This document applies to version 2.40 of VPCalc,
Copyright (c) 1981-2000 by author: Harry J. Smith, Saratoga, CA.

Introduction -

To use the program type the name of the EXE file at the DOS prompt line with a return and no parameters. The program will load and respond with the following screen display:

```
-------------------------------------------------------------------------------

     ╔═════════════════════════════════════════════════════════════════════╗
     ║      VPCalc - Variable Precision floating decimal calculator        ║
     ║         Version 2.40, last revised: 2000/04/05, 1600 hours          ║
     ║         Copyright (c) 1981-2000 by author: Harry J. Smith,          ║
     ║   19628 Via Monte Dr., Saratoga, CA 95070.  All rights reserved.    ║
     ╚═════════════════════════════════════════════════════════════════════╝

          This is a Variable Precision floating decimal Calculator.
       It can compute with numbers of up to 114639 decimal digit each.
         At the Command: prompt, type a number, a primitive op code,
      an equation like: A = (12,345 + 2 * B * (C + D) / Sin(E + F)) ^ 2,
          or an If statement like: If A = B Then C = D Else C = E.
        Code files may contain Labels:, GoTo Label, and GoUpTo Label.

   Status:    1001 <- Max decimal digits allowed in mantissa
                49 <- Current max decimal digits in mantissa
                 7 <- Decimal digits to truncate in display
                42 <- Max decimal digits in display
                 1 <- Input lines (1, 2, 3, or 4)
                On <- Rounding mode
                On <- Degree Trig mode            On <- Save Top
               Off <- Echo screen to printer
               Off <- Diagnostic mode           MemAvail = 392664

               **** PRESS ANY KEY TO CONTINUE OR F1 FOR HELP **** _

Running code file "AutoExec.VPC"
Full name = A:\AUTOEXEC.VPC

X = 0.0 (False)

File "AutoExec.VPC" closed

X = 0.0 (False)

Command: _____________________________________________________________________
-------------------------------------------------------------------------------
```

The "MemAvail = 392664" is an example output and refers to the amount of memory in 8-bit bytes that is available to dynamically-allocate space to store the variables as they are created and as they change in their number of significant digits. Numbers are stored in memory in decimal, actually they are stored in base 10,000,000, as an array of super-digits. Each super-digit is stored as a single precision floating point number between 0 and

9,999,999. As variables change in value, memory is dynamically reallocated so no more memory is used than is needed to represent their current precision.

At any given time there is a current max decimal digits that will be computed for a mantissa and a max decimal digits to ever allow in a mantissa. These values are initialized to 56 and 1001 respectively and can be changed after the program is running.

At the "Command:" prompt, commands may be entered one at a time each followed by a return, or several on a line separated by one or more spaces or semicolons. A separator is never needed between primitive op codes and is only needed otherwise to prevent ambiguity of meaning. Commands are not case sensitive, upper and lower case letters are always interpreted the same.

There are four basic types of commands: 1) Enter a number, 2) Execute a primitive op code, 3) Evaluate an equation, and 4) Do a procedure.

The calculator contains a list of named numbers or variables. Initially the list contains only the item X = 0.0. Its name is X and its value is 0.0. Items can be added to the list by evaluating an equation. Equations are assignment statements like <variable> = <expression>. A <variable> is a name of a variable and an <expression> is an expression of terms, factors, functions variables, constants, and <expression>s. Item names are limited to 250 characters with all characters significant but not case sensitive.

Parentheses can be nested to any level in expressions. Any number of closing parentheses can be replaced with a single semi-colon or an end-of-line. Thus x = (a / (b * (c + d; is a legal assignment statement and is interpreted as (a / (b * (c + d))).

Any time a variable is referenced that is not currently on the list, it is added to the list with a value of 0.0.

At any given time, one item on the list is the active item. This is referred as the item on top of the list. Initially item X is the active item. When an expression is evaluated, the variable being assigned a value becomes the active item. If the <expression> part of an assignment statement is left blank, the referenced variable becomes the active item without changing its value.

