byte in code string
EFFE FE2C                CP      ","             ; Comma (more to come)?
F000 CAF0EF              JP      Z,SCPTLP        ; Yes - More subscripts
F003 CD90E6              CALL    CHKSYN          ; Make sure ")" follows
F006 29                  DEFB    ")"
F007 22D010              LD      (NXTOPR),HL     ; Save code string address
F00A E1                  POP     HL              ; Get LCRFLG and TYPE
F00B 22AC10              LD      (LCRFLG),HL     ; Restore Locate/create & type
F00E 1E00                LD      E,0             ; Flag not CSAVE* or CLOAD*
F010 D5                  PUSH    DE              ; Save number of dimensions (D)
F011 11                  DEFB    (LD DE,nn)      ; Skip "PUSH HL" and "PUSH AF"
```

# NASCOM

# ROM

# BASIC

# DIS-ASSEMBLED

# PART 5

BY CARL LLOYD-PARKER

```
F012 E5          ARLDSV: PUSH    HL              ; Save code string address
F013 F5                  PUSH    AF              ; A = 00 , Flags set = Z,N
F014 2AD810              LD      HL,(VAREND)     ; Start of arrays
F017 3E                  DEFB    (LD A,n)        ; Skip "ADD HL,DE"
F018 19          FNDARY: ADD     HL,DE           ; Move to next array start
F019 EB                  EX      DE,HL
F01A 2ADA10              LD      HL,(ARREND)     ; End of arrays
F01D EB                  EX      DE,HL           ; Current array pointer
F01E CD8AE6              CALL    CPDEHL          ; End of arrays found?
F021 CA4AF0              JP      Z,CREARY        ; Yes - Create array
F024 7E                  LD      A,(HL)          ; Get second byte of name
F025 B9                  CP      C               ; Compare with name given
F026 23                  INC     HL              ; Move on
F027 C22CF0              JP      NZ,NXTARY       ; Different - Find next array
F02A 7E                  LD      A,(HL)          ; Get first byte of name
F02B B8                  CP      B               ; Compare with name given
F02C 23          NXTARY: INC     HL              ; Move on
F02D 5E                  LD      E,(HL)          ; Get LSB of next array address
F02E 23                  INC     HL
F02F 56                  LD      D,(HL)          ; Get MSB of next array address
F030 23                  INC     HL
F031 C218F0              JP      NZ,FNDARY       ; Not found - Keep looking
F034 3AAC10              LD      A,(LCRFLG)      ; Found - Locate or Create it?
F037 B7                  OR      A
F038 C2B6E3              JP      NZ,DDERR        ; Create - ?DD Error
F03B F1                  POP     AF              ; Locate - Get number of dim'ns
F03C 44                  LD      B,H             ; BC Points to array dim'ns
F03D 4D                  LD      C,L
F03E CA54F7              JP      Z,POPHRT        ; Jump if array load/save
F041 96                  SUB     (HL)            ; Same number of dimensions?
F042 CAA8F0              JP      Z,FINDEL        ; Yes - Find element
F045 1E10        BSERR:  LD      E,BS            ; ?BS Error
F047 C3C1E3              JP      ERROR           ; Output error
```

```
F04A 110400     CREARY: LD      DE,4            ; 4 Bytes per entry
F04D F1                 POP     AF              ; Array to save or 0 dim'ns?
F04E CAA0E9             JP      Z,FCERR         ; Yes - ?FC Error
F051 71                 LD      (HL),C          ; Save second byte of name
F052 23                 INC     HL
F053 70                 LD      (HL),B          ; Save first byte of name
F054 23                 INC     HL
F055 4F                 LD      C,A             ; Number of dimensions to C
F056 CD8AE3             CALL    CHKSTK          ; Check if enough memory
F059 23                 INC     HL              ; Point to number of dimensions
F05A 23                 INC     HL
F05B 22C510             LD      (CUROPR),HL     ; Save address of pointer
F05E 71                 LD      (HL),C          ; Set number of dimensions
F05F 23                 INC     HL
F060 3AAC10             LD      A,(LCRFLG)      ; Locate of Create?
F063 17                 RLA                     ; Carry set = Create
F064 79                 LD      A,C             ; Get number of dimensions
F065 010B00     CRARLP: LD      BC,10+1         ; Default dimension size 10
F068 D26DF0             JP      NC,DEFSIZ       ; Locate - Set default size
F06B C1                 POP     BC              ; Get specified dimension size
F06C 03                 INC     BC              ; Include zero element
F06D 71         DEFSIZ: LD      (HL),C          ; Save LSB of dimension size
F06E 23                 INC     HL
F06F 70                 LD      (HL),B          ; Save MSB of dimension size
F070 23                 INC     HL
F071 F5                 PUSH    AF              ; Save num' of dim'ns an status
F072 E5                 PUSH    HL              ; Save address of dim'n size
F073 CDFFF8             CALL    MLDEBC          ; Multiply DE by BC to find
F076 EB                 EX      DE,HL           ; amount of mem needed (to DE)
F077 E1                 POP     HL              ; Restore address of dimension
F078 F1                 POP     AF              ; Restore number of dimensions
F079 3D                 DEC     A               ; Count them
F07A C265F0             JP      NZ,CRARLP       ; Do next dimension if more
F07D F5                 PUSH    AF              ; Save locate/create flag
F07E 42                 LD      B,D             ; MSB of memory needed
F07F 4B                 LD      C,E             ; LSB of memory needed
F080 EB                 EX      DE,HL
F081 19                 ADD     HL,DE           ; Add bytes to array start
F082 DAA2E3             JP      C,OMERR         ; Too big - Error
F085 CD93E3             CALL    ENFMEM          ; See if enough memory
F088 22DA10             LD      (ARREND),HL     ; Save new end of array
```

```
F08B 2B          ZERARY: DEC     HL              ; Back through array data
F08C 3600                LD      (HL),0          ; Set array element to zero
F08E CD8AE6              CALL    CPDEHL          ; All elements zeroed?
F091 C28BF0              JP      NZ,ZERARY       ; No - Keep on going
F094 03                  INC     BC              ; Number of bytes + 1
F095 57                  LD      D,A             ; A=0
F096 2AC510              LD      HL,(CUROPR)     ; Get address of array
F099 5E                  LD      E,(HL)          ; Number of dimensions
F09A EB                  EX      DE,HL           ; To HL
F09B 29                  ADD     HL,HL           ; Two bytes per dimension size
F09C 09                  ADD     HL,BC           ; Add number of bytes
F09D EB                  EX      DE,HL           ; Bytes needed to DE
F09E 2B                  DEC     HL
F09F 2B                  DEC     HL
F0A0 73                  LD      (HL),E          ; Save LSB of bytes needed
F0A1 23                  INC     HL
F0A2 72                  LD      (HL),D          ; Save MSB of bytes needed
F0A3 23                  INC     HL
F0A4 F1                  POP     AF              ; Locate / Create?
F0A5 DACCF0              JP      C,ENDDIM        ; A is 0 , End if create
F0A8 47          FINDEL: LD      B,A             ; Find array element
F0A9 4F                  LD      C,A
F0AA 7E                  LD      A,(HL)          ; Number of dimensions
F0AB 23                  INC     HL
F0AC 16                  DEFB    (LD D,n)        ; Skip "POP HL"
F0AD E1          FNDELP: POP     HL              ; Address of next dim' size
F0AE 5E                  LD      E,(HL)          ; Get LSB of dim'n size
F0AF 23                  INC     HL
F0B0 56                  LD      D,(HL)          ; Get MSB of dim'n size
F0B1 23                  INC     HL
F0B2 E3                  EX      (SP),HL         ; Save address - Get index
F0B3 F5                  PUSH    AF              ; Save number of dim'ns
F0B4 CD8AE6              CALL    CPDEHL          ; Dimension too large?
F0B7 D245F0              JP      NC,BSERR        ; Yes - ?BS Error
F0BA E5                  PUSH    HL              ; Save index
F0BB CDFFF8              CALL    MLDEBC          ; Multiply previous by size
F0BE D1                  POP     DE              ; Index supplied to DE
F0BF 19                  ADD     HL,DE           ; Add index to pointer
F0C0 F1                  POP     AF              ; Number of dimensions
F0C1 3D                  DEC     A               ; Count them
F0C2 44                  LD      B,H             ; MSB of pointer
F0C3 4D                  LD      C,L             ; LSB of pointer
F0C4 C2ADF0              JP      NZ,FNDELP       ; More - Keep going
F0C7 29                  ADD     HL,HL           ; 4 Bytes per element
F0C8 29                  ADD     HL,HL
F0C9 C1                  POP     BC              ; Start of array
F0CA 09                  ADD     HL,BC           ; Point to element
F0CB EB                  EX      DE,HL           ; Address of element to DE
F0CC 2AD010      ENDDIM: LD      HL,(NXTOPR)     ; Get code string address
F0CF C9                  RET
```

```
FOD0 2ADA10     FRE:    LD      HL,(ARREND)     ; Start of free memory
FOD3 EB                 EX      DE,HL           ; To DE
FOD4 210000             LD      HL,0            ; End of free memory
FOD7 39                 ADD     HL,SP           ; Current stack value
FOD8 3AAD10             LD      A,(TYPE)        ; Dummy argument type
FODB B7                 OR      A
FODC CAECFO             JP      Z,FRENUM        ; Numeric - Free variable space
FODF CD53F3             CALL    GSTRCU          ; Current string to pool
FOE2 CD53F2             CALL    GARBGE          ; Garbage collection
FOE5 2A5A10             LD      HL,(STRSPC)     ; Bottom of string space in use
FOE8 EB                 EX      DE,HL           ; To DE
FOE9 2AC310             LD      HL,(STRBOT)     ; Bottom of string space
FOEC 7D         FRENUM: LD      A,L             ; Get LSB of end
FOED 93                 SUB     E               ; Subtract LSB of beginning
FOEE 4F                 LD      C,A             ; Save difference if C
FOEF 7C                 LD      A,H             ; Get MSB of end
FOF0 9A                 SBC     A,D             ; Subtract MSB of beginning
FOF1 41         ACPASS: LD      B,C             ; Return integer AC
FOF2 50         ABPASS: LD      D,B             ; Return integer AB
FOF3 1E00               LD      E,0
FOF5 21AD10             LD      HL,TYPE         ; Point to type
FOF8 73                 LD      (HL),E          ; Set type to numeric
FOF9 0690               LD      B,80H+16        ; 16 bit integer
FOFB C32AF8             JP      RETINT          ; Return the integr

FOFE 3AAB10     POS:    LD      A,(CURPOS)      ; Get cursor position
F101 47         PASSA:  LD      B,A             ; Put A into AB
F102 AF                 XOR     A               ; Zero A
F103 C3F2F0             JP      ABPASS          ; Return integer AB

F106 CD89F1     DEF:    CALL    CHEKFN          ; Get "FN" and name
F109 CD7BF1             CALL    IDTEST          ; Test for illegal direct
F10C 0170EA             LD      BC,DATA         ; To get next statement
F10F C5                 PUSH    BC              ; Save address for RETurn
F110 D5                 PUSH    DE              ; Save address of function ptr
F111 CD90E6             CALL    CHKSYN          ; Make sure "(" follows
F114 28                 DEFB    "("
F115 CD2DEF             CALL    GETVAR          ; Get argument variable name
F118 E5                 PUSH    HL              ; Save code string address
F119 EB                 EX      DE,HL           ; Argument address to HL
F11A 2B                 DEC     HL
F11B 56                 LD      D,(HL)          ; Get first byte of arg name
F11C 2B                 DEC     HL
F11D 5E                 LD      E,(HL)          ; Get second byte of arg name
F11E E1                 POP     HL              ; Restore code string address
F11F CD44ED             CALL    TSTNUM          ; Make sure numeric argument
F122 CD90E6             CALL    CHKSYN          ; Make sure ")" follows
F125 29                 DEFB    ")"
F126 CD90E6             CALL    CHKSYN          ; Make sure "=" follows
F129 B4                 DEFB    ZEQUAL          ; "=" token
F12A 44                 LD      B,H             ; Code string address to BC
F12B 4D                 LD      C,L
F12C E3                 EX      (SP),HL         ; Save code str , Get FN ptr
F12D 71                 LD      (HL),C          ; Save LSB of FN code string
F12E 23                 INC     HL
F12F 70                 LD      (HL),B          ; Save MSB of FN code string
```

```
F133 CD89F1     DOFN:   CALL    CHEKFN          ; Make sure FN follows
F136 D5                 PUSH    DE              ; Save function pointer address
F137 CD09EE             CALL    EVLPAR          ; Evaluate expression in "()"
F13A CD44ED             CALL    TSTNUM          ; Make sure numeric result
F13D E3                 EX      (SP),HL         ; Save code str , Get FN ptr
F13E 5E                 LD      E,(HL)          ; Get LSB of FN code string
F13F 23                 INC     HL
F140 56                 LD      D,(HL)          ; Get MSB of FN code string
F141 23                 INC     HL
F142 7A                 LD      A,D             ; And function DEFined?
F143 B3                 OR      E
F144 CAB9E3             JP      Z,UFERR         ; No - ?UF Error
F147 7E                 LD      A,(HL)          ; Get LSB of argument address
F148 23                 INC     HL
F149 66                 LD      H,(HL)          ; Get MSB of argument address
F14A 6F                 LD      L,A             ; HL = Arg variable address
F14B E5                 PUSH    HL              ; Save it
F14C 2ADE10             LD      HL,(FNRGNM)     ; Get old argument name
F14F E3                 EX      (SP),HL         ; Save old , Get new
F150 22DE10             LD      (FNRGNM),HL     ; Set new argument name
F153 2AE210             LD      HL,(FNARG+2)    ; Get LSB,NLSB of old arg value
F156 E5                 PUSH    HL              ; Save it
F157 2AE010             LD      HL,(FNARG)      ; Get MSB,EXP of old arg value
F15A E5                 PUSH    HL              ; Save it
F15B 21E010             LD      HL,FNARG        ; HL = Value of argument
F15E D5                 PUSH    DE              ; Save FN code string address
F15F CD6BF8             CALL    FPTHL           ; Move FPREG to argument
F162 E1                 POP     HL              ; Get FN code string address
F163 CD41ED             CALL    GETNUM          ; Get value from function
F166 2B                 DEC     HL              ; DEC 'cos GETCHR INCs
F167 CD36E8             CALL    GETCHR          ; Get next character
F16A C2ADE3             JP      NZ,SNERR        ; Bad character in FN - Error
F16D E1                 POP     HL              ; Get MSB,EXP of old arg
F16E 22E010             LD      (FNARG),HL      ; Restore it
F171 E1                 POP     HL              ; Get LSB,NLSB of old arg
F172 22E210             LD      (FNARG+2),HL    ; Restore it
F175 E1                 POP     HL              ; Get name of old arg
F176 22DE10             LD      (FNRGNM),HL     ; Restore it
F179 E1                 POP     HL              ; Restore code string address
F17A C9                 RET

F17B E5         IDTEST: PUSH    HL              ; Save code string address
F17C 2A5C10             LD      HL,(LINEAT)     ; Get current line number
F17F 23                 INC     HL              ; -1 means direct statement
F180 7C                 LD      A,H
F181 B5                 OR      L
F182 E1                 POP     HL              ; Restore code string address
F183 C0                 RET     NZ              ; Return if in program
F184 1E16               LD      E,ID            ; ?ID Error
F186 C3C1E3             JP      ERROR
```

```
F189 CD90E6     CHEKFN: CALL    CHKSYN          ; Make sure FN follows
F18C A7                 DEFB    ZFN             ; "FN" token
F18D 3E80               LD      A,80H
F18F 32CB10             LD      (FORFLG),A      ; Flag FN name to find
F192 B6                 OR      (HL)            ; FN name has bit 7 set
F193 47                 LD      B,A             ; in first byte of name
F194 CD32EF             CALL    GTFNAM          ; Get FN name
F197 C344ED             JP      TSTNUM          ; Make sure numeric function

F19A CD44ED     STR:    CALL    TSTNUM          ; Make sure it's a number
F19D CDB8F9             CALL    NUMASC          ; Turn number into text
F1A0 CDCEF1             CALL    CRTST           ; Create string entry for it
F1A3 CD53F3             CALL    GSTRCU          ; Current string to pool
F1A6 01AEF3             LD      BC,TOPOOL       ; Save in string pool
F1A9 C5                 PUSH    BC              ; Save address on stack

F1AA 7E         SAVSTR: LD      A,(HL)          ; Get string length
F1AB 23                 INC     HL
F1AC 23                 INC     HL
F1AD E5                 PUSH    HL              ; Save pointer to string
F1AE CD29F2             CALL    TESTR           ; See if enough string space
F1B1 E1                 POP     HL              ; Restore pointer to string
F1B2 4E                 LD      C,(HL)          ; Get LSB of address
F1B3 23                 INC     HL
F1B4 46                 LD      B,(HL)          ; Get MSB of address
F1B5 CDC2F1             CALL    CRTMST          ; Create string entry
F1B8 E5                 PUSH    HL              ; Save pointer to MSB of addr
F1B9 6F                 LD      L,A             ; Length of string
F1BA CD46F3             CALL    TOSTRA          ; Move to string area
F1BD D1                 POP     DE              ; Restore pointer to MSB
F1BE C9                 RET

F1BF CD29F2     MKTMST: CALL    TESTR           ; See if enough string space
F1C2 21BF10     CRTMST: LD      HL,TMPSTR       ; Temporary string
F1C5 E5                 PUSH    HL              ; Save it
F1C6 77                 LD      (HL),A          ; Save length of string
F1C7 23                 INC     HL
F1C8 23         SVSTAD: INC     HL
F1C9 73                 LD      (HL),E          ; Save LSB of address
F1CA 23                 INC     HL
F1CB 72                 LD      (HL),D          ; Save MSB of address
F1CC E1                 POP     HL              ; Restore pointer
F1CD C9                 RET
```

```
F1CE 2B          CRTST:  DEC     HL              ; DEC - INCed after
F1CF 0622        QTSTR:  LD      B,'"'           ; Terminating quote
F1D1 50                  LD      D,B             ; Quote to D
F1D2 E5          DTSTR:  PUSH    HL              ; Save start
F1D3 0EFF                LD      C,-1            ; Set counter to -1
F1D5 23          QTSTLP: INC     HL              ; Move on
F1D6 7E                  LD      A,(HL)          ; Get byte
F1D7 0C                  INC     C               ; Count bytes
F1D8 B7                  OR      A               ; End of line?
F1D9 CAE4F1              JP      Z,CRTSTE        ; Yes - Create string entry
F1DC BA                  CP      D               ; Terminator D found?
F1DD CAE4F1              JP      Z,CRTSTE        ; Yes - Create strig entry
F1E0 B8                  CP      B               ; Terminator B found?
F1E1 C2D5F1              JP      NZ,QTSTLP       ; No - Keep looking
F1E4 FE22        CRTSTE: CP      '"'             ; End with '"'?
F1E6 CC36E8              CALL    Z,GETCHR        ; Yes - Get next character
F1E9 E3                  EX      (SP),HL         ; Starting quote
F1EA 23                  INC     HL              ; First byte of string
F1EB EB                  EX      DE,HL           ; To DE
F1EC 79                  LD      A,C             ; Get length
F1ED CDC2F1              CALL    CRTMST          ; Create string entry
F1F0 11BF10      TSTOPL: LD      DE,TMPSTR       ; Temporary string
F1F3 2AB110              LD      HL,(TMSTPT)     ; Temporary string pool pointer
F1F6 22E410              LD      (FPREG),HL      ; Save address of string ptr
F1F9 3E01                LD      A,1
F1FB 32AD10              LD      (TYPE),A        ; Set type to string
F1FE CD6EF8              CALL    DETHL4          ; Move string to pool
F201 CD8AE6              CALL    CPDEHL          ; Out of string pool?
F204 22B110              LD      (TMSTPT),HL     ; Save new pointer
F207 E1                  POP     HL              ; Restore code string address
F208 7E                  LD      A,(HL)          ; Get next code byte
F209 C0                  RET     NZ              ; Return if pool OK
F20A 1E1E                LD      E,ST            ; ?ST Error
F20C C3C1E3              JP      ERROR           ; String pool overflow

F20F 23          PRNUMS: INC     HL              ; Skip leading space
F210 CDCEF1      PRS:    CALL    CRTST           ; Create string entry for it
F213 CD53F3      PRS1:   CALL    GSTRCU          ; Current string to pool
F216 CD62F8              CALL    LOADFP          ; Move string block to BCDE
F219 1C                  INC     E               ; Length + 1
F21A 1D          PRSLP:  DEC     E               ; Count characters
F21B C8                  RET     Z               ; End of string
F21C 0A                  LD      A,(BC)          ; Get byte to output
F21D CD9BE6              CALL    OUTC            ; Output character in A
F220 FE0D                CP      CR              ; Return?
F222 CC86EB              CALL    Z,DONULL        ; Yes - Do nulls
F225 03                  INC     BC              ; Next byte in string
F226 C31AF2              JP      PRSLP           ; More characters to output
```