When a number is entered, it replaces the value of the active item. Numbers (constants) may have a leading sign and embedded commas. An example of a constant is -12,345.678,9E+1,234. The commas, plus signs, decimal point, and the E power of 10 factor are optional. The numbers 1.0E+1,50323,85525 is at the upper end of the dynamic range of the calculator. Because commas are allowed in input constants to make them readable, commas are not used to separate arguments in functions calls. An example of this is: X = Atan2(12,345' 78,901). A tic mark separates the two arguments instead of a comma as is normally done.

If the program is executed from the DOS prompt with one or more parameters, the initial help menu is not displayed and the parameters are taken as an initial VPCalc command line. This allows you to control the execution of VPCalc from batch files and VPCalc code files with no operator intervention. The VPCalc code file AutoExec.VPC is always run first, even before the DOS command line commands.

Special handling is given to the first parameter on the DOS command line. If it ends in .VPC, it is changed to Run("... .VPC") so this VPCalc code file will be run. If it ends in .VPN, it is changed to ReadN("... .VPN") so this VPCalc number file will be read and added to the list of items. This allows VPCalc to be run by shell programs, such as DosShell or XTree, by associating the file VPCalc.Exe with the extensions VPC and VPN, and then opening a file with one of these extensions.

Primitives -

Primitives that act on a number, acts on the currently active item. In the following description of primitives, the currently active item is called x for convenience.

? => General Help:

This causes the help screens to be displayed. The first is the same as the screen displayed when you enter the program, but the values currently set by the D, E, H, M, T, U, V, and @ commands and set by the Diag(X), InputLines(X), SetD(X), and SetMax(X) procedures are displayed. The other 4 screens are as follows:

```
--------------------------------------------------------------------------------

                       The primitive op codes are:

A => Auto Display on/off              |  T => Set digits to truncate
B => Display learn line               |  U => Set rounding mode
C => Change sign of x                 |  V => Set non-rounding mode
D => Set degree trig mode             |  W => Write number to file
E => Set radian trig mode             |  X => Learn, Execute
F => ! => Factorial                   |  Y => Delete (Yank) number from list
G => Set Digits/Group                 |  Z => Output list
H => Echo screen to printer/Log file  |  @ => Substitute Log file for printer
I => Input number from file           |  " => Start/Stop file name or comment
J => Run VPCalc code from file        |  % => Set FMB = x, FMB on list
K => Execute learn line x times       |  / => x = x Mod FMB, FMB on list
L => Reduce precision of x            |  $ => Restart
M => Set digits in Mantissa           |  > => Write configuration: Config.VPC
N => Generate a random number         |  < => Read configuration: Config.VPC
O => x = 1 / x                        |  ] => Write entry history: Hist.VPT
P => Compute Pi                       |  [ => Read entry history: Hist.VPT
Q => Quit to end the program          |  ? => General Help
R => Square root of x                 |  F1 => Hot Help
S => Square x                         |  ESC => Interrupt a long process
```

```
            **** PRESS ANY KEY TO CONTINUE HELP OR PRESS ESCAPE **** _

-------------------------------------------------------------------------------

 The infix operators are: +, -, *, /, ^, @, #, %, \, &, |, <, =, >, <=, <>, >=

         A = X + Y  =>  Set A to X plus Y
         A = X - Y  =>  Set A to X minus Y
         A = X * Y  =>  Set A to X times Y
         A = X / Y  =>  Set A to X divided by Y
         A = X ^ Y  =>  Set A to X to the power Y
         A = Y @ X  =>  Set A to Atan2(Y over X)
         A = X # Y  =>  Set A to Mag(X' Y) = SqRt(Sq(X) + Sq(Y))
         A = X % Y  =>  Set A to Mod(X' Y) = X Modulo Y
         A = X \ Y  =>  Set A to GCD(X' Y) = Greatest Common Divisor
         A = X & Y  =>  Set A to 1 if X and Y are not 0, else set A to 0
         A = X | Y  =>  Set A to 1 if X or Y, is not 0, else set A to 0
         A = X < Y  =>  Set A to 1 if X < Y, else set A to 0
         A = X = Y  =>  Set A to 1 if X = Y, else set A to 0
         A = X > Y  =>  Set A to 1 if X > Y, else set A to 0
         A = X <= Y  =>  Set A to 1 if X <= Y, else set A to 0
         A = X <> Y  =>  Set A to 1 if X <> Y, else set A to 0
         A = X >= Y  =>  Set A to 1 if X >= Y, else set A to 0

            **** PRESS ANY KEY TO CONTINUE HELP OR PRESS ESCAPE **** _

-------------------------------------------------------------------------------


-------------------------------------------------------------------------------

                         The procedures supported are:

          AutoDisplay(X) => Set Auto display on if X <> 0, else off
            ClearHist    => Clear history of previous operator entries
                 Diag(X) => Set diagnostic mode on or off
           EchoScreen(X) => Echo screen to printer/Log file, on or off
           InputLines(X) => Set number of input lines (1, 2, 3, or 4)
              LogFile(X) => Substitute Log file for printer, on or off
             LX => LT    => Restore LastTop to top of the list
                 Next    => Move to next item on the list (no argument)
                ReadN(F) => Read file F = "ccc...c", F is optional
         Restore/Save    => Restore or Save Configuration, History, & List
                  Run(F) => Run VPCalc code from file F, F is optional
              SaveTop(X) => Set "save top value in LastTop" on or off
          ScientificN(X) => Force scientific notation on iff X <> 0
                 SetD(X) => Set max decimal digits in display
               SetMax(X) => Set max decimal digits allowed in mantissa
                VPCIn(F) => Enter file name F = "ccc...c" for J command
               VPLOut(F) => Enter file name F = "ccc...c" for @ command
                VPNIn(F) => Enter file name F = "ccc...c" for I command
               VPNOut(F) => Enter file name F = "ccc...c" for W command
                Write(X) => Output X, (X may be "ccc...c", X is optional)
              WriteLn(X) => Write(X) and a line feed
               WriteN(F) => Write X to file F = "ccc...c", F is optional
           **** PRESS ANY KEY TO CONTINUE HELP OR PRESS ESCAPE **** _
--------------------------------------------------------------------------------

                         The functions supported are:

     Abs(X) = AbsoluteValue(X)         |       LnL(X) = NaturalLog(X + 1)
    Acos(X) = ArcCoSine(X)             |       Log(X) = LogBase10(X)
   Acosh(X) = ArcHyperbolicCoSine(X)   |       Lop(X) = ReducePrecision(X)
    Asin(X) = ArcSin(X)                |    Mag(X' Y) = SqRt(Sq(X), Sq(Y))
   Asinh(X) = ArcHyperbolicSine(X)     |    Mod(X' Y) = X - (Int(X/Y) * Y)
    Atan(X) = ArcTangent(X)            |   PowM(X' Y) = (X to the Y) Mod FMB
Atan2(Y' X) = ArcTangent(Y over X)     |        RN(X) = RandomNumber(Seed=X)
   Atanh(X) = ArcHyperbolicTangent(X)  |       Sin(X) = Sine(X)
     Cos(X) = CoSine(X)                |      SinH(X) = HyperbolicSine(X)
    CosH(X) = HyperbolicCoSine(X)      |        Sq(X) = X Squared
     Exp(X) = eToThePower(X)           |      SqRt(X) = SquareRoot(X)
    ExpL(X) = Etothepower(X) - 1       |       Tan(X) = Tangent(X)
     Fac(X) = Factorial of Int(X)      |      TanH(X) = HyperbolicTangent(X)
    Frac(X) = FractionalPart(X)        |     ToDeg(X) = RadiansToDegrees(X)
  GCD(X' Y) = Greatest Common Divisor  |     ToRad(X) = DegreesToRadians(X)
     Int(X) = IntegerPart(X)           |           -X = Negative of X, 0 - X
     Inv(X) = 1 / X                    |           +X = Positive of X, 0 + X
      Ln(X) = NaturalLog(X)            |           !X = Not X, 0 -> 1 else 0

               **** PRESS ANY KEY TO ENTER CALCULATOR MODE **** _
--------------------------------------------------------------------------------
```

A => Auto Display on/off:

Normally, after each command line is executed, the name and decimal value of the currently active item on the list is displayed. When computing with numbers with many significant digits, the time spent in producing this display can be excessively large. It is desirable then to be able to prevent this automatic display. Each time the A command is given the selection status of this option is reversed.