```
F229 B7          TESTR:  OR      A               ; Test if enough room
F22A OE                  DEFB    (LD C,n)        ; No garbage collection done
F22B F1          GRBDON: POP     AF              ; Garbage collection done
F22C F5                  PUSH    AF              ; Save status
F22D 2A5A10              LD      HL,(STRSPC)     ; Bottom of string space in use
F230 EB                  EX      DE,HL           ; To DE
F231 2AC310              LD      HL,(STRBOT)     ; Bottom of string area
F234 2F                  CPL                     ; Negate length (Top down)
F235 4F                  LD      C,A             ; -Length to BC
F236 06FF                LD      B,-1            ; BC = -ve length of string
F238 09                  ADD     HL,BC           ; Add to bottom of space in use
F239 23                  INC     HL              ; Plus one for 2's complement
F23A CD8AE6              CALL    CPDEHL          ; Below string RAM area?
F23D DA47F2              JP      C,TESTOS        ; Tidy up if not done else err
F240 22C310              LD      (STRBOT),HL     ; Save new bottom of area
F243 23                  INC     HL              ; Point to first byte of string
F244 EB                  EX      DE,HL           ; Address to DE
F245 F1          POPAF:  POP     AF              ; Throw away status push
F246 C9                  RET

F247 F1          TESTOS: POP     AF              ; Garbage collect been done?
F248 1E1A                LD      E,OS            ; ?OS Error
F24A CAC1E3              JP      Z,ERROR         ; Yes - Not enough string space
F24D BF                  CP      A               ; Flag garbage collect done
F24E F5                  PUSH    AF              ; Save status
F24F 012BF2              LD      BC,GRBDON       ; Garbage collection done
F252 C5                  PUSH    BC              ; Save for RETurn
F253 2AAF10      GARBGE: LD      HL,(LSTRAM)     ; Get end of RAM pointer
F256 22C310      GARBLP: LD      (STRBOT),HL     ; Reset string pointer
F259 210000              LD      HL,0
F25C E5                  PUSH    HL              ; Flag no string found
F25D 2A5A10              LD      HL,(STRSPC)     ; Get bottom of string space
F260 E5                  PUSH    HL              ; Save bottom of string space
F261 21B310              LD      HL,TMSTPL       ; Temporary string pool
F264 EB          GRBLP:  EX      DE,HL
F265 2AB110              LD      HL,(TMSTPT)     ; Temporary string pool pointer
F268 EB                  EX      DE,HL
F269 CD8AE6              CALL    CPDEHL          ; Temporary string pool done?
F26C 0164F2              LD      BC,GRBLP        ; Loop until string pool done
F26F C2B8F2              JP      NZ,STPOOL       ; No - See if in string area
F272 2AD610              LD      HL,(PROGND)     ; Start of simple variables
F275 EB          SMPVAR: EX      DE,HL
F276 2AD810              LD      HL,(VAREND)     ; End of simple variables
F279 EB                  EX      DE,HL
F27A CD8AE6              CALL    CPDEHL          ; All simple strings done?
F27D CA8BF2              JP      Z,ARRLP         ; Yes - Do string arrays
F280 7E                  LD      A,(HL)          ; Get type of variable
F281 23                  INC     HL
F282 23                  INC     HL
F283 B7                  OR      A               ; "S" flag set if string
F284 CDBBF2              CALL    STRADD          ; See if string in string area
F287 C375F2              JP      SMPVAR          ; Loop until simple ones done
```

```
F28A C1          GNXARY: POP     BC              ; Scrap address of this array
F28B EB          ARRLP:  EX      DE,HL
F28C 2ADA10              LD      HL,(ARREND)     ; End of string arrays
F28F EB                  EX      DE,HL
F290 CD8AE6              CALL    CPDEHL          ; All string arrays done?
F293 CAE1F2              JP      Z,SCNEND        ; Yes - Move string if found
F296 CD62F8              CALL    LOADFP          ; Get array name to BCDE
F299 7B                  LD      A,E             ; Get type of array
F29A E5                  PUSH    HL              ; Save address of num of dim'ns
F29B 09                  ADD     HL,BC           ; Start of next array
F29C B7                  OR      A               ; Test type of array
F29D F28AF2              JP      P,GNXARY        ; Numeric array - Ignore it
F2A0 22C510              LD      (CUROPR),HL     ; Save address of next array
F2A3 E1                  POP     HL              ; Get address of num of dim'ns
F2A4 4E                  LD      C,(HL)          ; BC = Number of dimensions
F2A5 0600                LD      B,0
F2A7 09                  ADD     HL,BC           ; Two bytes per dimension size
F2A8 09                  ADD     HL,BC
F2A9 23                  INC     HL              ; Plus one for number of dim'ns
F2AA EB          GRBARY: EX      DE,HL
F2AB 2AC510              LD      HL,(CUROPR)     ; Get address of next array
F2AE EB                  EX      DE,HL
F2AF CD8AE6              CALL    CPDEHL          ; Is this array finished?
F2B2 CA8BF2              JP      Z,ARRLP         ; Yes - Get next one
F2B5 01AAF2              LD      BC,GRBARY       ; Loop until array all done
F2B8 C5          STPOOL: PUSH    BC              ; Save return address
F2B9 F680                OR      80H             ; Flag string type
F2BB 7E          STRADD: LD      A,(HL)          ; Get string length
F2BC 23                  INC     HL
F2BD 23                  INC     HL
F2BE 5E                  LD      E,(HL)          ; Get LSB of string address
F2BF 23                  INC     HL
F2C0 56                  LD      D,(HL)          ; Get MSB of string address
F2C1 23                  INC     HL
F2C2 F0                  RET     P               ; Not a string - Return
F2C3 B7                  OR      A               ; Set flags on string length
F2C4 C8                  RET     Z               ; Null string - Return
F2C5 44                  LD      B,H             ; Save variable pointer
F2C6 4D                  LD      C,L
F2C7 2AC310              LD      HL,(STRBOT)     ; Bottom of new area
F2CA CD8AE6              CALL    CPDEHL          ; String been done?
F2CD 60                  LD      H,B             ; Restore variable pointer
F2CE 69                  LD      L,C
F2CF D8                  RET     C               ; String done - Ignore
F2D0 E1                  POP     HL              ; Return address
F2D1 E3                  EX      (SP),HL         ; Lowest available string area
F2D2 CD8AE6              CALL    CPDEHL          ; String within string area?
F2D5 E3                  EX      (SP),HL         ; Lowest available string area
F2D6 E5                  PUSH    HL              ; Re-save return address
F2D7 60                  LD      H,B             ; Restore variable pointer
F2D8 69                  LD      L,C
F2D9 D0                  RET     NC              ; Outside string area - Ignore
F2DA C1F1F1              POP     BC,AF,AF        ; Get return , Throw 2 away
F2DD E5                  PUSH    HL              ; Save variable pointer
F2DE D5                  PUSH    DE              ; Save address of current
F2DF C5                  PUSH    BC              ; Put back return address
F2E0 C9                  RET                     ; Go to it
```

```
F2E1 D1E1      SCNEND: POP     DE,HL           ; Addresses of strings
F2E3 7D                LD      A,L             ; HL = 0 if no more to do
F2E4 B4                OR      H
F2E5 C8                RET     Z               ; No more to do - Return
F2E6 2B                DEC     HL
F2E7 46                LD      B,(HL)          ; MSB of address of string
F2E8 2B                DEC     HL
F2E9 4E                LD      C,(HL)          ; LSB of address of string
F2EA E5                PUSH    HL              ; Save variable address
F2EB 2B                DEC     HL
F2EC 2B                DEC     HL
F2ED 6E                LD      L,(HL)          ; HL = Length of string
F2EE 2600              LD      H,0
F2F0 09                ADD     HL,BC           ; Address of end of string+1
F2F1 50                LD      D,B             ; String address to DE
F2F2 59                LD      E,C
F2F3 2B                DEC     HL              ; Last byte in string
F2F4 44                LD      B,H             ; Address to BC
F2F5 4D                LD      C,L
F2F6 2AC310            LD      HL,(STRBOT)     ; Current bottom of string area
F2F9 CD7CE3            CALL    MOVSTR          ; Move string to new address
F2FC E1                POP     HL              ; Restore variable address
F2FD 71                LD      (HL),C          ; Save new LSB of address
F2FE 23                INC     HL
F2FF 70                LD      (HL),B          ; Save new MSB of address
F300 69                LD      L,C             ; Next string area+1 to HL
F301 60                LD      H,B
F302 2B                DEC     HL              ; Next string area address
F303 C356F2            JP      GARBLP          ; Look for more strings

F306 C5E5      CONCAT: PUSH    BC,HL           ; Save prec' opr & code string
F308 2AE410            LD      HL,(FPREG)      ; Get first string
F30B E3                EX      (SP),HL         ; Save first string
F30C CDD1ED            CALL    OPRND           ; Get second string
F30F E3                EX      (SP),HL         ; Restore first string
F310 CD45ED            CALL    TSTSTR          ; Make sure it's a string
F313 7E                LD      A,(HL)          ; Get length of second string
F314 E5                PUSH    HL              ; Save first string
F315 2AE410            LD      HL,(FPREG)      ; Get second string
F318 E5                PUSH    HL              ; Save second string
F319 86                ADD     A,(HL)          ; Add length of second string
F31A 1E1C              LD      E,LS            ; ?LS Error
F31C DAC1E3            JP      C,ERROR         ; String too long - Error
F31F CDBFF1            CALL    MKTMST          ; Make temporary string
F322 D1                POP     DE              ; Get second string to DE
F323 CD57F3            CALL    GSTRDE          ; Move to string pool if needed
F326 E3                EX      (SP),HL         ; Get first string
F327 CD56F3            CALL    GSTRHL          ; Move to string pool if needed
F32A E5                PUSH    HL              ; Save first string
F32B 2AC110            LD      HL,(TMPSTR+2)   ; Temporary string address
F32E EB                EX      DE,HL           ; To DE
F32F CD3DF3            CALL    SSTSA           ; First string to string area
F332 CD3DF3            CALL    SSTSA           ; Second string to string area
F335 2166ED            LD      HL,EVAL2        ; Return to evaluation loop
F338 E3                EX      (SP),HL         ; Save return,get code string
F339 E5                PUSH    HL              ; Save code string address
F33A C3F0F1            JP      TSTOPL          ; To temporary string to pool
```

```
F33D E1          SSTSA:  POP     HL              ; Return address
F33E E3                  EX      (SP),HL         ; Get string block,save return
F33F 7E                  LD      A,(HL)          ; Get length of string
F340 23                  INC     HL
F341 23                  INC     HL
F342 4E                  LD      C,(HL)          ; Get LSB of string address
F343 23                  INC     HL
F344 46                  LD      B,(HL)          ; Get MSB of string address
F345 6F                  LD      L,A             ; Length to L
F346 2C          TOSTRA: INC     L               ; INC - DECed after
F347 2D          TSALP:  DEC     L               ; Count bytes moved
F348 C8                  RET     Z               ; End of string - Return
F349 0A                  LD      A,(BC)          ; Get source
F34A 12                  LD      (DE),A          ; Save destination
F34B 03                  INC     BC              ; Next source
F34C 13                  INC     DE              ; Next destination
F34D C347F3              JP      TSALP           ; Loop until string moved

F350 CD45ED      GETSTR: CALL    TSTSTR          ; Make sure it's a string
F353 2AE410      GSTRCU: LD      HL,(FPREG)      ; Get current string
F356 EB          GSTRHL: EX      DE,HL           ; Save DE
F357 CD71F3      GSTRDE: CALL    BAKTMP          ; Was it last tmp-str?
F35A EB                  EX      DE,HL           ; Restore DE
F35B C0                  RET     NZ              ; No - Return
F35C D5                  PUSH    DE              ; Save string
F35D 50                  LD      D,B             ; String block address to DE
F35E 59                  LD      E,C
F35F 1B                  DEC     DE              ; Point to length
F360 4E                  LD      C,(HL)          ; Get string length
F361 2AC310              LD      HL,(STRBOT)     ; Current bottom of string area
F364 CD8AE6              CALL    CPDEHL          ; Last one in string area?
F367 C26FF3              JP      NZ,POPHL        ; No - Return
F36A 47                  LD      B,A             ; Clear B (A=0)
F36B 09                  ADD     HL,BC           ; Remove string from str' area
F36C 22C310              LD      (STRBOT),HL     ; Save new bottom of str' area
F36F E1          POPHL:  POP     HL              ; Restore string
F370 C9                  RET

F371 2AB110      BAKTMP: LD      HL,(TMSTPT)     ; Get temporary string pool top
F374 2B                  DEC     HL              ; Back
F375 46                  LD      B,(HL)          ; Get MSB of address
F376 2B                  DEC     HL              ; Back
F377 4E                  LD      C,(HL)          ; Get LSB of address
F378 2B                  DEC     HL              ; Back
F379 2B                  DEC     HL              ; Back
F37A CD8AE6              CALL    CPDEHL          ; String last in string pool?
F37D C0                  RET     NZ              ; Yes - Leave it
F37E 22B110              LD      (TMSTPT),HL     ; Save new string pool top
F381 C9                  RET
```

```
F382 0101F1     LEN:    LD      BC,PASSA        ; To return integer A
F385 C5                 PUSH    BC              ; Save address
F386 CD50F3     GETLEN: CALL    GETSTR          ; Get string and its length
F389 AF                 XOR     A
F38A 57                 LD      D,A             ; Clear D
F38B 32AD10             LD      (TYPE),A        ; Set type to numeric
F38E 7E                 LD      A,(HL)          ; Get length of string
F38F B7                 OR      A               ; Set status flags
F390 C9                 RET

F391 0101F1     ASC:    LD      BC,PASSA        ; To return integer A
F394 C5                 PUSH    BC              ; Save address
F395 CD86F3     GTFLNM: CALL    GETLEN          ; Get length of string
F398 CAA0E9             JP      Z,FCERR         ; Null string - Error
F39B 23                 INC     HL
F39C 23                 INC     HL
F39D 5E                 LD      E,(HL)          ; Get LSB of address
F39E 23                 INC     HL
F39F 56                 LD      D,(HL)          ; Get MSB of address
F3A0 1A                 LD      A,(DE)          ; Get first byte of string
F3A1 C9                 RET

F3A2 3E01       CHR:    LD      A,1             ; One character string
F3A4 CDBFF1             CALL    MKTMST          ; Make a temporary string
F3A7 CD87F4             CALL    MAKINT          ; Make it integer A
F3AA 2AC110             LD      HL,(TMPSTR+2)   ; Get address of string
F3AD 73                 LD      (HL),E          ; Save character
F3AE C1         TOPOOL: POP     BC              ; Clean up stack
F3AF C3F0F1             JP      TSTOPL          ; Temporary string to pool
```

```
F3B2 CD37F4     LEFT:    CALL    LFRGNM          ; Get number and ending ")"
F3B5 AF                  XOR     A               ; Start at first byte in string
F3B6 E3         RIGHT1:  EX      (SP),HL         ; Save code string,Get string
F3B7 4F                  LD      C,A             ; Starting position in string
F3B8 E5         MID1:    PUSH    HL              ; Save string block address
F3B9 7E                  LD      A,(HL)          ; Get length of string
F3BA B8                  CP      B               ; Compare with number given
F3BB DAC0F3              JP      C,ALLFOL        ; All following bytes required
F3BE 78                  LD      A,B             ; Get new length
F3BF 11                  DEFB    (LD DE,nn)      ; Skip "LD C,0"
F3C0 0E00       ALLFOL:  LD      C,0             ; First byte of string
F3C2 C5                  PUSH    BC              ; Save position in string
F3C3 CD29F2              CALL    TESTR           ; See if enough string space
F3C6 C1                  POP     BC              ; Get position in string
F3C7 E1                  POP     HL              ; Restore string block address
F3C8 E5                  PUSH    HL              ; And re-save it
F3C9 23                  INC     HL
F3CA 23                  INC     HL
F3CB 46                  LD      B,(HL)          ; Get LSB of address
F3CC 23                  INC     HL
F3CD 66                  LD      H,(HL)          ; Get MSB of address
F3CE 68                  LD      L,B             ; HL = address of string
F3CF 0600                LD      B,0             ; BC = starting address
F3D1 09                  ADD     HL,BC           ; Point to that byte
F3D2 44                  LD      B,H             ; BC = source string
F3D3 4D                  LD      C,L
F3D4 CDC2F1              CALL    CRTMST          ; Create a string entry
F3D7 6F                  LD      L,A             ; Length of new string
F3D8 CD46F3              CALL    TOSTRA          ; Move string to string area
F3DB D1                  POP     DE              ; Clear stack
F3DC CD57F3              CALL    GSTRDE          ; Move to string pool if needed
F3DF C3F0F1              JP      TSTOPL          ; Temporary string to pool

F3E2 CD37F4     RIGHT:   CALL    LFRGNM          ; Get number and ending ")"
F3E5 D1                  POP     DE              ; Get string length
F3E6 D5                  PUSH    DE              ; And re-save
F3E7 1A                  LD      A,(DE)          ; Get length
F3E8 90                  SUB     B               ; Move back N bytes
F3E9 C3B6F3              JP      RIGHT1          ; Go and get sub-string
```

```
F3EC EB          MID:    EX      DE,HL       ; Get code string address
F3ED 7E                  LD      A,(HL)      ; Get next byte "," or ")"
F3EE CD3CF4              CALL    MIDNUM      ; Get number supplied
F3F1 04                  INC     B           ; Is it character zero?
F3F2 05                  DEC     B
F3F3 CAA0E9              JP      Z,FCERR     ; Yes - Error
F3F6 C5                  PUSH    BC          ; Save starting position
F3F7 1EFF                LD      E,255       ; All of string
F3F9 FE29                CP      ")"         ; Any length given?
F3FB CA05F4              JP      Z,RSTSTR    ; No - Rest of string
F3FE CD90E6              CALL    CHKSYN      ; Make sure "," follows
F401 2C                  DEFB    ","
F402 CD84F4              CALL    GETINT      ; Get integer 0-255
F405 CD90E6      RSTSTR: CALL    CHKSYN      ; Make sure ")" follows
F408 29                  DEFB    ")"
F409 F1                  POP     AF          ; Restore starting position
F40A E3                  EX      (SP),HL     ; Get string,save code string
F40B 01B8F3              LD      BC,MID1     ; Continuation of MID$ routine
F40E C5                  PUSH    BC          ; Save for return
F40F 3D                  DEC     A           ; Starting position-1
F410 BE                  CP      (HL)        ; Compare with length
F411 0600                LD      B,0         ; Zero bytes length
F413 D0                  RET     NC          ; Null string if start past end
F414 4F                  LD      C,A         ; Save starting position-1
F415 7E                  LD      A,(HL)      ; Get length of string
F416 91                  SUB     C           ; Subtract start
F417 BB                  CP      E           ; Enough string for it?
F418 47                  LD      B,A         ; Save maximum length available
F419 D8                  RET     C           ; Truncate string if needed
F41A 43                  LD      B,E         ; Set specified length
F41B C9                  RET                 ; Go and create string

F41C CD86F3      VAL:    CALL    GETLEN      ; Get length of string
F41F CA33F6              JP      Z,RESZER    ; Result zero
F422 5F                  LD      E,A         ; Save length
F423 23                  INC     HL
F424 23                  INC     HL
F425 7E                  LD      A,(HL)      ; Get LSB of address
F426 23                  INC     HL
F427 66                  LD      H,(HL)      ; Get MSB of address
F428 6F                  LD      L,A         ; HL = String address
F429 E5                  PUSH    HL          ; Save string address
F42A 19                  ADD     HL,DE       ;
F42B 46                  LD      B,(HL)      ; Get end of string+1 byte
F42C 72                  LD      (HL),D      ; Zero it to terminate
F42D E3                  EX      (SP),HL     ; Save string end,get start
F42E C5                  PUSH    BC          ; Save end+1 byte
F42F 7E                  LD      A,(HL)      ; Get starting byte
F430 CD1AF9              CALL    ASCTFP      ; Convert ASCII string to FP
F433 C1                  POP     BC          ; Restore end+1 byte
F434 E1                  POP     HL          ; Restore end+1 address
F435 70                  LD      (HL),B      ; Put back original byte
F436 C9                  RET
```

```
F437 EB        LFRGNM: EX      DE,HL           ; Code string address to HL
F438 CD90E6            CALL    CHKSYN          ; Make sure ")" follows
F43B 29                DEFB    ")"
F43C C1        MIDNUM: POP     BC              ; Get return address
F43D D1                POP     DE              ; Get number supplied
F43E C5                PUSH    BC              ; Re-save return address
F43F 43                LD      B,E             ; Number to B
F440 C9                RET