A word about the displayed value is in order. As an example, if the first command line you enter after starting the program is 3O, the response will be:

```
   X = 3.33333,33333,33333,33...333,33333,33333,3 E-1 (42) [49]
```

The E-1 means that X = 3.3... times 10 to the minus one, the 42 in parentheses means there are 42 decimal digits displayed, and the 49 in brackets means that X is stored in memory with 49 decimal digits of precision. The number in brackets is always a multiple of seven since an item's value is stored in memory in an array of super-digits of seven decimal digits each.

B => Display learn line:

The calculator contains a learned line, see the X primitive to enter and execute the learned line. The B command displays the current contents of the learned line. After the B command is executed, the F3 and F4 keys will restore the input line to the contents of the learned line instead of the previously typed command line.

C => Change sign of x:

This is the same as multiplying x by minus one. Negative numbers can be entered by preceding them with a minus sign. The x referenced here is the current active item on the list of variables.

D => Set degree trig mode (nominal):

The trigonometric functions, Sin(X), Cos(X), Asin(X), Acos(X), Tan(X), Atan(X), and Atan2(Y' X), normally assume the angle involved in either the input or output is expressed in degrees. If radians are desired, use the E command. When degrees are desired, use the D command. The degree trig mode stays selected until changed by the E command.

E => Set radian trig mode:

The trigonometric functions, Sin(X), Cos(X), Asin(X), Acos(X), Tan(X), Atan(X), and ATan2(Y' X), normally assume the angle involved in either the input or output is expressed in degrees. If radians are desired, use the E command. When degrees are desired, use the D command. The radian trig mode stays selected until changed by the D command.

F => ! => Factorial:

Replaces x with the factorial of x = 1 * 2 * 3 * ... * x. Only the integer portion of x is used in the calculation.

G => Set Digits/Group:

The G command will set the number of digits per group to the current value of x. If this is set to 3, numbers will be displayed with a comma after every 3rd digit like 1.234,567,89 E+34,457. If this is set to less than 1, no commas will be displayed.

H => Echo screen to printer/Log file:

The H command causes all output to the screen to be echoed to the printer or to a disk file. See the @ primitive command for opening a file for this purpose. Each time the H command is given the selection status of this option is reversed.

I => Input number from file:

The I command will use the last entered comment as a file name and read this file as a VPCalc formatted number and assign it to the current active item. It is assumed that the file was created by the W command. See the W command for the format of file names. If a comment has not been entered, the file name NoName.VPN is used. The VPNIn procedure can be used to give an override file name for the I command. As files are input by the I command, the most significant bit of each byte read is set to zero to allow the files to be created or modified by a text editor that uses these upper bits as flag bits.

J => Run VPCalc code from file:

The J command will use the last entered comment as a file name and read this file as a text file. Each line of the file will be interpreted as a VPCalc command line and executed. If comment commands and J commands exist in the text file, these other referenced files will be opened and processed. The only limitation to this nesting of code files is the availability of memory and buffers. If a comment has not been entered, the file name NoName.VPC is used. The VPCIn procedure can be used to give an override file name for the J command. As files are input by the J command, the most significant bit of each byte read is set to zero to allow the files to be created or modified by a text editor that uses these upper bits as flag bits.

K => Execute learn line x times:

This will cause the learned line to be executed x time. x must be in the range 0 <= x <= 2,147,483,647 (2^31 - 1). The x referenced here is the current active item on the list of variables. If x is larger than this max, then the max will be used. A long repetition of a learned line can always be interrupted by using the ESC key.

L => Reduce precision of x:

This command removes or Lops off the least significant super-digit of x. If rounding is turned on, the removed super-digit is used to round into the new least significant super-digit. In VPCalc numbers are normalized from both sides. If a calculation results in a number with some trailing zero bytes, these bytes are removed, the count of the number of bytes in the mantissa is reduced and memory is reallocated. The L command can result in many bytes being removed if removing one byte results in many trailing zeros.