F441 CD87F4    INP:    CALL    MAKINT          ; Make it integer A
F444 323F10            LD      (INPORT),A      ; Set input port
F447 CD3E10            CALL    INPSUB          ; Get input from port
F44A C301F1            JP      PASSA           ; Return integer A

F44D CD71F4    POUT:   CALL    SETIO           ; Set up port number
F450 C30610            JP      OUTSUB          ; Output data and return

F453 CD71F4    WAIT:   CALL    SETIO           ; Set up port number
F456 F5                PUSH    AF              ; Save AND mask
F457 1E00              LD      E,0             ; Assume zero if none given
F459 2B                DEC     HL              ; DEC 'cos GETCHR INCs
F45A CD36E8            CALL    GETCHR          ; Get next character
F45D CA67F4            JP      Z,NOXOR         ; No XOR byte given
F460 CD90E6            CALL    CHKSYN          ; Make sure "," follows
F463 2C                DEFB    ","
F464 CD84F4            CALL    GETINT          ; Get integer 0-255 to XOR with
F467 C1        NOXOR:  POP     BC              ; Restore AND mask
F468 CD3E10    WAITLP: CALL    INPSUB          ; Get input
F46B AB                XOR     E               ; Flip selected bits
F46C A0                AND     B               ; Result non-zero?
F46D CA68F4            JP      Z,WAITLP        ; No = keep waiting
F470 C9                RET

F471 CD84F4    SETIO:  CALL    GETINT          ; Get integer 0-255
F474 323F10            LD      (INPORT),A      ; Set input port
F477 320710            LD      (OTPORT),A      ; Set output port
F47A CD90E6            CALL    CHKSYN          ; Make sure "," follows
F47D 2C                DEFB    ","
F47E C384F4            JP      GETINT          ; Get integer 0-255 and return

F481 CD36E8    FNDNUM: CALL    GETCHR          ; Get next character
F484 CD41ED    GETINT: CALL    GETNUM          ; Get a number from 0 to 255
F487 CD85E9    MAKINT: CALL    DEPINT          ; Make sure value 0 - 255
F48A 7A                LD      A,D             ; Get MSB of number
F48B B7                OR      A               ; Zero?
F48C C2A0E9            JP      NZ,FCERR        ; No - Error
F48F 2B                DEC     HL              ; DEC 'cos GETCHR INCs
F490 CD36E8            CALL    GETCHR          ; Get next character
F493 7B                LD      A,E             ; Get number to A
F494 C9                RET
```

```
                        ; << NO REFERENCE TO THIS SECTION OF CODE >>
                        ; << Set up another program area (can be in ROM) >>

F495 2A5E10             LD      HL,(BASTXT)     ; Get start of program text
F498 22D610             LD      (PROGND),HL     ; Set more variable space
F49B 210080             LD      HL,8000H        ; Address of new program
F49E 5E                 LD      E,(HL)          ; Get LSB of new RAM end
F49F 23                 INC     HL
F4A0 56                 LD      D,(HL)          ; Get MSB of new RAM end
F4A1 23                 INC     HL
F4A2 23                 INC     HL              ; Null at start of program
F4A3 225E10             LD      (BASTXT),HL     ; New program text area 8003H
F4A6 EB                 EX      DE,HL           ; New RAM end to HL
F4A7 22AF10             LD      (LSTRAM),HL     ; Set new RAM end
F4AA 225A10             LD      (STRSPC),HL     ; Clear string space
F4AD 01F2E7             LD      BC,RUNCNT       ; Execution driver loop
F4B0 C5                 PUSH    BC              ; Save for return
F4B1 C3C5E4             JP      RUNFST          ; Clear variables and continue

F4B4 C356FD     RUART:  JP      GUART           ; Get a byte from UART

F4B7 CDBAF4     WUART2: CALL    WUART           ; Send 2 Bytes to UART
F4BA F5         WUART:  PUSH    AF              ; Save byte
F4BB C5                 PUSH    BC              ; Save BC
F4BC 4F                 LD      C,A             ; Byte to C
F4BD CD68FD             CALL    SUART           ; Send byte to UART
F4C0 C1                 POP     BC              ; Restore BC
F4C1 F1                 POP     AF              ; Restore byte
F4C2 C9                 RET

F4C3 0601       CSAVE:  LD      B,1             ; Flag "CSAVE"
F4C5 FEAE               CP      ZTIMES          ; "*" token? ("CSAVE*")
F4C7 CABBE8             JP      Z,ARRSV1        ; Yes - Array save
F4CA CD5AED             CALL    EVAL            ; Evaluate expression
F4CD E5                 PUSH    HL              ; Save code string address
F4CE CD95F3             CALL    GTFLNM          ; Get file name
F4D1 D5                 PUSH    DE              ; Save file name
F4D2 CDC8FC             CALL    CASFFW          ; Turn on motor and wait
F4D5 D1                 POP     DE              ; Restore file name
F4D6 3ED3               LD      A,11010011B     ; Header byte
F4D8 CDBAF4             CALL    WUART           ; Send byte to UART
F4DB CDB7F4             CALL    WUART2          ; Send byte twice more
F4DE 1A                 LD      A,(DE)          ; Get file name
F4DF CDBAF4             CALL    WUART           ; Send it to UART
F4E2 00                 NOP
F4E3 00                 NOP
F4E4 00                 NOP
F4E5 21D610             LD      HL,PROGND       ; Start of program information
F4E8 220C0C             LD      (ARG1),HL       ; Save for monitor save routine
F4EB 2AD610             LD      HL,(PROGND)     ; End of program information
F4EE 220E0C             LD      (ARG2),HL       ; Save for monitor save routine
F4F1 CD73FE             CALL    SAVE            ; Save program to tape
F4F4 CDD8FC             CALL    ARET            ; Not much there!
F4F7 E1                 POP     HL              ; Restore code string address
F4F8 C9                 RET
```

```
F4F9 7E          CLOAD:  LD      A,(HL)          ; Get byte after "CLOAD"
F4FA FEAE                CP      ZTIMES          ; "*" token? ("CLOAD*")
F4FC CAB9E8              JP      Z,ARRLD1        ; Yes - Array load
F4FF CDD1FF              CALL    SMOTOR          ; Start motor and get "?"
F502 D69E                SUB     ZPRINT          ; "?" ("PRINT" token) Verify?
F504 CA09F5              JP      Z,FLGVER        ; Yes - Flag "verify"
F507 AF                  XOR     A               ; Flag "load"
F508 01                  DEFB    (LD BC,nn)      ; Skip "CPL" and "INC HL"
F509 2F          FLGVER: CPL                     ; Flag "verify"
F50A 23                  INC     HL              ; Skip over "?"
F50B F5                  PUSH    AF              ; Save verify flag
F50C 2B                  DEC     HL              ; DEC 'cos GETCHR INCs
F50D CD36E8              CALL    GETCHR          ; Get next character
F510 3E00                LD      A,0             ; Any file will do
F512 CA1CF5              JP      Z,ANYNAM        ; No name given - Any will do
F515 CD5AED              CALL    EVAL            ; Evaluate expression
F518 CD95F3              CALL    GTFLNM          ; Get file name
F51B 1A                  LD      A,(DE)          ; Get first byte of name
F51C 6F          ANYNAM: LD      L,A             ; Save name to find
F51D F1                  POP     AF              ; Get verify flag
F51E F5                  PUSH    AF              ; And re-save
F51F B7                  OR      A               ; Verify of load?
F520 67                  LD      H,A
F521 22E410              LD      (FPREG),HL      ; Save nam of file to find
F524 CCBAE4              CALL    Z,CLRPTR        ; Load - Clear pointers
F527 2AE410              LD      HL,(FPREG)      ; Get name of program to find
F52A EB                  EX      DE,HL           ; Name to DE
F52B 0603        CLOAD1: LD      B,3             ; 3 Header bytes
F52D CDB4F4      CLOAD2: CALL    RUART           ; Get a byte from UART
F530 D6D3                SUB     11010011B       ; Header byte?
F532 C22BF5              JP      NZ,CLOAD1       ; Look for header
F535 05                  DEC     B               ; Count header bytes
F536 C22DF5              JP      NZ,CLOAD2       ; More to find?
F539 CDB4F4              CALL    RUART           ; Get name of file
F53C CD74F5              CALL    FILFND          ; Display "file X found"
F53F 1C                  INC     E               ; Any file name given?
F540 1D                  DEC     E
F541 CA48F5              JP      Z,THSFIL        ; No - This file will do
F544 BB                  CP      E               ; Has file been found?
F545 C22BF5              JP      NZ,CLOAD1       ; No - Look for another
F548 00          THSFIL: NOP
F549 00                  NOP
F54A 00                  NOP
F54B F1                  POP     AF              ; Get verify flag
F54C B7                  OR      A               ; Load or verify?
F54D C25CF5              JP      NZ,CLOADV       ; Verify program
F550 CD88FE              CALL    MONLD           ; Use monitor to load program
F553 2AD610              LD      HL,(PROGND)     ; Get end of program
F556 CD93E3              CALL    ENFMEM          ; See if enough memory
F559 C35FF5              JP      CLOADE          ; "Ok" and set up pointers

F55C CDAAFE      CLOADV: CALL    MONVE           ; Use monitor to verify program
F55F 214BE3      CLOADE: LD      HL,OKMSG        ; "Ok" message
F562 CD10F2              CALL    PRS             ; Output string
F565 CDD8FC              CALL    ARET            ; Not a lot there!
F568 C37CE4              JP      SETPTR          ; Set up line pointers
```

```
F56B 219DF5     OUTBAD: LD      HL,BAD          ; "Bad" message
F56E CD10F2             CALL    PRS             ; Output string
F571 C3E1E3             JP      ERRIN           ; In line message

F574 C5         FILFND: PUSH    BC              ; <- Save
F575 E5                 PUSH    HL              ; <- all
F576 D5                 PUSH    DE              ; <- the
F577 F5                 PUSH    AF              ; <- registers
F578 218EF5             LD      HL,FILE         ; "File" message
F57B CD10F2             CALL    PRS             ; Output string
F57E F1                 POP     AF              ; Get file name
F57F F5                 PUSH    AF              ; And re-save
F580 CDD9FC             CALL    CONMON          ; Output file name to screen
F583 2194F5             LD      HL,FOUND        ; "Found" message
F586 CD10F2             CALL    PRS             ; Output string
F589 F1                 POP     AF              ; <- Restore
F58A D1                 POP     DE              ; <- all
F58B E1                 POP     HL              ; <- the
F58C C1                 POP     BC              ; <- registers
F58D C9                 RET

F58E 46696C65   FILE:   DEFB    "File ",0
F594 20466F75   FOUND:  DEFB    " Found",CR,LF,0
F59D 42616400   BAD:    DEFB    "Bad",0,0,0

F5A3 CD8BE9     PEEK:   CALL    DEINT           ; Get memory address
F5A6 1A                 LD      A,(DE)          ; Get byte in memory
F5A7 C301F1             JP      PASSA           ; Return integer A

F5AA CD41ED     POKE:   CALL    GETNUM          ; Get memory address
F5AD CD8BE9             CALL    DEINT           ; Get integer -32768 to 3276
F5B0 D5                 PUSH    DE              ; Save memory address
F5B1 CD90E6             CALL    CHKSYN          ; Make sure "," follows
F5B4 2C                 DEFB    ","
F5B5 CD84F4             CALL    GETINT          ; Get integer 0-255
F5B8 D1                 POP     DE              ; Restore memory address
F5B9 12                 LD      (DE),A          ; Load it into memory
F5BA C9                 RET

F5BB 2191FA     ROUND:  LD      HL,HALF         ; Add 0.5 to FPREG
F5BE CD62F8     ADDPHL: CALL    LOADFP          ; Load FP at (HL) to BCDE
F5C1 C3CDF5             JP      FPADD           ; Add BCDE to FPREG
```

```
F5C4 CD62F8     SUBPHL: CALL    LOADFP          ; FPREG = -FPREG + number at HL
F5C7 21                 DEFB    (LD HL,nn)      ; Skip "POP BC" and "POP DE"
F5C8 C1         PSUB:   POP     BC              ; Get FP number from stack
F5C9 D1                 POP     DE
F5CA CD3CF8     SUBCDE: CALL    INVSGN          ; Negate FPREG
F5CD 78         FPADD:  LD      A,B             ; Get FP exponent
F5CE B7                 OR      A               ; Is number zero?
F5CF C8                 RET     Z               ; Yes - Nothing to add
F5D0 3AE710             LD      A,(FPEXP)       ; Get FPREG exponent
F5D3 B7                 OR      A               ; Is this number zero?
F5D4 CA54F8             JP      Z,FPBCDE        ; Yes - Move BCDE to FPREG
F5D7 90                 SUB     B               ; BCDE number larger?
F5D8 D2E7F5             JP      NC,NOSWAP       ; No - Don't swap thm
F5DB 2F                 CPL                     ; Two's complement
F5DC 3C                 INC     A               ;  FP exponent
F5DD EB                 EX      DE,HL
F5DE CD44F8             CALL    STAKFP          ; Put FPREG on stack
F5E1 EB                 EX      DE,HL
F5E2 CD54F8             CALL    FPBCDE          ; Move BCDE to FPREG
F5E5 C1                 POP     BC              ; Restore number from stack
F5E6 D1                 POP     DE
F5E7 FE19       NOSWAP: CP      24+1            ; Second number insignificant?
F5E9 D0                 RET     NC              ; Yes - First number is result
F5EA F5                 PUSH    AF              ; Save number of bits to scale
F5EB CD79F8             CALL    SIGNS           ; Set MSBs & sign of result
F5EE 67                 LD      H,A             ; Save sign of result
F5EF F1                 POP     AF              ; Restore scaling factor
F5F0 CD92F6             CALL    SCALE           ; Scale BCDE to same exponent
F5F3 B4                 OR      H               ; Result to be positive?
F5F4 21E410             LD      HL,FPREG        ; Point to FPREG
F5F7 F20DF6             JP      P,MINCDE        ; No - Subtract FPREG from CDE
F5FA CD72F6             CALL    PLUCDE          ; Add FPREG to CDE
F5FD D253F6             JP      NC,RONDUP       ; No overflow - Round it up
F600 23                 INC     HL              ; Point to exponent
F601 34                 INC     (HL)            ; Increment it
F602 CABCE3             JP      Z,OVERR         ; Number overflowed - Error
F605 2E01               LD      L,1             ; 1 bit to shift right
F607 CDA8F6             CALL    SHRT1           ; Shift result right
F60A C353F6             JP      RONDUP          ; Round it up
F60D AF         MINCDE: XOR     A               ; Clear A and carry
F60E 90                 SUB     B               ; Negate exponent
F60F 47                 LD      B,A             ; Re-save exponent
F610 7E                 LD      A,(HL)          ; Get LSB of FPREG
F611 9B                 SBC     A,E             ; Subtract LSB of BCDE
F612 5F                 LD      E,A             ; Save LSB of BCDE
F613 23                 INC     HL
F614 7E                 LD      A,(HL)          ; Get NMSB of FPREG
F615 9A                 SBC     A,D             ; Subtract NMSB of BCDE
F616 57                 LD      D,A             ; Save NMSB of BCDE
F617 23                 INC     HL
F618 7E                 LD      A,(HL)          ; Get MSB of FPREG
F619 99                 SBC     A,C             ; Subtract MSB of BCDE
F61A 4F                 LD      C,A             ; Save MSB of BCDE
F61B DC7EF6     CONPOS: CALL    C,COMPL         ; Overflow - Make it positive
```

```
F61E 68          BNORM:   LD      L,B             ; L = Exponent
F61F 63                   LD      H,E             ; H = LSB
F620 AF                   XOR     A
F621 47          BNRMLP:  LD      B,A             ; Save bit count
F622 79                   LD      A,C             ; Get MSB
F623 B7                   OR      A               ; Is it zero?
F624 C240F6               JP      NZ,PNORM        ; No - Do it bit at a time
F627 4A                   LD      C,D             ; MSB = NMSB
F628 54                   LD      D,H             ; NMSB= LSB
F629 65                   LD      H,L             ; LSB = VLSB
F62A 6F                   LD      L,A             ; VLSB= 0
F62B 78                   LD      A,B             ; Get exponent
F62C D608                 SUB     8               ; Count 8 bits
F62E FEE0                 CP      -24-8           ; Was number zero?
F630 C221F6               JP      NZ,BNRMLP       ; No - Keep normalising
F633 AF          RESZER:  XOR     A               ; Result is zero
F634 32E710      SAVEXP:  LD      (FPEXP),A       ; Save result as zero
F637 C9                   RET

F638 05          NORMAL:  DEC     B               ; Count bits
F639 29                   ADD     HL,HL           ; Shift HL left
F63A 7A                   LD      A,D             ; Get NMSB
F63B 17                   RLA                     ; Shift left with last bit
F63C 57                   LD      D,A             ; Save NMSB
F63D 79                   LD      A,C             ; Get MSB
F63E 8F                   ADC     A,A             ; Shift left with last bit
F63F 4F                   LD      C,A             ; Save MSB
F640 F238F6      PNORM:   JP      P,NORMAL        ; Not done - Keep going
F643 78                   LD      A,B             ; Number of bits shifted
F644 5C                   LD      E,H             ; Save HL in EB
F645 45                   LD      B,L
F646 B7                   OR      A               ; Any shifting done?
F647 CA53F6               JP      Z,RONDUP        ; No - Round it up
F64A 21E710               LD      HL,FPEXP        ; Point to exponent
F64D 86                   ADD     A,(HL)          ; Add shifted bits
F64E 77                   LD      (HL),A          ; Re-save exponent
F64F D233F6               JP      NC,RESZER       ; Underflow - Result is zero
F652 C8                   RET     Z               ; Result is zero
F653 78          RONDUP:  LD      A,B             ; Get VLSB of number
F654 21E710      RONDB:   LD      HL,FPEXP        ; Point to exponent
F657 B7                   OR      A               ; Any rounding?
F658 FC65F6               CALL    M,FPROND        ; Yes - Round number up
F65B 46                   LD      B,(HL)          ; B = Exponent
F65C 23                   INC     HL
F65D 7E                   LD      A,(HL)          ; Get sign of result
F65E E680                 AND     10000000B       ; Only bit 7 needed
F660 A9                   XOR     C               ; Set correct sign
F661 4F                   LD      C,A             ; Save correct sign in number
F662 C354F8               JP      FPBCDE          ; Move BCDE to FPREG
```

```
F665 1C          FPROND: INC     E               ; Round LSB
F666 C0                  RET     NZ              ; Return if ok
F667 14                  INC     D               ; Round NMSB
F668 C0                  RET     NZ              ; Return if ok
F669 0C                  INC     C               ; Round MSB
F66A C0                  RET     NZ              ; Return if ok
F66B 0E80                LD      C,80H           ; Set normal value
F66D 34                  INC     (HL)            ; Increment exponent
F66E C0                  RET     NZ              ; Return if ok
F66F C3BCE3              JP      OVERR           ; Overflow error

F672 7E          PLUCDE: LD      A,(HL)          ; Get LSB of FPREG
F673 83                  ADD     A,E             ; Add LSB of BCDE
F674 5F                  LD      E,A             ; Save LSB of BCDE
F675 23                  INC     HL
F676 7E                  LD      A,(HL)          ; Get NMSB of FPREG
F677 8A                  ADC     A,D             ; Add NMSB of BCDE
F678 57                  LD      D,A             ; Save NMSB of BCDE
F679 23                  INC     HL
F67A 7E                  LD      A,(HL)          ; Get MSB of FPREG
F67B 89                  ADC     A,C             ; Add MSB of BCDE
F67C 4F                  LD      C,A             ; Save MSB of BCDE
F67D C9                  RET

F67E 21E810      COMPL:  LD      HL,SGNRES       ; Sign of result
F681 7E                  LD      A,(HL)          ; Get sign of result
F682 2F                  CPL                     ; Negate it
F683 77                  LD      (HL),A          ; Put it back
F684 AF                  XOR     A
F685 6F                  LD      L,A             ; Set L to zero
F686 90                  SUB     B               ; Negate exponent,set carry
F687 47                  LD      B,A             ; Re-save exponent
F688 7D                  LD      A,L             ; Load zero
F689 9B                  SBC     A,E             ; Negate LSB
F68A 5F                  LD      E,A             ; Re-save LSB
F68B 7D                  LD      A,L             ; Load zero
F68C 9A                  SBC     A,D             ; Negate NMSB
F68D 57                  LD      D,A             ; Re-save NMSB
F68E 7D                  LD      A,L             ; Load zero
F68F 99                  SBC     A,C             ; Negate MSB
F690 4F                  LD      C,A             ; Re-save MSB
F691 C9                  RET