M => Set digits in Mantissa:

The M command will set the current value of the maximum number of decimal digits allowed in a floating point number to the current value of x. If there are items on the list containing more than this number of digits, they will be reduced to contain at most this number of digits. Some messages output by the calculator contain the name FMC. For example, the message "Error in Pi, FMC = 143" would be given if you were running at 1001 decimal digits of precision and the value if Pi stored in file "Pi.VPN" had less than 143 super-digits. FMC is the current max number of super-digits in a floating point number. If x is not a multiple of 7, when the M command is given, then the next higher multiple of 7 is used. If x is less than 14, it is set to 14.

N => Generate a random number:

The N command generates a random number between zero and 1.0 and assigns it to the current active item on the list. This number will never have more than 35 significant decimal digits. Theoretically the random number generator will cycle after 10 ^ 35 numbers, but the earth will not last that long. The items RN, RNA, and RNC are put on the list by the random number command. The equation used is: x = RN = (RNA * RN * 10^35 + RNC) mod (10^35) / 10^35, where RNA and RNC are 35 digit integers.

O => x = 1 / x:

Replace x with 1.0 divided by x, error if x = 0.

P => Compute Pi:

If Pi is on the list, then x = Pi. If Pi is not on the list, the file Pi.VPN is read-in, Pi added to the list, and x = Pi. If the file Pi.VPN is not found, Pi is computed by algorithm b. Algorithm b is documented in Scientific American, Feb 1988, Ramanujan and Pi, by Jonathan M. Borwein and Peter B. Borwein.

```
   Pi = 3.14159,26535,89793,23846,26433,83279,50288 E+0 (36) [49]
```

Q => Quit to end the program:

The program exits back to the operating system with no questions asked.

R => Square root of x:

x is replaced with the positive square root of x, error if x < 0.

S => Square x:

x is replaced with the square of x.

T => Set digits to truncate:

The T command will set the number of decimal digits to truncate for display to the current value of x. The calculator is initialized with this set to 7 decimal digits.

U => Set rounding mode:

This command sets rounding on. When rounding is on, the results of all numerical operations are rounded to the maximum number of bytes in mantissa. When rounding is off, these results are truncated to the maximum number of bytes in mantissa. Use the M command to set the maximum number of bytes in mantissa. The IEEE standard of round to even is used, e.g., all numbers in the closed interval [11.5, 12.5] round to 12.

V => Set non-rounding mode:

This command sets rounding off. When rounding is off, the results of all numerical operations are truncated to the maximum number of bytes in mantissa. Use the M command to set the maximum number of bytes in mantissa.

W => Write number to file:

The W command will use the last entered comment as a file name and write register x into this file as a VPCalc formatted number. This number can be reread into x by the I command. If the file already exists, it will be erased and recreated. For example, the following are valid file names:

```
"File.Ext" File.Ext is on default drive and directory
"B:FileName.Ext" FileName.Ext is on B: drive, current B: directory
"Pi.VPN" PI.VPN is on default drive and directory
"C:\Direct\File.Ext" File.Ext is on C: drive, Direct directory
```

If the file name does not have a period, the extension .VPN is added. If a comment has not been entered, the file name NoName.VPN is used. The VPNOut procedure can be used to give an override file name for the W command. The file written is a text file and can easily be browsed and read by other programs. The contents of the file for 1/7 is:

```
                         <3 blank lines>
OneOver7 = m.n E-1, m.n =

1.
42857 14285 71428 57142 85714 28571 42857 14285 71428 57142
85714
E-1 (56)
                        <51 blank lines>
                             Page 1
                         <4 blank lines>
```

X => Learn, Execute:

If this is the last command on a command line, then it caused the learned line to be executed once. If not the last command on the line, this command stores all the commands following on the same line as this one into the learned line. Execution of the current line is stopped. This line, like every command line, is limited to 250 characters. Type the learned line:

```
   X =0  Fact=1  X =X+1  Fact=Fact*X  Z  X
```

and then do two separate X commands. You might want to key in an H command before the second X command to turn your printer on. This will print a table of factorials from 2! to (1.70854E+9)! or so, if you wait long enough. Hit the ESC key twice to interrupt and abort the operation if you get tired of waiting. After the X command is executed, the F3 and F4 keys will restore the input line to the contents of the learned line instead of the previously typed command line.