F692 0600        SCALE:  LD      B,0             ; Clear underflow
F694 D608        SCALLP: SUB     8               ; 8 bits (a whole byte)?
F696 DAA1F6              JP      C,SHRITE        ; No - Shift right A bits
F699 43                  LD      B,E             ; <- Shift
F69A 5A                  LD      E,D             ; <- right
F69B 51                  LD      D,C             ; <- eight
F69C 0E00                LD      C,0             ; <- bits
F69E C394F6              JP      SCALLP          ; More bits to shift
```

```
F6A1 C609       SHRITE: ADD     A,8+1           ; Adjust count
F6A3 6F                 LD      L,A             ; Save bits to shift
F6A4 AF         SHRLP:  XOR     A               ; Flag for all done
F6A5 2D                 DEC     L               ; All shifting done?
F6A6 C8                 RET     Z               ; Yes - Return
F6A7 79                 LD      A,C             ; Get MSB
F6A8 1F         SHRT1:  RRA                     ; Shift it right
F6A9 4F                 LD      C,A             ; Re-save
F6AA 7A                 LD      A,D             ; Get NMSB
F6AB 1F                 RRA                     ; Shift right with last bit
F6AC 57                 LD      D,A             ; Re-save it
F6AD 7B                 LD      A,E             ; Get LSB
F6AE 1F                 RRA                     ; Shift right with last bit
F6AF 5F                 LD      E,A             ; Re-save it
F6B0 78                 LD      A,B             ; Get underflow
F6B1 1F                 RRA                     ; Shift right with last bit
F6B2 47                 LD      B,A             ; Re-save underflow
F6B3 C3A4F6             JP      SHRLP           ; More bits to do

F6B6 00000081   UNITY:  DEFB    000H,000H,000H,081H     ; 1.00000

F6BA 03         LOGTAB: DEFB    3                       ; Table used by LOG
F6BB AA561980           DEFB    0AAH,056H,019H,080H     ; 0.59898
F6BF F1227680           DEFB    0F1H,022H,076H,080H     ; 0.96147
F6C3 45AA3882           DEFB    045H,0AAH,038H,082H     ; 2.88539

F6C7 CD13F8     LOG:    CALL    TSTSGN          ; Test sign of value
F6CA B7                 OR      A
F6CB EAA0E9             JP      PE,FCERR        ; ?FC Error if <= zero
F6CE 21E710             LD      HL,FPEXP        ; Point to exponent
F6D1 7E                 LD      A,(HL)          ; Get exponent
F6D2 013580             LD      BC,8035H        ; BCDE = SQR(1/2)
F6D5 11F304             LD      DE,04F3H
F6D8 90                 SUB     B               ; Scale value to be < 1
F6D9 F5                 PUSH    AF              ; Save scale factor
F6DA 70                 LD      (HL),B          ; Save new exponent
F6DB D5                 PUSH    DE              ; Save SQR(1/2)
F6DC C5                 PUSH    BC
F6DD CDCDF5             CALL    FPADD           ; Add SQR(1/2) to value
F6E0 C1                 POP     BC              ; Restore SQR(1/2)
F6E1 D1                 POP     DE
F6E2 04                 INC     B               ; Make it SQR(2)
F6E3 CD69F7             CALL    DVBCDE          ; Divide by SQR(2)
F6E6 21B6F6             LD      HL,UNITY        ; Point to 1.
F6E9 CDC4F5             CALL    SUBPHL          ; Subtract FPREG from 1
F6EC 21BAF6             LD      HL,LOGTAB       ; Coefficient table
F6EF CD5BFB             CALL    SUMSER          ; Evaluate sum of series
F6F2 018080             LD      BC,8080H        ; BCDE = -0.5
F6F5 110000             LD      DE,0000H
F6F8 CDCDF5             CALL    FPADD           ; Subtract 0.5 from FPREG
F6FB F1                 POP     AF              ; Restore scale factor
F6FC CD8EF9             CALL    RSCALE          ; Re-scale number
F6FF 013180     MULLN2: LD      BC,8031H        ; BCDE = Ln(2)
F702 111872             LD      DE,7218H
F705 21                 DEFB    (LD HL,nn)      ; Skip "POP BC" and "POP DE"
```

```
F706 C1          MULT:    POP     BC              ; Get number from stack
F707 D1                   POP     DE
F708 CD13F8      FPMULT:  CALL    TSTSGN          ; Test sign of FPREG
F70B C8                   RET     Z               ; Return zero if zero
F70C 2E00                 LD      L,0             ; Flag add exponents
F70E CDD1F7               CALL    ADDEXP          ; Add exponents
F711 79                   LD      A,C             ; Get MSB of multiplier
F712 32F610               LD      (MULVAL),A      ; Save MSB of multiplier
F715 EB                   EX      DE,HL
F716 22F710               LD      (MULVAL+1),HL   ; Save rest of multiplier
F719 010000               LD      BC,0            ; Partial product (BCDE) = zero
F71C 50                   LD      D,B
F71D 58                   LD      E,B
F71E 211EF6               LD      HL,BNORM        ; Address of normalise
F721 E5                   PUSH    HL              ; Save for return
F722 212AF7               LD      HL,MULT8        ; Address of 8 bit multiply
F725 E5E5                 PUSH    HL,HL           ; Save for NMSB,MSB
F727 21E410               LD      HL,FPREG        ; Point to number
F72A 7E          MULT8:   LD      A,(HL)          ; Get LSB of number
F72B 23                   INC     HL              ; Point to NMSB
F72C B7                   OR      A               ; Test LSB
F72D CA56F7               JP      Z,BYTSFT        ; Zero - shift to next byte
F730 E5                   PUSH    HL              ; Save address of number
F731 2E08                 LD      L,8             ; 8 bits to multiply by
F733 1F          MUL8LP:  RRA                     ; Shift LSB right
F734 67                   LD      H,A             ; Save LSB
F735 79                   LD      A,C             ; Get MSB
F736 D244F7               JP      NC,NOMADD       ; Bit was zero - Don't add
F739 E5                   PUSH    HL              ; Save LSB and count
F73A 2AF710               LD      HL,(MULVAL+1)   ; Get LSB and NMSB
F73D 19                   ADD     HL,DE           ; Add NMSB and LSB
F73E EB                   EX      DE,HL           ; Leave sum in DE
F73F E1                   POP     HL              ; Restore MSB and count
F740 3AF610               LD      A,(MULVAL)      ; Get MSB of multiplier
F743 89                   ADC     A,C             ; Add MSB
F744 1F          NOMADD:  RRA                     ; Shift MSB right
F745 4F                   LD      C,A             ; Re-save MSB
F746 7A                   LD      A,D             ; Get NMSB
F747 1F                   RRA                     ; Shift NMSB right
F748 57                   LD      D,A             ; Re-save NMSB
F749 7B                   LD      A,E             ; Get LSB
F74A 1F                   RRA                     ; Shift LSB right
F74B 5F                   LD      E,A             ; Re-save LSB
F74C 78                   LD      A,B             ; Get VLSB
F74D 1F                   RRA                     ; Shift VLSB right
F74E 47                   LD      B,A             ; Re-save VLSB
F74F 2D                   DEC     L               ; Count bits multiplied
F750 7C                   LD      A,H             ; Get LSB of multiplier
F751 C233F7               JP      NZ,MUL8LP       ; More - Do it
F754 E1          POPHRT:  POP     HL              ; Restore address of number
F755 C9                   RET

F756 43          BYTSFT:  LD      B,E             ; Shift partial product left
F757 5A                   LD      E,D
F758 51                   LD      D,C
F759 4F                   LD      C,A
F75A C9                   RET
```

# NASCOM

# ROM

# BASIC

# DIS—ASSEMBLED

# PART 6

# BY CARL LLOYD—PARKER

```
F75B CD44F8     DIV10:  CALL    STAKFP          ; Save FPREG on stack
F75E 012084             LD      BC,8420H        ; BCDE = 10.
F761 110000             LD      DE,0000H
F764 CD54F8             CALL    FPBCDE          ; Move 10 to FPREG

F767 C1         DIV:    POP     BC              ; Get number from stack
F768 D1                 POP     DE
F769 CD13F8     DVBCDE: CALL    TSTSGN          ; Test sign of FPREG
F76C CAB0E3             JP      Z,DZERR         ; Error if division by zero
F76F 2EFF               LD      L,-1            ; Flag subtract exponents
F771 CDD1F7             CALL    ADDEXP          ; Subtract exponents
F774 34                 INC     (HL)            ; Add 2 to exponent to adjust
F775 34                 INC     (HL)
F776 2B                 DEC     HL              ; Point to MSB
F777 7E                 LD      A,(HL)          ; Get MSB of dividend
F778 321210             LD      (DIV3),A        ; Save for subtraction
F77B 2B                 DEC     HL
F77C 7E                 LD      A,(HL)          ; Get NMSB of dividend
F77D 320E10             LD      (DIV2),A        ; Save for subtraction
F780 2B                 DEC     HL
F781 7E                 LD      A,(HL)          ; Get MSB of dividend
F782 320A10             LD      (DIV1),A        ; Save for subtraction
F785 41                 LD      B,C             ; Get MSB
F786 EB                 EX      DE,HL           ; NMSB,LSB to HL
F787 AF                 XOR     A
F788 4F                 LD      C,A             ; Clear MSB of quotient
F789 57                 LD      D,A             ; Clear NMSB of quotient
F78A 5F                 LD      E,A             ; Clear LSB of quotient
F78B 321510             LD      (DIV4),A        ; Clear overflow count
F78E E5         DIVLP:  PUSH    HL              ; Save divisor
F78F C5                 PUSH    BC
F790 7D                 LD      A,L             ; Get LSB of number
F791 CD0910             CALL    DIVSUP          ; Subt' divisor from dividend
F794 DE00               SBC     A,0             ; Count for overflows
F796 3F                 CCF
F797 D2A1F7             JP      NC,RESDIV       ; Restore divisor if borrow
F79A 321510             LD      (DIV4),A        ; Re-save overflow count
F79D F1                 POP     AF              ; Scrap divisor
F79E F1                 POP     AF
F79F 37                 SCF                     ; Set carry to
F7A0 D2                 DEFB    (JP NC,nn)      ; Skip "POP BC" and "POP HL"
```

```
F7A1 C1          RESDIV: POP     BC              ; Restore divisor
F7A2 E1                  POP     HL
F7A3 79                  LD      A,C             ; Get MSB of quotient
F7A4 3C                  INC     A
F7A5 3D                  DEC     A
F7A6 1F                  RRA                     ; Bit 0 to bit 7
F7A7 FA54F6              JP      M,RONDB         ; Done - Normalise result
F7AA 17                  RLA                     ; Restore carry
F7AB 7B                  LD      A,E             ; Get LSB of quotient
F7AC 17                  RLA                     ; Double it
F7AD 5F                  LD      E,A             ; Put it back
F7AE 7A                  LD      A,D             ; Get NMSB of quotient
F7AF 17                  RLA                     ; Double it
F7B0 57                  LD      D,A             ; Put it back
F7B1 79                  LD      A,C             ; Get MSB of quotient
F7B2 17                  RLA                     ; Double it
F7B3 4F                  LD      C,A             ; Put it back
F7B4 29                  ADD     HL,HL           ; Double NMSB,LSB of divisor
F7B5 78                  LD      A,B             ; Get MSB of divisor
F7B6 17                  RLA                     ; Double it
F7B7 47                  LD      B,A             ; Put it back
F7B8 3A1510              LD      A,(DIV4)        ; Get VLSB of quotient
F7BB 17                  RLA                     ; Double it
F7BC 321510              LD      (DIV4),A        ; Put it back
F7BF 79                  LD      A,C             ; Get MSB of quotient
F7C0 B2                  OR      D               ; Merge NMSB
F7C1 B3                  OR      E               ; Merge LSB
F7C2 C28EF7              JP      NZ,DIVLP        ; Not done - Keep dividing
F7C5 E5                  PUSH    HL              ; Save divisor
F7C6 21E710              LD      HL,FPEXP        ; Point to exponent
F7C9 35                  DEC     (HL)            ; Divide by 2
F7CA E1                  POP     HL              ; Restore divisor
F7CB C28EF7              JP      NZ,DIVLP        ; Ok - Keep going
F7CE C3BCE3              JP      OVERR           ; Overflow error

F7D1 78          ADDEXP: LD      A,B             ; Get exponent of dividend
F7D2 B7                  OR      A               ; Test it
F7D3 CAF5F7              JP      Z,OVTST3        ; Zero - Result zero
F7D6 7D                  LD      A,L             ; Get add/subtract flag
F7D7 21E710              LD      HL,FPEXP        ; Point to exponent
F7DA AE                  XOR     (HL)            ; Add or subtract it
F7DB 80                  ADD     A,B             ; Add the other exponent
F7DC 47                  LD      B,A             ; Save new exponent
F7DD 1F                  RRA                     ; Test exponent for overflow
F7DE A8                  XOR     B
F7DF 78                  LD      A,B             ; Get exponent
F7E0 F2F4F7              JP      P,OVTST2        ; Positive - Test for overflow
F7E3 C680                ADD     A,80H           ; Add excess 128
F7E5 77                  LD      (HL),A          ; Save new exponent
F7E6 CA54F7              JP      Z,POPHRT        ; Zero - Result zero
F7E9 CD79F8              CALL    SIGNS           ; Set MSBs and sign of result
F7EC 77                  LD      (HL),A          ; Save new exponent
F7ED 2B                  DEC     HL              ; Point to MSB
F7EE C9                  RET
```

```
F7EF CD13F8     OVTST1: CALL    TSTSGN          ; Test sign of FPREG
F7F2 2F                 CPL                     ; Invert sign
F7F3 E1                 POP     HL              ; Clean up stack
F7F4 B7         OVTST2: OR      A               ; Test if new exponent zero
F7F5 E1         OVTST3: POP     HL              ; Clear off return address
F7F6 F233F6             JP      P,RESZER        ; Result zero
F7F9 C3BCE3             JP      OVERR           ; Overflow error

F7FC CD5FF8     MLSP10: CALL    BCDEFP          ; Move FPREG to BCDE
F7FF 78                 LD      A,B             ; Get exponent
F800 B7                 OR      A               ; Is it zero?
F801 C8                 RET     Z               ; Yes - Result is zero
F802 C602               ADD     A,2             ; Multiply by 4
F804 DABCE3             JP      C,OVERR         ; Overflow - ?OV Error
F807 47                 LD      B,A             ; Re-save exponent
F808 CDCDF5             CALL    FPADD           ; Add BCDE to FPREG (Times 5)
F80B 21E710             LD      HL,FPEXP        ; Point to exponent
F80E 34                 INC     (HL)            ; Double number (Times 10)
F80F C0                 RET     NZ              ; Ok - Return
F810 C3BCE3             JP      OVERR           ; Overflow error

F813 3AE710     TSTSGN: LD      A,(FPEXP)       ; Get sign of FPREG
F816 B7                 OR      A
F817 C8                 RET     Z               ; RETurn if number is zero
F818 3AE610             LD      A,(FPREG+2)     ; Get MSB of FPREG
F81B FE                 DEFB    (CP 2FH         ; Test sign
F81C 2F         RETREL: CPL                     ; Invert sign
F81D 17                 RLA                     ; Sign bit to carry
F81E 9F         FLGDIF: SBC     A,A             ; Carry to all bits of A
F81F C0                 RET     NZ              ; Return -1 if negative
F820 3C                 INC     A               ; Bump to +1
F821 C9                 RET                     ; Positive - Return +1

F822 CD13F8     SGN:    CALL    TSTSGN          ; Test sign of FPREG
F825 0688       FLGREL: LD      B,80H+8         ; 8 bit integer in exponent
F827 110000             LD      DE,0            ; Zero NMSB and LSB
F82A 21E710     RETINT: LD      HL,FPEXP        ; Point to exponent
F82D 4F                 LD      C,A             ; CDE = MSB,NMSB and LSB
F82E 70                 LD      (HL),B          ; Save exponent
F82F 0600               LD      B,0             ; CDE = integer to normalise
F831 23                 INC     HL              ; Point to sign of result
F832 3680               LD      (HL),80H        ; Set sign of result
F834 17                 RLA                     ; Carry = sign of integer
F835 C31BF6             JP      CONPOS          ; Set sign of result
```

```
F838 CD13F8     ABS:    CALL    TSTSGN          ; Test sign of FPREG
F83B F0                 RET     P               ; Return if positive
F83C 21E610     INVSGN: LD      HL,FPREG+2      ; Point to MSB
F83F 7E                 LD      A,(HL)          ; Get sign of mantissa
F840 EE80               XOR     80H             ; Invert sign of mantissa
F842 77                 LD      (HL),A          ; Re-save sign of mantissa
F843 C9                 RET

F844 EB         STAKFP: EX      DE,HL           ; Save code string address
F845 2AE410             LD      HL,(FPREG)      ; LSB,NLSB of FPREG
F848 E3                 EX      (SP),HL         ; Stack them,get return
F849 E5                 PUSH    HL              ; Re-save return
F84A 2AE610             LD      HL,(FPREG+2)    ; MSB and exponent of FPREG
F84D E3                 EX      (SP),HL         ; Stack them,get return
F84E E5                 PUSH    HL              ; Re-save return
F84F EB                 EX      DE,HL           ; Restore code string address
F850 C9                 RET

F851 CD62F8     PHLTFP: CALL    LOADFP          ; Number at HL to BCDE
F854 EB         FPBCDE: EX      DE,HL           ; Save code string address
F855 22E410             LD      (FPREG),HL      ; Save LSB,NLSB of number
F858 60                 LD      H,B             ; Exponent of number
F859 69                 LD      L,C             ; MSB of number
F85A 22E610             LD      (FPREG+2),HL    ; Save MSB and exponent
F85D EB                 EX      DE,HL           ; Restore code string address
F85E C9                 RET

F85F 21E410     BCDEFP: LD      HL,FPREG        ; Point to FPREG
F862 5E         LOADFP: LD      E,(HL)          ; Get LSB of number
F863 23                 INC     HL
F864 56                 LD      D,(HL)          ; Get NMSB of number
F865 23                 INC     HL
F866 4E                 LD      C,(HL)          ; Get MSB of number
F867 23                 INC     HL
F868 46                 LD      B,(HL)          ; Get exponent of number
F869 23         INCHL:  INC     HL              ; Used for conditional "INC HL"
F86A C9                 RET

F86B 11E410     FPTHL:  LD      DE,FPREG        ; Point to FPREG
F86E 0604       DETHL4: LD      B,4             ; 4 bytes to move
F870 1A         DETHLB: LD      A,(DE)          ; Get source
F871 77                 LD      (HL),A          ; Save destination
F872 13                 INC     DE              ; Next source
F873 23                 INC     HL              ; Next destination
F874 05                 DEC     B               ; Count bytes
F875 C270F8             JP      NZ,DETHLB       ; Loop if more
F878 C9                 RET
```

```
F879 21E610     SIGNS:  LD      HL,FPREG+2      ; Point to MSB of FPREG
F87C 7E                 LD      A,(HL)          ; Get MSB
F87D 07                 RLCA                    ; Old sign to carry
F87E 37                 SCF                     ; Set MSBit
F87F 1F                 RRA                     ; Set MSBit of MSB
F880 77                 LD      (HL),A          ; Save new MSB
F881 3F                 CCF                     ; Complement sign
F882 1F                 RRA                     ; Old sign to carry
F883 23                 INC     HL
F884 23                 INC     HL
F885 77                 LD      (HL),A          ; Set sign of result
F886 79                 LD      A,C             ; Get MSB
F887 07                 RLCA                    ; Old sign to carry
F888 37                 SCF                     ; Set MSBit
F889 1F                 RRA                     ; Set MSBit of MSB
F88A 4F                 LD      C,A             ; Save MSB
F88B 1F                 RRA
F88C AE                 XOR     (HL)            ; New sign of result
F88D C9                 RET

F88E 78         CMPNUM: LD      A,B             ; Get exponent of number
F88F B7                 OR      A
F890 CA13F8             JP      Z,TSTSGN        ; Zero - Test sign of FPREG
F893 211CF8             LD      HL,RETREL       ; Return relation routine
F896 E5                 PUSH    HL              ; Save for return
F897 CD13F8             CALL    TSTSGN          ; Test sign of FPREG
F89A 79                 LD      A,C             ; Get MSB of number
F89B C8                 RET     Z               ; FPREG zero - Number's MSB
F89C 21E610             LD      HL,FPREG+2      ; MSB of FPREG
F89F AE                 XOR     (HL)            ; Combine signs
F8A0 79                 LD      A,C             ; Get MSB of number
F8A1 F8                 RET     M               ; Exit if signs different
F8A2 CDA8F8             CALL    CMPFP           ; Compare FP numbers
F8A5 1F                 RRA                     ; Get carry to sign
F8A6 A9                 XOR     C               ; Combine with MSB of number
F8A7 C9                 RET