Y => Delete (Yank) number from list:

The Y command removes the currently active item from the list and makes the next older item the active item. The age of an item is judged by when it was created. X is always the oldest item and is never removed from the list. If the Y command is executed, when X is the active item, X is not removed, but the youngest item becomes the active item. Thus, a long string of Y commands will always remove all items from the list except X.

Z => Output list:

The Z command will display the name and value of all items on the list. Some items may be found on the list that were not explicitly put there. The item Pi is put on the list by the P command and when needed by the trig functions. The item Ln10 = Ln(10) is put on the list when needed by the exponential functions. The items RN, RNA, and RNC are put on the list by the random number command N. The items RNA and RNC are put on the list by the random number function RN(X). The item File: "comment" is put on the list by the "comment" command. The item Lrn: <learned line> is put on the list by the X command. The items File: "comment" and the item Lrn: <learned line> also have a value associated with them, normally = 0.0. This value has no meaning and is not used.

@ => Substitute Log file for printer:

The @ primitive command will use the last entered comment as a file name and open this file as a text file for echoing screen output as a substitute for the printer. See the W command for the format of file names. See the H command for activating the echo output. If a comment has not been entered, the file name NoName.VPL is used. The VPLOut procedure can be used to give an override file name for the @ command. If the file already exists, output will be appended to it. The start of the file and the start of appended data is identified with "YY/MM/DD HH:MM:SS.SS NewLog ---...---". If the file does not already exist, a new Log file will be created. Each time the @ command is given the selection status of this option is reversed.

" => Start/Stop file name or comment:

Comments can be entered anywhere on the command line. The comment is started with a " mark. The comment is ended with a " mark or the end of the line. All spaces between the " marks become part of the comment. Comments are also used as file names, see the VPCIn, VPNIn, VPNOut, and VPLOut procedures. The item File: <comment> is put on the list by this "<comment>" command.

% => Set FMB = x, FMB on list:

The % primitive command is equivalent to FMB = x, where x is the currently active item. FMB stands for Floating Modulo Base. The / primitive command and the PowM(X' Y) function use FMB from the list.

/ => x = x Mod FMB, FMB on list:

The / primitive command replaces x with x modulo FMB, where x is the currently active item and FMB is an item on the list. If FMB is not on the list, it is added to the list with a value of zero. If FMB is zero, the value of x is not changed.

$ => Restart:

This reinitializes the program, the same as reloading from disk, except total running time is not reset, the history of previous operator entries is not cleared, the echo to printer or log file state is not changed, and the help menus are not automatically displayed. The parameters set by the D, E, M, T, U, and V commands are reset to their nominal values, and all items on the list are deleted except X and it is cleared. This also reset the random number generator.

> => Write configuration: Config.VPC:

The file Config.VPC is written to disk. It contains the VPCalc commands that will restore the configuration of VPCalc to its current state.

< => Read configuration: Config.VPC:

The file Config.VPC is read and run as a VPCalc code file. This will restore the configuration of VPCalc to its configuration when the file was written by the > command.

] => Write entry history: Hist.VPT:

The file Hist.VPT is written to disk. This is a text file and contains a copy of the current history of operator entries.

[=> Read entry history: Hist.VPT:

The file Hist.VPT is read and used to restore the history of operator entries with the history when the file was written by the] command. The current history is not cleared, but some or all of it may be lost since only 20 entries are saved.

F1 => Hot Help:

The function keys, F1 through F10, are not true primitive commands. They are meant to be used during the keying in of an input line. Press the F1 key and the following help menu will pop-up:

```
-------------------------------------------------------------------------------

          ╔═════════════════════ Help ══════════════════════╗
          ║   F1  => This help menu                          ║
          ║   F2  => Quit and exit to operating system*      ║
          ║   F3  => Restore previous input*                 ║
          ║   F4  => Restore previous input and accept*      ║
          ║   F5  => Active control characters for editing   ║
          ║   F6  => Status                                  ║
          ║   F7  => Primitive op codes                      ║
          ║   F8  => Infix operators                         ║
          ║   F9  => Procedures supported                    ║
          ║   F10 => Functions supported                     ║
          ║   ESC => Exit Help (* active on Command: line)   ║
          ╚══════════════════════════════════════════════════╝

-------------------------------------------------------------------------------
```

F2 => Quit and exit to operating system:

The F2 key will cause the program to exit back to the operating system, but a reprieve message is displayed first.