F8A8 23         CMPFP:  INC     HL              ; Point to exponent
F8A9 78                 LD      A,B             ; Get exponent
F8AA BE                 CP      (HL)            ; Compare exponents
F8AB C0                 RET     NZ              ; Different
F8AC 2B                 DEC     HL              ; Point to MBS
F8AD 79                 LD      A,C             ; Get MSB
F8AE BE                 CP      (HL)            ; Compare MSBs
F8AF C0                 RET     NZ              ; Different
F8B0 2B                 DEC     HL              ; Point to NMSB
F8B1 7A                 LD      A,D             ; Get NMSB
F8B2 BE                 CP      (HL)            ; Compare NMSBs
F8B3 C0                 RET     NZ              ; Different
F8B4 2B                 DEC     HL              ; Point to LSB
F8B5 7B                 LD      A,E             ; Get LSB
F8B6 96                 SUB     (HL)            ; Compare LSBs
F8B7 C0                 RET     NZ              ; Different
F8B8 E1                 POP     HL              ; Drop RETurn
F8B9 E1                 POP     HL              ; Drop another RETurn
F8BA C9                 RET
```

```
F8BB 47          FPINT:  LD      B,A             ; <- Move
F8BC 4F                  LD      C,A             ; <- exponent
F8BD 57                  LD      D,A             ; <- to all
F8BE 5F                  LD      E,A             ; <- bits
F8BF B7                  OR      A               ; Test exponent
F8C0 C8                  RET     Z               ; Zero - Return zero
F8C1 E5                  PUSH    HL              ; Save pointer to number
F8C2 CD5FF8              CALL    BCDEFP          ; Move FPREG to BCDE
F8C5 CD79F8              CALL    SIGNS           ; Set MSBs & sign of result
F8C8 AE                  XOR     (HL)            ; Combine with sign of FPREG
F8C9 67                  LD      H,A             ; Save combined signs
F8CA FCDFF8              CALL    M,DCBCDE        ; Negative - Decrement BCDE
F8CD 3E98                LD      A,80H+24        ; 24 bits
F8CF 90                  SUB     B               ; Bits to shift
F8D0 CD92F6              CALL    SCALE           ; Shift BCDE
F8D3 7C                  LD      A,H             ; Get combined sign
F8D4 17                  RLA                     ; Sign to carry
F8D5 DC65F6              CALL    C,FPROND        ; Negative - Round number up
F8D8 0600                LD      B,0             ; Zero exponent
F8DA DC7EF6              CALL    C,COMPL         ; If negative make positive
F8DD E1                  POP     HL              ; Restore pointer to number
F8DE C9                  RET

F8DF 1B          DCBCDE: DEC     DE              ; Decrement BCDE
F8E0 7A                  LD      A,D             ; Test LSBs
F8E1 A3                  AND     E
F8E2 3C                  INC     A
F8E3 C0                  RET     NZ              ; Exit if LSBs not FFFF
F8E4 0B                  DEC     BC              ; Decrement MSBs
F8E5 C9                  RET

F8E6 21E710      INT:    LD      HL,FPEXP        ; Point to exponent
F8E9 7E                  LD      A,(HL)          ; Get exponent
F8EA FE98                CP      80H+24          ; Integer accuracy only?
F8EC 3AE410              LD      A,(FPREG)       ; Get LSB
F8EF D0                  RET     NC              ; Yes - Already intger
F8F0 7E                  LD      A,(HL)          ; Get exponent
F8F1 CDBBF8              CALL    FPINT           ; F.P to integer
F8F4 3698                LD      (HL),80H+24     ; Save 24 bit integer
F8F6 7B                  LD      A,E             ; Get LSB of number
F8F7 F5                  PUSH    AF              ; Save LSB
F8F8 79                  LD      A,C             ; Get MSB of number
F8F9 17                  RLA                     ; Sign to carry
F8FA CD1BF6              CALL    CONPOS          ; Set sign of result
F8FD F1                  POP     AF              ; Restore LSB of number
F8FE C9                  RET
```

```
F8FF 210000     MLDEBC: LD      HL,0            ; Clear partial product
F902 78                 LD      A,B             ; Test multiplier
F903 B1                 OR      C
F904 C8                 RET     Z               ; Return zero if zero
F905 3E10               LD      A,16            ; 16 bits
F907 29         MLDBLP: ADD     HL,HL           ; Shift P.P left
F908 DA45F0             JP      C,BSERR         ; ?BS Error if overflow
F90B EB                 EX      DE,HL
F90C 29                 ADD     HL,HL           ; Shift multiplier left
F90D EB                 EX      DE,HL
F90E D215F9             JP      NC,NOMLAD       ; Bit was zero - No add
F911 09                 ADD     HL,BC           ; Add multiplicand
F912 DA45F0             JP      C,BSERR         ; ?BS Error if overflow
F915 3D         NOMLAD: DEC     A               ; Count bits
F916 C207F9             JP      NZ,MLDBLP       ; More
F919 C9                 RET

F91A FE2D       ASCTFP: CP      "-"             ; Negative?
F91C F5                 PUSH    AF              ; Save it and flags
F91D CA26F9             JP      Z,CNVNUM        ; Yes - Convert number
F920 FE2B               CP      "+"             ; Positive?
F922 CA26F9             JP      Z,CNVNUM        ; Yes - Convert number
F925 2B                 DEC     HL              ; DEC 'cos GETCHR INCs
F926 CD33F6     CNVNUM: CALL    RESZER          ; Set result to zero
F929 47                 LD      B,A             ; Digits after point counter
F92A 57                 LD      D,A             ; Sign of exponent
F92B 5F                 LD      E,A             ; Exponent of ten
F92C 2F                 CPL
F92D 4F                 LD      C,A             ; Before or after point flag
F92E CD36E8     MANLP:  CALL    GETCHR          ; Get next character
F931 DA77F9             JP      C,ADDIG         ; Digit - Add to number
F934 FE2E               CP      "."
F936 CA52F9             JP      Z,DPOINT        ; "." - Flag point
F939 FE45               CP      "E"
F93B C256F9             JP      NZ,CONEXP       ; Not "E" - Scale number
F93E CD36E8             CALL    GETCHR          ; Get next character
F941 CD70EE             CALL    SGNEXP          ; Get sign of exponent
F944 CD36E8     EXPLP:  CALL    GETCHR          ; Get next character
F947 DA99F9             JP      C,EDIGIT        ; Digit - Add to exponent
F94A 14                 INC     D               ; Is sign negative?
F94B C256F9             JP      NZ,CONEXP       ; No - Scale number
F94E AF                 XOR     A
F94F 93                 SUB     E               ; Negate exponent
F950 5F                 LD      E,A             ; And re-save it
F951 0C                 INC     C               ; Flag end of number
F952 0C         DPOINT: INC     C               ; Flag point passed
F953 CA2EF9             JP      Z,MANLP         ; Zero - Get another digit
F956 E5         CONEXP: PUSH    HL              ; Save code string address
F957 7B                 LD      A,E             ; Get exponent
F958 90                 SUB     B               ; Subtract digits after point
F959 F46FF9     SCALMI: CALL    P,SCALPL        ; Positive - Multiply number
F95C F265F9             JP      P,ENDCON        ; Positive - All done
F95F F5                 PUSH    AF              ; Save number of times to /10
F960 CD5BF7             CALL    DIV10           ; Divide by 10
F963 F1                 POP     AF              ; Restore count
F964 3C                 INC     A               ; Count divides
```

```
F965 C259F9     ENDCON: JP      NZ,SCALMI       ; More to do
F968 D1                 POP     DE              ; Restore code string address
F969 F1                 POP     AF              ; Restore sign of number
F96A CC3CF8             CALL    Z,INVSGN        ; Negative - Negate number
F96D EB                 EX      DE,HL           ; Code string address to HL
F96E C9                 RET

F96F C8         SCALPL: RET     Z               ; Exit if no scaling needed
F970 F5         MULTEN: PUSH    AF              ; Save count
F971 CDFCF7             CALL    MLSP10          ; Multiply number by 10
F974 F1                 POP     AF              ; Restore count
F975 3D                 DEC     A               ; Count multiplies
F976 C9                 RET

F977 D5         ADDIG:  PUSH    DE              ; Save sign of exponent
F978 57                 LD      D,A             ; Save digit
F979 78                 LD      A,B             ; Get digits after point
F97A 89                 ADC     A,C             ; Add one if after point
F97B 47                 LD      B,A             ; Re-save counter
F97C C5                 PUSH    BC              ; Save point flags
F97D E5                 PUSH    HL              ; Save code string address
F97E D5                 PUSH    DE              ; Save digit
F97F CDFCF7             CALL    MLSP10          ; Multiply number by 10
F982 F1                 POP     AF              ; Restore digit
F983 D630               SUB     "0"             ; Make it absolute
F985 CD8EF9             CALL    RSCALE          ; Re-scale number
F988 E1                 POP     HL              ; Restore code string address
F989 C1                 POP     BC              ; Restore point flags
F98A D1                 POP     DE              ; Restore sign of exponent
F98B C32EF9             JP      MANLP           ; Get another digit

F98E CD44F8     RSCALE: CALL    STAKFP          ; Put number on stack
F991 CD25F8             CALL    FLGREL          ; Digit to add to FPREG
F994 C1         PADD:   POP     BC              ; Restore number
F995 D1                 POP     DE
F996 C3CDF5             JP      FPADD           ; Add BCDE to FPREG and return

F999 7B         EDIGIT: LD      A,E             ; Get digit
F99A 07                 RLCA                    ; Times 2
F99B 07                 RLCA                    ; Times 4
F99C 83                 ADD     A,E             ; Times 5
F99D 07                 RLCA                    ; Times 10
F99E 86                 ADD     A,(HL)          ; Add next digit
F99F D630               SUB     "0"             ; Make it absolute
F9A1 5F                 LD      E,A             ; Save new digit
F9A2 C344F9             JP      EXPLP           ; Look for another digit
```

```
F9A5 E5          LINEIN: PUSH    HL              ; Save code string address
F9A6 2146E3              LD      HL,INMSG        ; Output " in "
F9A9 CD10F2              CALL    PRS             ; Output string at HL
F9AC E1                  POP     HL              ; Restore code string address
F9AD EB          PRNTHL: EX      DE,HL           ; Code string address to DE
F9AE AF                  XOR     A
F9AF 0698                LD      B,80H+24        ; 24 bits
F9B1 CD2AF8              CALL    RETINT          ; Return the integer
F9B4 210FF2              LD      HL,PRNUMS       ; Print number string
F9B7 E5                  PUSH    HL              ; Save for return
F9B8 21E910      NUMASC: LD      HL,PBUFF        ; Convert number to ASCII
F9BB E5                  PUSH    HL              ; Save for return
F9BC CD13F8              CALL    TSTSGN          ; Test sign of FPREG
F9BF 3620                LD      (HL)," "        ; Space at start
F9C1 F2C6F9              JP      P,SPCFST        ; Positive - Space to start
F9C4 362D                LD      (HL),"-"        ; "-" sign at start
F9C6 23          SPCFST: INC     HL              ; First byte of number
F9C7 3630                LD      (HL),"0"        ; "0" if zero
F9C9 CA7CFA              JP      Z,JSTZER        ; Return "0" if zero
F9CC E5                  PUSH    HL              ; Save buffer address
F9CD FC3CF8              CALL    M,INVSGN        ; Negate FPREG if negative
F9D0 AF                  XOR     A               ; Zero A
F9D1 F5                  PUSH    AF              ; Save it
F9D2 CD82FA              CALL    RNGTST          ; Test number is in range
F9D5 014391      SIXDIG: LD      BC,9143H        ; BCDE = 99999.9
F9D8 11F84F              LD      DE,4FF8H
F9DB CD8EF8              CALL    CMPNUM          ; Compare numbers
F9DE B7                  OR      A
F9DF E2F3F9              JP      PO,INRNG        ; > 99999.9 - Sort it out
F9E2 F1                  POP     AF              ; Restore count
F9E3 CD70F9              CALL    MULTEN          ; Multiply by ten
F9E6 F5                  PUSH    AF              ; Re-save count
F9E7 C3D5F9              JP      SIXDIG          ; Test it again

F9EA CD5BF7      GTSIXD: CALL    DIV10           ; Divide by 10
F9ED F1                  POP     AF              ; Get count
F9EE 3C                  INC     A               ; Count divides
F9EF F5                  PUSH    AF              ; Re-save count
F9F0 CD82FA              CALL    RNGTST          ; Test number is in range
F9F3 CDBBF5      INRNG:  CALL    ROUND           ; Add 0.5 to FPREG
F9F6 3C                  INC     A
F9F7 CDBBF8              CALL    FPINT           ; F.P to integer
F9FA CD54F8              CALL    FPBCDE          ; Move BCDE to FPREG
F9FD 010603              LD      BC,0306H        ; 1E+06 to 1E-03 range
FA00 F1                  POP     AF              ; Restore count
FA01 81                  ADD     A,C             ; 6 digits before point
FA02 3C                  INC     A               ; Add one
FA03 FA0FFA              JP      M,MAKNUM        ; Do it in "E" form if < 1E-02
FA06 FE08                CP      6+1+1           ; More than 999999 ?
FA08 D20FFA              JP      NC,MAKNUM       ; Yes - Do it in "E" form
FA0B 3C                  INC     A               ; Adjust for exponent
FA0C 47                  LD      B,A             ; Exponent of number
FA0D 3E02                LD      A,2             ; Make it zero after
```

```
FA0F 3D          MAKNUM: DEC     A               ; Adjust for digits to do
FA10 3D                  DEC     A
FA11 E1                  POP     HL              ; Restore buffer address
FA12 F5                  PUSH    AF              ; Save count
FA13 1195FA              LD      DE,POWERS       ; Powers of ten
FA16 05                  DEC     B               ; Count digits before point
FA17 C220FA              JP      NZ,DIGTXT       ; Not zero - Do number
FA1A 362E                LD      (HL),"."        ; Save point
FA1C 23                  INC     HL              ; Move on
FA1D 3630                LD      (HL),"0"        ; Save zero
FA1F 23                  INC     HL              ; Move on
FA20 05          DIGTXT: DEC     B               ; Count digits before point
FA21 362E                LD      (HL),"."        ; Save point in case
FA23 CC69F8              CALL    Z,INCHL         ; Last digit - move on
FA26 C5                  PUSH    BC              ; Save digits before point
FA27 E5                  PUSH    HL              ; Save buffer address
FA28 D5                  PUSH    DE              ; Save powers of ten
FA29 CD5FF8              CALL    BCDEFP          ; Move FPREG to BCDE
FA2C E1                  POP     HL              ; Powers of ten table
FA2D 062F                LD      B,"0"-1         ; ASCII "0" - 1
FA2F 04          TRYAGN: INC     B               ; Count subtractions
FA30 7B                  LD      A,E             ; Get LSB
FA31 96                  SUB     (HL)            ; Subtract LSB
FA32 5F                  LD      E,A             ; Save LSB
FA33 23                  INC     HL
FA34 7A                  LD      A,D             ; Get NMSB
FA35 9E                  SBC     A,(HL)          ; Subtract NMSB
FA36 57                  LD      D,A             ; Save NMSB
FA37 23                  INC     HL
FA38 79                  LD      A,C             ; Get MSB
FA39 9E                  SBC     A,(HL)          ; Subtract MSB
FA3A 4F                  LD      C,A             ; Save MSB
FA3B 2B                  DEC     HL              ; Point back to start
FA3C 2B                  DEC     HL
FA3D D22FFA              JP      NC,TRYAGN       ; No overflow - Try again
FA40 CD72F6              CALL    PLUCDE          ; Restore number
FA43 23                  INC     HL              ; Start of next number
FA44 CD54F8              CALL    FPBCDE          ; Move BCDE to FPREG
FA47 EB                  EX      DE,HL           ; Save point in table
FA48 E1                  POP     HL              ; Restore buffer address
FA49 70                  LD      (HL),B          ; Save digit in buffer
FA4A 23                  INC     HL              ; And move on
FA4B C1                  POP     BC              ; Restore digit count
FA4C 0D                  DEC     C               ; Count digits
FA4D C220FA              JP      NZ,DIGTXT       ; More - Do them
FA50 05                  DEC     B               ; Any decimal part?
FA51 CA60FA              JP      Z,DOEBIT        ; No - Do "E" bit
FA54 2B          SUPTLZ: DEC     HL              ; Move back through buffer
FA55 7E                  LD      A,(HL)          ; Get character
FA56 FE30                CP      "0"             ; "0" character?
FA58 CA54FA              JP      Z,SUPTLZ        ; Yes - Look back for more
FA5B FE2E                CP      "."             ; A decimal point?
FA5D C469F8              CALL    NZ,INCHL        ; Move back over digit
```

```
FA60 F1          DOEBIT: POP     AF              ; Get "E" flag
FA61 CA7FFA              JP      Z,NOENED        ; No "E" needed - End buffer
FA64 3645                LD      (HL),"E"        ; Put "E" in buffer
FA66 23                  INC     HL              ; And move on
FA67 362B                LD      (HL),"+"        ; Put "+" in buffer
FA69 F270FA              JP      P,OUTEXP        ; Positive - Output exponent
FA6C 362D                LD      (HL),"-"        ; Put "-" in buffer
FA6E 2F                  CPL                     ; Negate exponent
FA6F 3C                  INC     A
FA70 062F        OUTEXP: LD      B,"0"-1         ; ASCII "0" - 1
FA72 04          EXPTEN: INC     B               ; Count subtractions
FA73 D60A                SUB     10              ; Tens digit
FA75 D272FA              JP      NC,EXPTEN       ; More to do
FA78 C63A                ADD     A,10+"0"        ; Restore and make ASCII
FA7A 23                  INC     HL              ; Move on
FA7B 70                  LD      (HL),B          ; Save MSB of exponent
FA7C 23          JSTZER: INC     HL
FA7D 77                  LD      (HL),A          ; Save LSB of exponent
FA7E 23                  INC     HL
FA7F 71          NOENED: LD      (HL),C          ; Mark end of buffer
FA80 E1                  POP     HL              ; Restore code string address
FA81 C9                  RET

FA82 017494      RNGTST: LD      BC,9474H        ; BCDE = 999999.
FA85 11F723              LD      DE,23F7H
FA88 CD8EF8              CALL    CMPNUM          ; Compare numbers
FA8B B7                  OR      A
FA8C E1                  POP     HL              ; Return address to HL
FA8D E2EAF9              JP      PO,GTSIXD       ; Too big - Divide by ten
FA90 E9                  JP      (HL)            ; Otherwise return to caller

FA91 00000080    HALF:   DEFB    00H,00H,00H,80H ; 0.5

FA95 A08601      POWERS: DEFB    0A0H,086H,001H  ;100000
FA98 102700              DEFB    010H,027H,000H  ; 10000
FA9B E80300              DEFB    0E8H,003H,000H  ;  1000
FA9E 640000              DEFB    064H,000H,000H  ;   100
FAA1 0A0000              DEFB    00AH,000H,000H  ;    10
FAA4 010000              DEFB    001H,000H,000H  ;     1

FAA7 213CF8      NEGAFT: LD      HL,INVSGN       ; Negate result
FAAA E3                  EX      (SP),HL         ; To be done after caller
FAAB E9                  JP      (HL)            ; Return to caller

FAAC CD44F8      SQR:    CALL    STAKFP          ; Put value on stack
FAAF 2191FA              LD      HL,HALF         ; Set power to 1/2
FAB2 CD51F8              CALL    PHLTFP          ; Move 1/2 to FPREG
```

```
FAB5 C1          POWER:  POP     BC              ; Get base
FAB6 D1                  POP     DE
FAB7 CD13F8              CALL    TSTSGN          ; Test sign of power
FABA 78                  LD      A,B             ; Get exponent of base
FABB CAFAFA              JP      Z,EXP           ; Make result 1 if zero
FABE F2C5FA              JP      P,POWER1        ; Positive base - Ok
FAC1 B7                  OR      A               ; Zero to negative power?
FAC2 CAB0E3              JP      Z,DZERR         ; Yes - ?/0 Error
FAC5 B7          POWER1: OR      A               ; Base zero?
FAC6 CA34F6              JP      Z,SAVEXP        ; Yes - Return zero
FAC9 D5                  PUSH    DE              ; Save base
FACA C5                  PUSH    BC
FACB 79                  LD      A,C             ; Get MSB of base
FACC F67F                OR      01111111B       ; Get sign status
FACE CD5FF8              CALL    BCDEFP          ; Move power to BCDE
FAD1 F2E2FA              JP      P,POWER2        ; Positive base - Ok
FAD4 D5                  PUSH    DE              ; Save power
FAD5 C5                  PUSH    BC
FAD6 CDE6F8              CALL    INT             ; Get integer of power
FAD9 C1                  POP     BC              ; Restore power
FADA D1                  POP     DE
FADB F5                  PUSH    AF              ; MSB of base
FADC CD8EF8              CALL    CMPNUM          ; Power an integer?
FADF E1                  POP     HL              ; Restore MSB of base
FAE0 7C                  LD      A,H             ; but don't affect flags
FAE1 1F                  RRA                     ; Exponent odd or even?
FAE2 E1          POWER2: POP     HL              ; Restore MSB and exponent
FAE3 22E610              LD      (FPREG+2),HL    ; Save base in FPREG
FAE6 E1                  POP     HL              ; LSBs of base
FAE7 22E410              LD      (FPREG),HL      ; Save in FPREG
FAEA DCA7FA              CALL    C,NEGAFT        ; Odd power - Negate result
FAED CC3CF8              CALL    Z,INVSGN        ; Negative base - Negate it
FAF0 D5                  PUSH    DE              ; Save power
FAF1 C5                  PUSH    BC
FAF2 CDC7F6              CALL    LOG             ; Get LOG of base
FAF5 C1                  POP     BC              ; Restore power
FAF6 D1                  POP     DE
FAF7 CD08F7              CALL    FPMULT          ; Multiply LOG by power
```

```
FAFA CD44F8     EXP:    CALL    STAKFP          ; Put value on stack
FAFD 013881             LD      BC,08138H       ; BCDE = 1/Ln(2)
FB00 113BAA             LD      DE,0AA3BH
FB03 CD08F7             CALL    FPMULT          ; Multiply value by 1/LN(2)
FB06 3AE710             LD      A,(FPEXP)       ; Get exponent
FB09 FE88               CP      80H+8           ; Is it in range?
FB0B D2EFF7             JP      NC,OVTST1       ; No - Test for overflow
FB0E CDE6F8             CALL    INT             ; Get INT of FPREG
FB11 C680               ADD     A,80H           ; For excess 128
FB13 C602               ADD     A,2             ; Exponent > 126?
FB15 DAEFF7             JP      C,OVTST1        ; Yes - Test for overflow
FB18 F5                 PUSH    AF              ; Save scaling factor
FB19 21B6F6             LD      HL,UNITY        ; Point to 1.
FB1C CDBEF5             CALL    ADDPHL          ; Add 1 to FPREG
FB1F CDFFF6             CALL    MULLN2          ; Multiply by LN(2)
FB22 F1                 POP     AF              ; Restore scaling factor
FB23 C1                 POP     BC              ; Restore exponent
FB24 D1                 POP     DE
FB25 F5                 PUSH    AF              ; Save scaling factor
FB26 CDCAF5             CALL    SUBCDE          ; Subtract exponent from FPREG
FB29 CD3CF8             CALL    INVSGN          ; Negate result
FB2C 213AFB             LD      HL,EXPTAB       ; Coefficient table
FB2F CD6AFB             CALL    SMSER1          ; Sum the series
FB32 110000             LD      DE,0            ; Zero LSBs
FB35 C1                 POP     BC              ; Scaling factor
FB36 4A                 LD      C,D             ; Zero MSB
FB37 C308F7             JP      FPMULT          ; Scale result to correct value

FB3A 08         EXPTAB: DEFB    8                       ; Table used by EXP
FB3B 402E9474           DEFB    040H,02EH,094H,074H     ; -1/7! (-1/5040)
FB3F 704F2E77           DEFB    070H,04FH,02EH,077H     ;  1/6! ( 1/720)
FB43 6E02887A           DEFB    06EH,002H,088H,07AH     ; -1/5! (-1/120)
FB47 E6A02A7C           DEFB    0E6H,0A0H,02AH,07CH     ;  1/4! ( 1/24)
FB4B 50AAAA7E           DEFB    050H,0AAH,0AAH,07EH     ; -1/3! (-1/6)
FB4F FFFF7F7F           DEFB    0FFH,0FFH,07FH,07FH     ;  1/2! ( 1/2)
FB53 00008081           DEFB    000H,000H,080H,081H     ; -1/1! (-1/1)
FB57 00000081           DEFB    000H,000H,000H,081H     ;  1/0! ( 1/1)
```

```
FB5B CD44F8     SUMSER: CALL    STAKFP          ; Put FPREG on stack
FB5E 1106F7             LD      DE,MULT         ; Multiply by "X"
FB61 D5                 PUSH    DE              ; To be done after
FB62 E5                 PUSH    HL              ; Save address of table
FB63 CD5FF8             CALL    BCDEFP          ; Move FPREG to BCDE
FB66 CD08F7             CALL    FPMULT          ; Square the value
FB69 E1                 POP     HL              ; Restore address of table
FB6A CD44F8     SMSER1: CALL    STAKFP          ; Put value on stack
FB6D 7E                 LD      A,(HL)          ; Get number of coefficients
FB6E 23                 INC     HL              ; Point to start of table
FB6F CD51F8             CALL    PHLTFP          ; Move coefficient to FPREG
FB72 06                 DEFB    (LD B,n)        ; Skip "POP AF"
FB73 F1         SUMLP:  POP     AF              ; Restore count
FB74 C1                 POP     BC              ; Restore number
FB75 D1                 POP     DE
FB76 3D                 DEC     A               ; Cont coefficients
FB77 C8                 RET     Z               ; All done
FB78 D5                 PUSH    DE              ; Save number
FB79 C5                 PUSH    BC
FB7A F5                 PUSH    AF              ; Save count
FB7B E5                 PUSH    HL              ; Save address in table
FB7C CD08F7             CALL    FPMULT          ; Multiply FPREG by BCDE
FB7F E1                 POP     HL              ; Restore address in table
FB80 CD62F8             CALL    LOADFP          ; Number at HL to BCDE
FB83 E5                 PUSH    HL              ; Save address in table
FB84 CDCDF5             CALL    FPADD           ; Add coefficient to FPREG
FB87 E1                 POP     HL              ; Restore address in table
FB88 C373FB             JP      SUMLP           ; More coefficients
```

LAST

PART

NEXT

ISSUE!

# NASCOM

# ROM

# BASIC

# DIS–ASSEMBLED

# PART 7

# BY CARL LLOYD–PARKER

```
FB8B CD13F8     RND:    CALL    TSTSGN          ; Test sign of FPREG
FB8E 211910             LD      HL,SEED+2       ; Random number seed
FB91 FAECFB             JP      M,RESEED        ; Negative - Re-seed
FB94 213A10             LD      HL,LSTRND       ; Last random number
FB97 CD51F8             CALL    PHLTFP          ; Move last RND to FPREG
FB9A 211910             LD      HL,SEED+2       ; Random number seed
FB9D C8                 RET     Z               ; Return if RND(0)
FB9E 86                 ADD     A,(HL)          ; Add (SEED+2)
FB9F E607               AND     00000111B       ; 0 to 7
FBA1 0600               LD      B,0
FBA3 77                 LD      (HL),A          ; Re-save seed
FBA4 23                 INC     HL              ; Move to coefficient table
FBA5 87                 ADD     A,A             ; 4 bytes
FBA6 87                 ADD     A,A             ; per entry
FBA7 4F                 LD      C,A             ; BC = Offset into table
FBA8 09                 ADD     HL,BC           ; Point to coefficient
FBA9 CD62F8             CALL    LOADFP          ; Coefficient to BCDE
FBAC CD08F7             CALL    FPMULT          ; Multiply FPREG by coefficient
FBAF 3A1810             LD      A,(SEED+1)      ; Get (SEED+1)
FBB2 3C                 INC     A               ; Add 1
FBB3 E603               AND     00000011B       ; 0 to 3
FBB5 0600               LD      B,0
FBB7 FE01               CP      1               ; Is it zero?
FBB9 88                 ADC     A,B             ; Yes - Make it 1
FBBA 321810             LD      (SEED+1),A      ; Re-save seed
FBBD 21F0FB             LD      HL,RNDTAB-4     ; Addition table
FBC0 87                 ADD     A,A             ; 4 bytes
FBC1 87                 ADD     A,A             ; per entry
FBC2 4F                 LD      C,A             ; BC = Offset into table
FBC3 09                 ADD     HL,BC           ; Point to value
FBC4 CDBEF5             CALL    ADDPHL          ; Add value to FPREG
FBC7 CD5FF8     RND1:   CALL    BCDEFP          ; Move FPREG to BCDE
FBCA 7B                 LD      A,E             ; Get LSB
FBCB 59                 LD      E,C             ; LSB = MSB
FBCC EE4F               XOR     01001111B       ; Fiddle around
FBCE 4F                 LD      C,A             ; New MSB
FBCF 3680               LD      (HL),80H        ; Set exponent
FBD1 2B                 DEC     HL              ; Point to MSB
FBD2 46                 LD      B,(HL)          ; Get MSB
FBD3 3680               LD      (HL),80H        ; Make value -0.5
FBD5 211710             LD      HL,SEED         ; Random number seed
FBD8 34                 INC     (HL)            ; Count seed
FBD9 7E                 LD      A,(HL)          ; Get seed
FBDA D6AB               SUB     171             ; Do it modulo 171
FBDC C2E3FB             JP      NZ,RND2         ; Non-zero - Ok
FBDF 77                 LD      (HL),A          ; Zero seed
FBE0 0C                 INC     C               ; Fillde about
FBE1 15                 DEC     D               ; with the
FBE2 1C                 INC     E               ; number
FBE3 CD1EF6     RND2:   CALL    BNORM           ; Normalise number
FBE6 213A10             LD      HL,LSTRND       ; Save random number
FBE9 C36BF8             JP      FPTHL           ; Move FPREG to last and return
```

```
FBEC 77          RESEED: LD      (HL),A          ; Re-seed random numbers
FBED 2B                  DEC     HL
FBEE 77                  LD      (HL),A
FBEF 2B                  DEC     HL
FBF0 77                  LD      (HL),A
FBF1 C3C7FB              JP      RND1            ; Return RND seed

FBF4 68B14668    RNDTAB: DEFB    068H,0B1H,046H,068H     ; Table used by RND
FBF8 99E99269            DEFB    099H,0E9H,092H,069H
FBFC 10D17568            DEFB    010H,0D1H,075H,068H

FC00 214AFC      COS:    LD      HL,HALFPI       ; Point to PI/2
FC03 CDBEF5              CALL    ADDPHL          ; Add it to FPREG
FC06 CD44F8      SIN:    CALL    STAKFP          ; Put angle on stack
FC09 014983              LD      BC,8349H        ; BCDE = 2 PI
FC0C 11DB0F              LD      DE,0FDBH
FC0F CD54F8              CALL    FPBCDE          ; Move 2 PI to FPREG
FC12 C1                  POP     BC              ; Restore angle
FC13 D1                  POP     DE
FC14 CD69F7              CALL    DVBCDE          ; Divide angle by 2 PI
FC17 CD44F8              CALL    STAKFP          ; Put it on stack
FC1A CDE6F8              CALL    INT             ; Get INT of result
FC1D C1                  POP     BC              ; Restore number
FC1E D1                  POP     DE
FC1F CDCAF5              CALL    SUBCDE          ; Make it 0 <= value < 1
FC22 214EFC              LD      HL,QUARTR       ; Point to 0.25
FC25 CDC4F5              CALL    SUBPHL          ; Subtract value from 0.25
FC28 CD13F8              CALL    TSTSGN          ; Test sign of value
FC2B 37                  SCF                     ; Flag positive
FC2C F236FC              JP      P,SIN1          ; Positive - Ok
FC2F CDBBF5              CALL    ROUND           ; Add 0.5 to value
FC32 CD13F8              CALL    TSTSGN          ; Test sign of value
FC35 B7                  OR      A               ; Flag negative
FC36 F5          SIN1:   PUSH    AF              ; Save sign
FC37 F43CF8              CALL    P,INVSGN        ; Negate value if positive
FC3A 214EFC              LD      HL,QUARTR       ; Point to 0.25
FC3D CDBEF5              CALL    ADDPHL          ; Add 0.25 to value
FC40 F1                  POP     AF              ; Restore sign
FC41 D43CF8              CALL    NC,INVSGN       ; Negative - Make positive
FC44 2152FC              LD      HL,SINTAB       ; Coefficient table
FC47 C35BFB              JP      SUMSER          ; Evaluate sum of series

FC4A DB0F4981    HALFPI: DEFB    0DBH,00FH,049H,081H     ; 1.5708 (PI/2)

FC4E 0000007F    QUARTR: DEFB    000H,000H,000H,07FH     ; 0.25

FC52 05          SINTAB: DEFB    5                       ; Table used by SIN
FC53 BAD71E86            DEFB    0BAH,0D7H,01EH,086H     ; 39.711
FC57 64269987            DEFB    064H,026H,099H,087H     ;-76.575
FC5B 58342387            DEFB    058H,034H,023H,087H     ; 81.602
FC5F E05DA586            DEFB    0E0H,05DH,0A5H,086H     ;-41.342
FC63 DA0F4983            DEFB    0DAH,00FH,049H,083H     ;  6.2832
```

```
FC67 CD44F8     TAN:    CALL    STAKFP          ; Put angle on stack
FC6A CD06FC             CALL    SIN             ; Get SIN of angle
FC6D C1                 POP     BC              ; Restore angle
FC6E E1                 POP     HL
FC6F CD44F8             CALL    STAKFP          ; Save SIN of angle
FC72 EB                 EX      DE,HL           ; BCDE = Angle
FC73 CD54F8             CALL    FPBCDE          ; Angle to FPREG
FC76 CD00FC             CALL    COS             ; Get COS of angle
FC79 C367F7             JP      DIV             ; TAN = SIN / COS

FC7C CD13F8     ATN:    CALL    TSTSGN          ; Test sign of value
FC7F FCA7FA             CALL    M,NEGAFT        ; Negate result after if -ve
FC82 FC3CF8             CALL    M,INVSGN        ; Negate value if -ve
FC85 3AE710             LD      A,(FPEXP)       ; Get exponent
FC88 FE81               CP      81H             ; Number less than 1?
FC8A DA99FC             JP      C,ATN1          ; Yes - Get arc tangnt
FC8D 010081             LD      BC,8100H        ; BCDE = 1
FC90 51                 LD      D,C
FC91 59                 LD      E,C
FC92 CD69F7             CALL    DVBCDE          ; Get reciprocal of number
FC95 21C4F5             LD      HL,SUBPHL       ; Sub angle from PI/2
FC98 E5                 PUSH    HL              ; Save for angle > 1
FC99 21A3FC     ATN1:   LD      HL,ATNTAB       ; Coefficient table
FC9C CD5BFB             CALL    SUMSER          ; Evaluate sum of series
FC9F 214AFC             LD      HL,HALFPI       ; PI/2 - angle in case > 1
FCA2 C9                 RET                     ; Number > 1 - Sub from PI/2

FCA3 09         ATNTAB: DEFB    9                       ; Table used by ATN
FCA4 4AD73B78           DEFB    04AH,0D7H,03BH,078H     ; 1/17
FCA8 026E847B           DEFB    002H,06EH,084H,07BH     ;-1/15
FCAC FEC12F7C           DEFB    0FEH,0C1H,02FH,07CH     ; 1/13
FCB0 74319A7D           DEFB    074H,031H,09AH,07DH     ;-1/11
FCB4 843D5A7D           DEFB    084H,03DH,05AH,07DH     ; 1/9
FCB8 C87F917E           DEFB    0C8H,07FH,091H,07EH     ;-1/7
FCBC E4BB4C7E           DEFB    0E4H,0BBH,04CH,07EH     ; 1/5
FCC0 6CAAAA7F           DEFB    06CH,0AAH,0AAH,07FH     ;-1/3
FCC4 00000081           DEFB    000H,000H,000H,081H     ; 1/1
```

```
FCC8 CD39FE     CASFFW: CALL    FLPLED          ; Turn on cassette
FCCB 0600               LD      B,0             ; Set 1 second delay
FCCD CD9BFD     DELAYB: CALL    DELAY           ; Wait a bit
FCD0 05                 DEC     B               ; Count
FCD1 C2CDFC             JP      NZ,DELAYB       ; More delay needed
FCD4 C9                 RET

FCD5 C339FE     CASFF:  JP      FLPLED          ; Flip tape LED

FCD8 C9         ARET:   RET                     ; A RETurn instruction

FCD9 E5C5D5F5   CONMON: PUSH    HL,BC,DE,AF     ; Output character to screen
FCDD CD6DFE             CALL    MONTST          ; See if NAS-SYS
FCE0 C2FBFC             JP      NZ,NASOUT       ; NAS-SYS - Output ASCII
FCE3 F1                 POP     AF              ; Get character
FCE4 F5                 PUSH    AF              ; And re-save
FCE5 FE0A               CP      LF              ; ASCII Line feed?
FCE7 CA00FD             JP      Z,IGCHR         ; Yes - Ignore it
FCEA FE08               CP      BKSP            ; ASCII back space?
FCEC C2F1FC             JP      NZ,CONOT1       ; No - Test for CR
FCEF 3E1D               LD      A,TBS           ; NASBUG back space
FCF1 FE0D       CONOT1: CP      CR              ; ASCII CR?
FCF3 C2FDFC             JP      NZ,OUTCHR       ; No - Output character
FCF6 3E1F               LD      A,TCR           ; NASBUG CR
FCF8 C3FDFC             JP      OUTCHR          ; Output it

FCFB F1         NASOUT: POP     AF              ; Get character
FCFC F5                 PUSH    AF              ; And re-save
FCFD CD45FE     OUTCHR: CALL    MONOUT          ; Output it
FD00 F1D1C1E1   IGCHR:  POP     AF,DE,BC,HL     ; Restore character
FD04 C9                 RET

FD05 E5C5D5     GETINP: PUSH    HL,BC,DE        ; Get an input character
FD08 CD6DFE             CALL    MONTST          ; See if NAS-SYS
FD0B CA13FD             JP      Z,GETTIN        ; "T" monitor - Get input
FD0E DF7B               SCAL    BLINK
FD10 C319FD             JP      CONVIN          ; Convert to ASCII

FD13 CD4D0C     GETTIN: CALL    TIN             ; "T" input a character
FD16 D213FD             JP      NC,GETTIN       ; No input - wait
FD19 FE1D       CONVIN: CP      TBS             ; NASBUG back space?
FD1B C220FD             JP      NZ,CNVIN1       ; No - Test for break
FD1E 3E08               LD      A,BKSP          ; ASCII back space
FD20 FE1C       CNVIN1: CP      TBRK            ; NASBUG break?
FD22 C227FD             JP      NZ,CNVIN2       ; No - Test for control Z
FD25 3E03               LD      A,CTRLC         ; Control C
FD27 FE1A       CNVIN2: CP      CTRLZ           ; ^Z?
FD29 C22EFD             JP      NZ,CNVIN3       ; No - Test for escape
FD2C 3E7F               LD      A,DEL           ; Delete
FD2E FE1B       CNVIN3: CP      ESC             ; "ESC" ?
FD30 C235FD             JP      NZ,CNVIN4       ; No - Test for CR
FD33 3E03               LD      A,CTRLC         ; Control C
FD35 FE1F       CNVIN4: CP      TCR             ; NASBUG CR?
FD37 C23CFD             JP      NZ,CNVIN5       ; No - Return character
FD3A 3E0D               LD      A,CR            ; ASCII CR
FD3C D1C1E1     CNVIN5: POP     DE,BC,HL
FD3F C9                 RET
```

```
FD40 AF          CHKBRK: XOR     A               ; Check for break
FD41 CD70FD              CALL    SFTENT          ; Test for shift/enter
FD44 CA50FD              JP      Z,TBRK2         ; Yes - Test for second break
FD47 3A4D10              LD      A,(BRKFLG)      ; Get break flag
FD4A B7                  OR      A               ; Break flag set?
FD4B C250FD              JP      NZ,TBRK2        ; Yes - Test for second break
FD4E AF                  XOR     A               ; Flag no break
FD4F C9                  RET

FD50 CD53FE      TBRK2:  CALL    BREAK2          ; Second break?
FD53 3EFF                LD      A,-1            ; Flag break
FD55 C9                  RET

FD56 DB02        GUART:  IN      A,(UARTS)       ; Get UART status
FD58 17                  RLA                     ; Any data ready?
FD59 D256FD              JP      NC,GUART        ; No - wait until there is
FD5C DB01                IN      A,(UARTD)       ; Get data from UART
FD5E C9                  RET

FD5F D301        UARTOT: OUT     (UARTD),A       ; Send data to UART
FD61 DB02        URTOLP: IN      A,(UARTS)       ; Get status
FD63 87                  ADD     A,A             ; Byte sent?
FD64 F8                  RET     M               ; Yes - Return
FD65 C361FD              JP      URTOLP          ; Keep waiting

FD68 F5          SUART:  PUSH    AF              ; Save A
FD69 CD5FFD              CALL    UARTOT          ; Send it to UART
FD6C F1                  POP     AF              ; Restore A
FD6D C9                  RET

FD6E 00                  NOP
FD6F 00                  NOP

FD70 E5          SFTENT: PUSH    HL              ; Test for Shift Enter from KBD
FD71 3E02                LD      A,00000010B     ; Reset KBD counter mask
FD73 21000C              LD      HL,PORT0        ; Get old contents
FD76 AE                  XOR     (HL)            ; Toggle bit
FD77 D300                OUT     (0),A           ; Reset KBD counter
FD79 EE01                XOR     00000001B       ; Toggle bit
FD7B D300                OUT     (0),A           ; Next row
FD7D EE02                XOR     00000010B
FD7F D300                OUT     (0),A           ; Clear "clear" strobe
FD81 7E                  LD      A,(HL)          ; Get old value
FD82 D300                OUT     (0),A           ; Original contents
FD84 19                  ADD     HL,DE           ; ?? WHAT ??
FD85 E1                  POP     HL              ; Restore HL
FD86 DB00                IN      A,(0)           ; Read in row
FD88 E612                AND     00010010B       ; Mask SHIFT and ENTER
FD8A C9                  RET

FD8B CD6DFE      CLS:    CALL    MONTST          ; See if NAS-SYS
FD8E CA96FD              JP      Z,TCLS          ; "T" CLS
FD91 3E0C                LD      A,CS            ; ASCII Clear screen
FD93 C3D9FC              JP      CONMON          ; Output character

FD96 3E1E        TCLS:   LD      A,TCS           ; NASBUG Clear screen
FD98 C3D9FC              JP      CONMON          ; Output character
```

```
FD9B AF          DELAY:  XOR     A               ; Delay routine
FD9C F5          DELAY1: PUSH    AF              ; PUSHes and POPs delay
FD9D F1                  POP     AF
FD9E F5                  PUSH    AF
FD9F F1                  POP     AF
FDA0 3D                  DEC     A               ; Count delays
FDA1 C29CFD              JP      NZ,DELAY1       ; More delay
FDA4 C9                  RET

FDA5 CD84F4      WIDTH:  CALL    GETINT          ; Get integer 0-255
FDA8 7B                  LD      A,E             ; Width to A
FDA9 324210              LD      (LWIDTH),A      ; Set width
FDAC C9                  RET

FDAD CD41ED      LINES:  CALL    GETNUM          ; Get a number
FDB0 CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
FDB3 ED534610            LD      (LINESC),DE     ; Set lines counter
FDB7 ED534810            LD      (LINESN),DE     ; Set lines number
FDBB C9                  RET

FDBC CD8BE9      DEEK:   CALL    DEINT           ; Get integer -32768 to 32767
FDBF D5                  PUSH    DE              ; Save number
FDC0 E1                  POP     HL              ; Number to HL
FDC1 46                  LD      B,(HL)          ; Get LSB of contents
FDC2 23                  INC     HL
FDC3 7E                  LD      A,(HL)          ; Get MSB of contents
FDC4 C3F2F0              JP      ABPASS          ; Return integer AB

FDC7 CD41ED      DOKE:   CALL    GETNUM          ; Get a number
FDCA CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
FDCD D5                  PUSH    DE              ; Save address
FDCE CD90E6              CALL    CHKSYN          ; Make sure "," follows
FDD1 2C                  DEFB    ","
FDD2 CD41ED              CALL    GETNUM          ; Get a number
FDD5 CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
FDD8 E3                  EX      (SP),HL         ; Save value,get address
FDD9 73                  LD      (HL),E          ; Save LSB of value
FDDA 23                  INC     HL
FDDB 72                  LD      (HL),D          ; Save MSB of value
FDDC E1                  POP     HL              ; Restore code string address
FDDD C9                  RET

FDDE F3          JJUMP1: DI                      ; Disable interrupts
FDDF DD21FFFF            LD      IX,-1           ; Flag cold start
FDE3 C312E0              JP      CSTART          ; Go and initialise
```

```
FDE6 CD84F4     SCREEN: CALL    GETINT          ; Get integer 0 to 255
FDE9 F5                 PUSH    AF              ; Save column
FDEA CD90E6             CALL    CHKSYN          ; Make sure "," follows
FDED 2C                 DEFB    ","
FDEE CD84F4             CALL    GETINT          ; Get integer 0 to 255
FDF1 C1                 POP     BC              ; Column to B
FDF2 E5                 PUSH    HL              ; Save code string address
FDF3 C5                 PUSH    BC              ; Save column
FDF4 CD11FE             CALL    SCRADR          ; Calculate screen address
FDF7 E5                 PUSH    HL              ; Save screen address
FDF8 CD6DFE             CALL    MONTST          ; See if NAS-SYS
FDFB CA04FE             JP      Z,TMNCUR        ; "T" monitor - "T" cursor
FDFE E1                 POP     HL              ; Restore screen address
FDFF 22290C             LD      (CURSOR),HL     ; Set new cursor position
FE02 E1                 POP     HL              ; Rstore code string address
FE03 C9                 RET

FE04 2A180C     TMNCUR: LD      HL,(TCUR)       ; Get address or cursor
FE07 3620               LD      (HL)," "        ; Remove cursor
FE09 E1                 POP     HL              ; Get new cursor address
FE0A 22180C             LD      (TCUR),HL       ; Set new cursor
FE0D 365F               LD      (HL),"_"        ; Put it on screen
FE0F E1                 POP     HL              ; Restore code string address
FE10 C9                 RET

FE11 21C907     SCRADR: LD      HL,VDU+10-65    ; SCREEN VDU address (0,0)
FE14 0600               LD      B,0
FE16 4F                 LD      C,A             ; Line to BC
FE17 B7                 OR      A               ; Test it
FE18 CAA0E9             JP      Z,FCERR         ; Zero - ?FC Error
FE1B FE11               CP      16+1            ; 16 lines
FE1D F2A0E9             JP      P,FCERR         ; > 16 - ?FC Error
FE20 D1                 POP     DE              ; RETurn address
FE21 F1                 POP     AF              ; Get column
FE22 D5                 PUSH    DE              ; Re-save RETurn
FE23 1600               LD      D,0
FE25 5F                 LD      E,A             ; Column to DE
FE26 B7                 OR      A               ; Test it
FE27 CAA0E9             JP      Z,FCERR         ; Zero - ?FC Error
FE2A FE31               CP      48+1            ; 48 characters per line
FE2C F2A0E9             JP      P,FCERR         ; > 48 - ?FC Error
FE2F 19                 ADD     HL,DE           ; Add column to address
FE30 1600               LD      D,0
FE32 59                 LD      E,C             ; Line to DE
FE33 0640               LD      B,64            ; 64 Bytes per line
FE35 19         ADD64X: ADD     HL,DE           ; Add line
FE36 10FD               DJNZ    ADD64X          ; SIXTY FOUR TIMES!!!
FE38 C9                 RET

FE39 CD6DFE     FLPLED: CALL    MONTST          ; See if NAS-SYS
FE3C CA42FE             JP      Z,TMFLP         ; "T" MFLP
FE3F DF5F               SCAL    MFLP
FE41 C9                 RET

FE42 C35100     TMFLP:  JP      MFLP            ; Flip drive LED
```

```
FE45 F5          MONOUT: PUSH    AF              ; Save character
FE46 CD6DFE              CALL    MONTST          ; See if NAS-SYS
FE49 CA4FFE              JP      Z,TMNOUT        ; "T" output
FE4C F1                  POP     AF              ; Restore character
FE4D F7                  ROUT                    ; Output it
FE4E C9                  RET

FE4F F1          TMNOUT: POP     AF              ; Restore character
FE50 C34A0C              JP      TOUT            ; "T" output

FE53 3A4D10      BREAK2: LD      A,(BRKFLG)      ; Break flag set?
FE56 C265FE              JP      NZ,RETCTC       ; Yes - Return ^C
FE59 CD6DFE              CALL    MONTST          ; See if NAS-SYS
FE5C CA62FE              JP      Z,TCHINP        ; Get "T" character input
FE5F DF62                SCAL    RIN             ; Scan for a character
FE61 C9                  RET

FE62 C34D0C      TCHINP: JP      TIN             ; "T" input a character

FE65 3E00        RETCTC: LD      A,0             ; Clear Break flag
FE67 324D10              LD      (BRKFLG),A
FE6A 3E03                LD      A,CTRLC         ; Return ^C
FE6C C9                  RET

FE6D 3A0100      MONTST: LD      A,(MONSTT+1)    ; "T" monitor or NAS-SYS?
FE70 FE33                CP      33H             ; 31 00 10 / 31 33 0C
FE72 C9                  RET

FE73 CD39FE      SAVE:   CALL    FLPLED          ; Flip tape LED
FE76 CD6DFE              CALL    MONTST          ; See if NAS-SYS
FE79 CA7FFE              JP      Z,TSAVE         ; "T" save
FE7C DF57                SCAL    WRITE           ; Save program
FE7E C9                  RET

FE7F 3A8D00      TSAVE:  LD      A,(MONTYP)      ; "T2" or "T4" (FLAGS!!!)
FE82 CA0004              JP      Z,T4WR          ; T4 Write
FE85 C3D103              JP      T2DUMP          ; T2 Dump

FE88 CD39FE      MONLD:  CALL    FLPLED          ; Flip tape LED
FE8B CD6DFE              CALL    MONTST          ; See if NAS-SYS
FE8E CA99FE              JP      Z,TLOAD         ; "T" load
FE91 3E52                LD      A,"R"           ; Set READ
FE93 322B0C              LD      (ARGN),A
FE96 DF52                SCAL    READ            ; Load program
FE98 C9                  RET

FE99 3A8D00      TLOAD:  LD      A,(MONTYP)      ; "T2" or "T4" (FLAGS!!!)
FE9C CA0C07              JP      Z,T4READ        ; T4 Read
FE9F C3D103              JP      T2DUMP          ; T2 Dump ??????????

FEA2 CD6DFE      MONITR: CALL    MONTST          ; See if NAS-SYS
FEA5 CA0000              JP      Z,MONSTT        ; Jump to zero if "T"
FEA8 DF5B                SCAL    MRET            ; Return to NAS-SYS
```

```
FEAA CD39FE     MONVE:  CALL    FLPLED          ; Flip tape LED
FEAD CD6DFE             CALL    MONTST          ; See if NAS-SYS
FEB0 CAA0E9             JP      Z,FCERR         ; Verify not available on "T"
FEB3 3E56               LD      A,"V"           ; Set VERIFY
FEB5 322B0C             LD      (ARGN),A
FEB8 DF56               SCAL    VERIFY          ; Verify tape
FEBA C9                 RET

FEBB 3E00       INITST: LD      A,0             ; Clear break flag
FEBD 324D10             LD      (BRKFLG),A
FEC0 CD6DFE             CALL    MONTST          ; See if NAS-SYS
FEC3 CA19E0             JP      Z,INIT          ; "T" - No NMI vector
FEC6 21DEFE             LD      HL,BREAK        ; Set NMI gives break
FEC9 227E0C             LD      (NMI),HL
FECC DDE5               PUSH    IX              ; Get start up condition
FECE F1                 POP     AF              ; "Z" set if cold , Else clear
FECF B7                 OR      A               ; "Cold" or "Cool" start?
FED0 C219E0             JP      NZ,INIT         ; "Cool" don't init NAS-SYS
FED3 060F               LD      B,15            ; Delay for keyboard clear
FED5 CDCDFC             CALL    DELAYB          ; Allow time for key release
FED8 CD0D00             CALL    STMON           ; Initialise NAS-SYS
FEDB C319E0             JP      INIT            ; Initialise BASIC

FEDE F5         BREAK:  PUSH    AF              ; Save character
FEDF 3EFF               LD      A,-1
FEE1 324D10             LD      (BRKFLG),A      ; Flag break
FEE4 F1                 POP     AF              ; Restore character
FEE5 ED45       ARETN:  RETN                    ; Return from NMI

FEE7 00                 NOP

FEE8 DF63       INLINE: SCAL    INLIN           ; Get an input line
FEEA D5                 PUSH    DE              ; Save cursor address
FEEB D5                 PUSH    DE              ; Cursor address to HL
FEEC E1                 POP     HL
FEED 112F00             LD      DE,48-1         ; Lengh of line-1
FEF0 19                 ADD     HL,DE           ; Point to end of line
FEF1 7E         ENDLIN: LD      A,(HL)          ; Get end of line
FEF2 FE20               CP      " "             ; Space?
FEF4 C202FF             JP      NZ,LINTBF       ; No - Copy to buffer
FEF7 1D                 DEC     E               ; Back 1 character
FEF8 3E00               LD      A,0             ; Wasteful test on E
FEFA B3                 OR      E
FEFB CA02FF             JP      Z,LINTBF        ; Start of line - Copy it
FEFE 2B                 DEC     HL              ; Back 1 character
FEFF C3F1FE             JP      ENDLIN          ; Keep looking for end
```

```
FF02 D5          LINTBF: PUSH    DE              ; Line length to BC
FF03 C1                  POP     BC
FF04 03                  INC     BC              ; Length +1
FF05 116110              LD      DE,BUFFER       ; Input buffer
FF08 E1                  POP     HL              ; Line start
FF09 C5                  PUSH    BC              ; Save length
FF0A EDB0                LDIR                    ; Move line to buffer
FF0C 3E00                LD      A,0
FF0E 12                  LD      (DE),A          ; Mark end of buffer with 00
FF0F C1                  POP     BC              ; Restore buffer length
FF10 41                  LD      B,C             ; Length returned in B
FF11 216010              LD      HL,BUFFER-1     ; Point to start of buffer-1
FF14 C9                  RET

FF15 CD90E6      GETXYA: CALL    CHKSYN          ; Make sure "(" follows
FF18 28                  DEFB    "("
FF19 CD41ED              CALL    GETNUM          ; Get a number
FF1C CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
FF1F D5                  PUSH    DE              ; Save "X"
FF20 CD90E6              CALL    CHKSYN          ; Make sure "," follows
FF23 2C                  DEFB    ","
FF24 CD41ED              CALL    GETNUM          ; Get a number
FF27 CD90E6              CALL    CHKSYN          ; Make sure ")" follows
FF2A 29                  DEFB    ")"
FF2B CD8BE9              CALL    DEINT           ; Get integer -32768 to 32767
FF2E E5                  PUSH    HL              ; Save code string address
FF2F FDE1                POP     IY              ; In IY
FF31 CD96FF              CALL    XYPOS           ; Address and bit mask
FF34 F5                  PUSH    AF              ; Save mask
FF35 CDC2FF              CALL    ADJCOL          ; Adjust column
FF38 CD11FE              CALL    SCRADR          ; Get VDU address
FF3B F1                  POP     AF              ; Restore bit mask
FF3C 06C0                LD      B,11000000B     ; Block graphics base
FF3E B0                  OR      B               ; Set bits 7 & 6
FF3F C9                  RET

FF40 CD15FF      SETB:   CALL    GETXYA          ; Get co-ords and VDU address
FF43 F5                  PUSH    AF              ; Save bit mask
FF44 7E                  LD      A,(HL)          ; Get character from screen
FF45 FEC0                CP      11000000B       ; Is it a block graphic?
FF47 D250FF              JP      NC,SETOR        ; Yes - OR new bit
FF4A F1                  POP     AF              ; Restore bit mask
FF4B 77          PUTBIT: LD      (HL),A          ; Put character on screen
FF4C FDE5        RESCSA: PUSH    IY              ; Restore code string address
FF4E E1                  POP     HL              ; From IY
FF4F C9                  RET

FF50 C1          SETOR:  POP     BC              ; Restore bit mask
FF51 B0                  OR      B               ; Merge the bits
FF52 C34BFF              JP      PUTBIT          ; Save on screen
```

```
FF55 CD15FF     RESETB: CALL    GETXYA          ; Get co-ords and VDU address
FF58 F5                 PUSH    AF              ; Save bit mask
FF59 7E                 LD      A,(HL)          ; Get byte from screen
FF5A FEC0               CP      11000000B       ; Is it a block graphic?
FF5C DA75FF             JP      C,NORES         ; No - Leave it
FF5F 063F               LD      B,00111111B     ; Six bits per block
FF61 A0                 AND     B               ; Clear bits 7 & 6
FF62 C1                 POP     BC              ; Get bit mask
FF63 A0                 AND     B               ; Test for common bit
FF64 CA4CFF             JP      Z,RESCSA        ; None - Leave it
FF67 7E                 LD      A,(HL)          ; Get byte from screen
FF68 E63F               AND     00111111B       ; Isolate bit
FF6A A8                 XOR     B               ; Clear that bit
FF6B FEC0               CP      11000000B       ; Is it a graphic blank?
FF6D C24BFF             JP      NZ,PUTBIT       ; No - Save character
FF70 3E20               LD      A," "           ; Put a space there
FF72 C34BFF             JP      PUTBIT          ; Save the space

FF75 C1         NORES:  POP     BC              ; Drop bit mask
FF76 C34CFF             JP      RESCSA          ; Restore code string address

FF79 CD15FF     POINTB: CALL    GETXYA          ; Get co-ords and VDU address
FF7C 46                 LD      B,(HL)          ; Get character from screen
FF7D CDEDFF             CALL    TSTBIT          ; Test if bit is set
FF80 C291FF             JP      NZ,POINTO       ; Different - Return zero
FF83 3E00               LD      A,0
FF85 0601               LD      B,1             ; Integer AB = 1
FF87 E1         POINTX: POP     HL              ; Drop return
FF88 FDE5               PUSH    IY              ; PUSH code string address
FF8A 111DEE             LD      DE,RETNUM       ; To return a number
FF8D D5                 PUSH    DE              ; Save for return
FF8E C3F2F0             JP      ABPASS          ; Return integer AB

FF91 0600       POINTO: LD      B,0             ; Set zero
FF93 C387FF             JP      POINTX          ; Return value

FF96 C1         XYPOS:  POP     BC              ; Get return address
FF97 E1                 POP     HL              ; Get column
FF98 E5                 PUSH    HL              ; And re-save
FF99 C5                 PUSH    BC              ; Put back return address
FF9A 7D                 LD      A,L             ; Get column
FF9B 0601               LD      B,00000001B     ; 2 bits per character
FF9D A0                 AND     B               ; Odd or even bit
FF9E F5                 PUSH    AF              ; Save it
FF9F D5                 PUSH    DE              ; Get row
FFA0 E1                 POP     HL              ; to HL
FFA1 110000             LD      DE,0            ; Zero line count
FFA4 010300             LD      BC,3            ; 3 blocks per line
FFA7 23                 INC     HL
FFA8 ED42       DIV3LP: SBC     HL,BC           ; Subtract 3
FFAA 13                 INC     DE              ; Count the subtractions
FFAB CAB1FF             JP      Z,DIV3EX        ; Exactly - Exit
FFAE F2A8FF             JP      P,DIV3LP        ; More to do
```

```
FFB1 09        DIV3EX: ADD     HL,BC           ; Restore number
FFB2 F1                POP     AF              ; Restore column and odd/even
FFB3 B7                OR      A               ; Set flags (NZ or Z)
FFB4 7D                LD      A,L             ; Get remainder from /3
FFB5 CABAFF            JP      Z,NOREMD        ; No remainder
FFB8 C603              ADD     A,3             ; Adjust remainder
FFBA 47        NOREMD: LD      B,A             ; Bit number+1 to B
FFBB 3E01              LD      A,00000001B     ; Bit to rotate
FFBD 07        SHFTBT: RLCA                    ; Shift bit left
FFBE 10FD              DJNZ    SHFTBT          ; Count shifts
FFC0 1F                RRA                     ; Restore correct place
FFC1 C9                RET

FFC2 C1        ADJCOL: POP     BC              ; Restore return address
FFC3 F1                POP     AF              ; Get bit mask
FFC4 E1                POP     HL              ; Get column
FFC5 F5                PUSH    AF              ; Re-save but mask
FFC6 7D                LD      A,L             ; Get column
FFC7 1F                RRA                     ; Divide by 2
FFC8 C601              ADD     A,1             ; Start at column 1
FFCA E63F              AND     00111111B       ; 0 to 63
FFCC 67                LD      H,A             ; Save column in H
FFCD E5                PUSH    HL              ; Re-save column
FFCE C5                PUSH    BC              ; Put back return
FFCF 7B                LD      A,E             ; Get row
FFD0 C9                RET

FFD1 CDD5FC    SMOTOR: CALL    CASFF           ; Flip tape drive
FFD4 7E                LD      A,(HL)          ; Get byte
FFD5 C9                RET

FFD6 3ACE10    JPLDSV: LD      A,(BRKLIN)      ; CLOAD or CSAVE?
FFD9 FEFF              CP      -1
FFDB C206E9            JP      NZ,SNDHDR       ; CSAVE - Send header
FFDE C310E9            JP      GETHDR          ; CLOAD - Get header

FFE1 CD81EB    CRLIN1: CALL    PRNTCR          ; Output CRLF
FFE4 C3F2E5            JP      GETLIN          ; Get an input line

FFE7 CD81EB    CRLIN:  CALL    PRNTCR          ; Output CRLF
FFEA C3F2E5            JP      GETLIN          ; Get an input line

FFED F5        TSTBIT: PUSH    AF              ; Save bit mask
FFEE A0                AND     B               ; Get common bits
FFEF C1                POP     BC              ; Restore bit mask
FFF0 B8                CP      B               ; Same bit set?
FFF1 3E00              LD      A,0             ; Return 0 in A
FFF3 C9                RET

FFF4 CD9BE6    OUTNCR: CALL    OUTC            ; Output character in A
FFF7 C381EB            JP      PRNTCR          ; Output CRLF

FFFA C3DEFD    JJUMP:  JP      JJUMP1          ; "Cool" start

FFFD C3B1E0    ZJUMP:  JP      BRKRET          ; Warm start
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ABPASS | F0F2 | ABS | F838 | ACCSUM | E940 | ACPASS | F0F1 | ADD64X | FE35 |
| ADDEXP | F7D1 | ADDIG | F977 | ADDPHL | F58E | ADJCOL | FFC2 | ALLFOL | F3C0 |
| ANTVLU | EC65 | ANYNAM | F51C | ARET | FCD8 | ARETN | FEE5 | ARG1 | 0C0C |
| ARG2 | 0C0E | ARGN | 0C2B | ARLDSV | F012 | ARREND | 10DA | ARRLD1 | E8B9 |
| ARRLP | F28B | ARRSV1 | E8BB | ARYLP | E920 | ASC | F391 | ASCTFP | F91A |
| ASPCS | EBC7 | ATN | FC7C | ATN1 | FC99 | ATNTAB | FCA3 | ATOH | E9A5 |
| BAD | F59D | BADINP | EBEC | BAKSTK | E356 | BAKTMP | F371 | BASTXT | 105E |
| BCDEFP | F85F | BFREE | E0B7 | BKSP | 0008 | BNORM | F61E | BNRMLP | F621 |
| BREAK | FEDE | BREAK2 | FE53 | BRKFLG | 104D | BRKLIN | 10CE | BRKMSG | E350 |
| BRKRET | E0B1 | BS | 0010 | BSERR | F045 | BUFFER | 1061 | BYTSFT | F756 |
| CASFF | FCD5 | CASFFW | FCC8 | CFEVAL | EFA8 | CHARTY | EF56 | CHEKFN | F189 |
| CHKBRK | FD40 | CHKLTR | E977 | CHKSTK | E38A | CHKSUM | 104A | CHKSYN | E690 |
| CHKTYP | ED46 | CHR | F3A2 | CHSUMS | E960 | CIN | 0C75 | CLEAR | E9CA |
| CLOAD | F4F9 | CLOAD1 | F52B | CLOAD2 | F52D | CLOADE | F55F | CLOADV | F55C |
| CLOTST | E6CC | CLREG | E4DF | CLRPTR | E4BA | CLS | FD8B | CMPFP | F8A8 |
| CMPLOG | EEBA | CMPNUM | F88E | CMPRES | EEFE | CMPSTR | EEE6 | CN | 0020 |
| CNVIN1 | FD20 | CNVIN2 | FD27 | CNVIN3 | FD2E | CNVIN4 | FD35 | CNVIN5 | FD3C |
| CNVNUM | F926 | COMMAN | 1043 | COMPL | F67E | CONCAT | F306 | CONEXP | F956 |
| CONMON | FCD9 | CONOT1 | FCF1 | CONPOS | F61B | CONT | E89E | CONTAD | 10D4 |
| CONVAR | EE22 | CONVIN | FD19 | COPY | E021 | COS | FC00 | COUNT | E746 |
| CPDEHL | E68A | CPYLIT | E5B1 | CR | 000D | CRARLP | F065 | CREARY | F04A |
| CRESTR | EAC9 | CRLIN | FFE7 | CRLIN1 | FFE1 | CRNCLP | E512 | CRTMST | F1C2 |
| CRTST | F1CE | CRTSTE | F1E4 | CRUNCH | E509 | CS | 000C | CSAVE | F4C3 |
| CSTART | E012 | CTLOFG | 1045 | CTRLC | 0003 | CTRLG | 0007 | CTRLO | 000F |
| CTRLR | 0012 | CTRLS | 0013 | CTRLU | 0015 | CTRLZ | 001A | CUROPR | 10C5 |
| CURPOS | 10AB | CURSOR | 0C29 | DATA | EA70 | DATFLG | 10AE | DATLIN | 10C9 |
| DATSNR | E3A7 | DCBCDE | F8DF | DD | 0012 | DDERR | E3B6 | DEEK | FDBC |
| DEF | F106 | DEFSIZ | F06D | DEINT | E98B | DEL | 007F | DELAY | FD9B |
| DELAY1 | FD9C | DELAYB | FCCD | DELCHR | E5E1 | DEPINT | E985 | DETHL4 | F86E |
| DETHLB | F870 | DIGTXT | FA20 | DIM | EF28 | DIMRET | EF1F | DINPOS | E6BF |
| DIV | F767 | DIV1 | 100A | DIV10 | F75B | DIV2 | 100E | DIV3 | 1012 |
| DIV3EX | FFB1 | DIV3LP | FFA8 | DIV4 | 1015 | DIVLP | F78E | DIVSUP | 1009 |
| DOAGN | E4F8 | DOCOM | EB98 | DODEL | E5C1 | DOEBIT | FA60 | DOFN | F133 |
| DOKE | FDC7 | DONULL | EB86 | DOSPC | EBC2 | DOSUM | E94D | DOTAB | EBAF |
| DPOINT | F952 | DTSTR | F1D2 | DVBCDE | F769 | DZ | 0014 | DZERR | E3B0 |
| ECHDEL | E5D5 | EDIGIT | F999 | ENDBUF | E5B8 | ENDCON | F965 | ENDDIM | F0CC |
| ENDINP | EB7C | ENDLIN | FEF1 | ENDNAM | EF4A | ENDPRG | E87A | ENFMEM | E393 |
| ERRIN | E3E1 | ERRLIN | 10D2 | ERRMSG | E33F | ERROR | E3C1 | ERRORS | E2B9 |
| ESC | 001B | EVAL | ED5A | EVAL1 | ED5D | EVAL2 | ED66 | EVAL3 | ED69 |
| EVLPAR | EE09 | EVNOT | EF08 | EXCUTE | E816 | EXP | FAFA | EXPLP | F944 |
| EXPTAB | FB3A | EXPTEN | FA72 | EXTIG | ECC1 | FANDT | ECEB | FC | 0008 |
| FCERR | E9A0 | FDTLP | ECD2 | FILE | F58E | FILFND | F574 | FINDEL | F0A8 |
| FLGDIF | F81E | FLGREL | F825 | FLGVER | F509 | FLPLED | FE39 | FNARG | 10E0 |
| FNCTAB | E10F | FNDARY | F018 | FNDELP | F0AD | FNDEND | E48D | FNDNUM | F481 |
| FNDTOK | E71B | FNDVAR | EF8F | FNDWRD | E53C | FNOFST | EE33 | FNRGNM | 10DE |
| FNTHR | EF9D | FNVAL | EE5F | FOPRND | ED92 | FOR | E779 | FORFLG | 10CB |
| FORFND | E7A9 | FORSLP | E78D | FOUND | F594 | FPADD | F5CD | FPBCDE | F854 |
| FPEXP | 10E7 | FPINT | F8BB | FPMULT | F708 | FPREG | 10E4 | FPROND | F665 |
| FPSINT | E97F | FPTHL | F86B | FRE | F0D0 | FRENUM | F0EC | FRMEVL | EE25 |
| GARBGE | F253 | GARBLP | F256 | GETCHR | E836 | GETCMD | E405 | GETHDR | E910 |
| GETINP | FD05 | GETINT | F484 | GETLEN | F386 | GETLIN | E5F2 | GETLN | E9A6 |
| GETNUM | ED41 | GETNXT | E557 | GETSTR | F350 | GETTIN | FD13 | GETVAR | EF2D |
| GETXYA | FF15 | GNXARY | F28A | GOFUNC | EE67 | GOSUB | EA1C | GOTO | EA2D |
| GRBARY | F2AA | GRBDON | F22B | GRBLP | F264 | GSTRCU | F353 | GSTRDE | F357 |
| GSTRHL | F356 | GTFLNM | F395 | GTFNAM | EF32 | GTLNLP | E9A9 | GTSIXD | F9EA |
| GTVLUS | EC3D | GUART | FD56 | HALF | FA91 | HALFPI | FC4A | HDRLP | E912 |
| ID | 0016 | IDTEST | F17B | IF | EAFF | IFGO | EB0D | IFJMP | E81D |
| IGCHR | FD00 | INCHL | F869 | INCLEN | E6BB | INDFND | E370 | INEWLN | E455 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| INIT | E019 | INITAB | E2DF | INITBE | E33F | INITST | FEBB | INLINE | FEE8 |
| INMSG | E346 | INP | F441 | INPBIN | EC8F | INPBRK | E877 | INPORT | 103F |
| INPSUB | 103E | INPUT | EBFD | INRNG | F9F3 | INT | F8E6 | INTVAR | E4C9 |
| INVSGN | F83C | ITMSEP | EC80 | JJUMP | FFFA | JJUMP1 | FDDE | JPLDSV | FFD6 |
| JSTZER | FA7C | KILFOR | ED31 | KILIN | E5EC | LCRFLG | 10AC | LDNMI1 | E73C |
| LEFT | F3B2 | LEN | F382 | LET | EA87 | LETNUM | EADA | LETSTR | EAA2 |
| LF | 000A | LFRGNM | F437 | LINEAT | 105C | LINEIN | F9A5 | LINES | FDAD |
| LINESC | 1046 | LINESN | 1048 | LINFND | E43E | LINTBF | FF02 | LIST | E6DD |
| LISTLP | E6E9 | LOADFP | F862 | LOG | F6C7 | LOGTAB | F6BA | LOKFOR | E35A |
| LOOPST | 10C7 | LS | 001C | LSTBIN | 10CC | LSTLP2 | E709 | LSTLP3 | E70C |
| LSTRAM | 10AF | LSTRND | 103A | LTSTND | EC9A | LWIDTH | 1042 | MAKINT | F487 |
| MAKNUM | FA0F | MANLP | F92E | MATCH | E589 | MEMMSG | E103 | MFLP | 0051 |
| MID | F3EC | MID1 | F3B8 | MIDNUM | F43C | MINCDE | F60D | MINUS | EE11 |
| MKTMST | F1BF | MLDBLP | F907 | MLDEBC | F8FF | MLOOP | E049 | MLSP10 | F7FC |
| MO | 0024 | MONITR | FEA2 | MONLD | FE88 | MONOUT | FE45 | MONSTT | 0000 |
| MONTST | FE6D | MONTYP | 008D | MONVE | FEAA | MORDT | ECA6 | MORINP | E610 |
| MOVBUF | E474 | MOVDIR | E591 | MOVLP | E37F | MOVSTR | E37C | MOVUP | E379 |
| MRPRNT | EB1F | MSIZE | E036 | MUL8LP | F733 | MULLN2 | F6FF | MULT | F706 |
| MULT8 | F72A | MULTEN | F970 | MULVAL | 10F6 | MVSTPT | EAD1 | NASOUT | FCFB |
| NEDMOR | EC39 | NEGAFT | FAA7 | NEW | E4B9 | NEXITM | EBD2 | NEXT | ECF6 |
| NEXT1 | ECF9 | NF | 0000 | NFERR | E3B3 | NMI | 0C7E | NMIFLG | 104C |
| NOCHNG | E581 | NOENED | FA7F | NOLIN | E88D | NOMADD | F744 | NOMLAD | F915 |
| NOPMPT | EC17 | NOREMD | FFBA | NORES | FF75 | NORMAL | F638 | NOSPC | E578 |
| NOSWAP | F5E7 | NOTSTR | EF65 | NOXOR | F467 | NSCFOR | EF75 | NULFLG | 1044 |
| NULL | E8B1 | NULLP | EB8D | NULLS | 1041 | NUMASC | F9B8 | NXTARY | F02C |
| NXTBYT | E567 | NXTCHR | E5A8 | NXTDAT | 10DC | NXTDTA | EA6F | NXTITM | EC31 |
| NXTOPR | 10D0 | NXTSTL | EA76 | NXTSTT | EA79 | OD | 0006 | OKMSG | E34B |
| OM | 000C | OMERR | E3A2 | ON | EAE1 | ONGO | EAF0 | ONGOLP | EAF1 |
| ONJMP | E81E | OPNPAR | ED56 | OPRND | EDD1 | OS | 001A | OTKLN | E5E9 |
| OTPORT | 1007 | OUTBAD | F56B | OUTC | E69B | OUTCHR | FCFD | OUTEXP | FA70 |
| OUTIT | E67C | OUTNBS | E682 | OUTNCR | FFF4 | OUTSUB | 1006 | OUTWRD | E725 |
| OV | 000A | OVERR | E3BC | OVTST1 | F7EF | OVTST2 | F7F4 | OVTST3 | F7F5 |
| PADD | F994 | PAND | EE81 | PASSA | F101 | PBUFF | 10E9 | PEEK | F5A3 |
| PEND | E872 | PHLTFP | F851 | PLUCDE | F672 | PNORM | F640 | POINT | 1051 |
| POINT0 | FF91 | POINTB | FF79 | POINTX | FF87 | POKE | F5AA | POPAF | F245 |
| POPHL | F36F | POPHRT | F754 | POPNOK | E3F7 | POR | EE80 | POR1 | EEA3 |
| PORT0 | 0C00 | POS | F0FE | POSINT | E982 | POUT | F44D | POWER | FAB5 |
| POWER1 | FAC5 | POWER2 | FAE2 | POWERS | FA95 | PRINT | EB23 | PRITAB | E2A4 |
| PRNTCR | EB81 | PRNTHL | F9AD | PRNTLP | EB26 | PRNTNB | EB69 | PRNTOK | E3F8 |
| PRNTST | EB6D | PRNUMS | F20F | PROCES | E629 | PROGND | 10D6 | PROGST | 10F9 |
| PROMPT | E4FC | PRS | F210 | PRS1 | F213 | PRSLP | F21A | PSET | 1054 |
| PSUB | F5C8 | PTRLP | E481 | PUTBIT | FF4B | PUTBUF | E668 | PUTCTL | E66D |
| PUTFID | E7EE | QTSTLP | F1D5 | QTSTR | F1CF | QUARTR | FC4E | READ | EC2C |
| READFG | 10CD | REDO | EBD9 | REM | EA72 | RESCSA | FF4C | RESDIV | F7A1 |
| RESEED | FBEC | RESET | 1057 | RESETB | FF55 | RESTNL | E85B | RESTOR | E846 |
| RESZER | F633 | RETADR | EFDC | RETCTC | FE65 | RETINT | F82A | RETLIN | EA6A |
| RETNAD | E58D | RETNUL | EFDF | RETNUM | EE1D | RETREL | F81C | RETURN | EA4B |
| RG | 0004 | RIGHT | F3E2 | RIGHT1 | F3B6 | RINPUT | 104E | RLTLP | ED76 |
| RND | FB8B | RND1 | FBC7 | RND2 | FBE3 | RNDTAB | FBF4 | RNGTST | FA82 |
| RONDB | F654 | RONDUP | F653 | ROUND | F5BB | RSCALE | F98E | RSLNBK | E770 |
| RSTSTR | F405 | RUART | F4B4 | RUN | EA10 | RUNCNT | E7F2 | RUNFST | E4C5 |
| RUNLIN | EA2C | SAVE | FE73 | SAVEXP | F634 | SAVSTP | E7E5 | SAVSTR | F1AA |
| SBSCPT | EFEA | SCALE | F692 | SCALLP | F694 | SCALMI | F959 | SCALPL | F96F |
| SCNEND | F2E1 | SCPTLP | EFF0 | SCRADR | FE11 | SCREEN | FDE6 | SEARCH | E555 |
| SEED | 1017 | SETB | FF40 | SETIO | F471 | SETLIN | E733 | SETLIT | E59F |
| SETOR | FF50 | SETPTR | E47C | SETTOP | E06D | SFTENT | FD70 | SFTPRG | E446 |
| SGN | F822 | SGNEXP | EE70 | SGNRES | 10E8 | SHFTBT | FFBD | SHRITE | F6A1 |
| SHRLP | F6A4 | SHRT1 | F6A8 | SIGNON | E0C5 | SIGNS | F879 | SIN | FC06 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SIN1 | FC36 | SINTAB | FC52 | SIXDIG | F9D5 | SMOTOR | FFD1 | SMPVAR | F275 |
| SMSER1 | FB6A | SN | 0002 | SNDARY | E91D | SNDHDR | E906 | SNERR | E3AD |
| SPCFST | F9C6 | SPCLP | EBCB | SQR | FAAC | SRCHLN | E499 | SRCHLP | E49C |
| SSTSA | F33D | ST | 001E | STACK | 1066 | STAKFP | F844 | STALL | E866 |
| START | E000 | STARTB | E003 | STKTHS | EDBA | STLOOK | 115D | STMON | 000D |
| STOP | E870 | STORED | E9ED | STPOOL | F2B8 | STR | F19A | STRADD | F2BB |
| STRBOT | 10C3 | STRENT | EC83 | STRSPC | 105A | STTLIN | EB74 | SUART | FD68 |
| SUBCDE | F5CA | SUBPHL | F5C4 | SUMLP | FB73 | SUMOFF | E937 | SUMSER | FB5B |
| SUPTLZ | FA54 | SVNAM2 | EF49 | SVSTAD | F1C8 | T2DUMP | 03D1 | T4READ | 070C |
| T4WR | 0400 | TAN | FC67 | TBRK | 001C | TBRK2 | FD50 | TBS | 001D |
| TCHINP | FE62 | TCLS | FD96 | TCR | 001F | TCS | 001E | TCUR | 0C18 |
| TESTOS | F247 | TESTR | F229 | THSFIL | F548 | TIN | 0C4D | TLOAD | FE99 |
| TM | 0018 | TMERR | E3BF | TMFLP | FE42 | TMNCUR | FE04 | TMNOUT | FE4F |
| TMPSTR | 10BF | TMSTPL | 10B3 | TMSTPT | 10B1 | TOPOOL | F3AE | TOSTRA | F346 |
| TOUT | 0C4A | TRYAGN | FA2F | TSALP | F347 | TSAVE | FE7F | TSTBIT | FFED |
| TSTBRK | E861 | TSTMEM | E05B | TSTNUM | ED44 | TSTOPL | F1F0 | TSTRED | EEA8 |
| TSTREM | E5A2 | TSTSGN | F813 | TSTSTR | ED45 | TTYLIN | E607 | TYPE | 10AD |
| UARTD | 0001 | UARTOT | FD5F | UARTS | 0002 | UF | 0022 | UFERR | E3B9 |
| UL | 000E | ULERR | EA46 | UNITY | F6B6 | UPDATA | E85C | URTOLP | FD61 |
| USR | 1003 | VAL | F41C | VAREND | 10D8 | VDU | 0800 | WAIT | F453 |
| WAITLP | F468 | WARMST | E0AE | WIDTH | FDA5 | WORDS | E143 | WORDTB | E25A |
| WRKSPC | 1000 | WUART | F4BA | WUART2 | F4B7 | XYPOS | FF96 | ZDATA | 0083 |
| ZDIV | 00AF | ZEND | 0080 | ZEQUAL | 00B4 | ZERARY | F08B | ZERBYT | E34A |
| ZEROLP | EFCE | ZFN | 00A7 | ZFOR | 0081 | ZGOSUB | 008C | ZGOTO | 0088 |
| ZGTR | 00B3 | ZJUMP | FFFD | ZLEFT | 00CD | ZLTH | 00B5 | ZMINUS | 00AD |
| ZNEW | 00A4 | ZNOT | 00AA | ZONELP | EBA6 | ZOR | 00B2 | ZPLUS | 00AC |
| ZPOINT | 00C7 | ZPRINT | 009E | ZREM | 008E | ZSGN | 00B6 | ZSPC | 00A8 |
| ZSTEP | 00AB | ZTAB | 00A5 | ZTHEN | 00A9 | ZTIMES | 00AE | ZTO | 00A6 |



---

# THE END