F3 => Restore previous input:

The F3 key normally will restore the command line to the value of the previously executed command line. After the B, K, or X command is executed, this key will restore the command line with the learned line. If the learned line changes, when it executes, the previous value of the learned line will be restored by this key.

F4 => Restore previous input and accept:

The F4 key is the same as F3 except that the previous command is executed without the Enter key being required.

F5 => Help with input key control:

Press the F5 key and the following message will appear:

```
-------------------------------------------------------------------------------

The active control characters for editing (^ = Ctrl, BS = Backspace):

 0) Down   or Up => Retrieve history of previous operator entries
 1) ^Right or ^F => Jump to beginning of next word
 2) ^Left  or ^A => Jump to beginning of previous word
 3) Right  or ^D => Retype the character at current position
 4) Left   or ^S => Back up a space and delete if inserting
 5) Del    or ^G => Delete the character at current position
 6) BS     or ^H => Delete the character to left of cursor
 7) End    or ^X => Jump to end of input
 8) Home   or ^E => Jump to beginning of input
 9) ^End   or ^Y => Clear input from current position to end
10) ^Home  or ^B => Clear input to left of cursor
11) PgDn   or ^T => Clear word to right
12) PgUp   or ^W => Clear word to left
13) Ins    or ^V => Toggle insert mode
14) Enter  or ^M => Accept the entire input as is
15) ^Enter or ^J => Accept input, truncate if not at beginning or end
16)           F2 => Quit and exit to operating system
17)           F3 => Restore previous input
18)           F4 => Restore previous input and accept
19)           F5 => This menu: Help with input key control

                        >>>> PRESS ESC TO EXIT HELP <<<<

-------------------------------------------------------------------------------
```

F6 => Status:

Press the F6 key and the "Status" message (see PAGE 1) will appear.

F7 => Primitive op codes:

Press the F7 key and the "Primitive op codes" message (see PAGE 4) will appear.

F8 => Infix operators:

Press the F8 key and the "Infix operators" message (see PAGE 4) will appear.

F9 => Procedures supported:

Press the F9 key and the "Procedures supported" message (see PAGE 5) will appear.

F10 => Functions supported:

Press the F10 key and the "Functions supported" message (see PAGE 5) will appear.

ESC => Interrupt a long process, Restore previous value and accept as input or Exit Help:

The ESC key is not a true primitive command, it is meant to be used after the program has been asked to perform a task that is taking longer than the operator is willing to wait. If not at the command input line, press the ESC key once and the message

```
   *** INTERRUPT:  To continue Press RETURN Key;
   To Abort Computation Press ESCAPE Key;
   To Set SoftAbort Flag Press SPACE Bar.
```

will appear. If the SPACE bar is pressed, the message

```
   SoftAbort flag set by operator!
```

will appear. If instead the ESC key is pressed again, the message

```
   Computation aborted by operator!
```

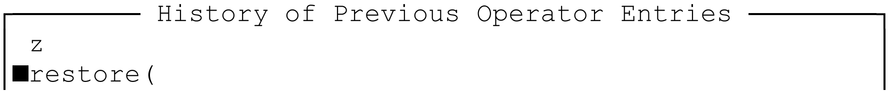
will appear, and if auto display is on, the value of the currently active item will be displayed, followed by the Command: prompt. If the ESC key is pressed during the display of a value, the same messages will appear, but if the second ESC is pressed, the trailing part of the value "E+xxx (xx) [xx]" is still displayed correctly. Pressing ESC twice during execution of the Z command causes all item on the list not yet displayed to be displayed to a small precision.

When the SoftAbort flag is set, the variable SoftAbort is put on the list and is set to a value of 1.0. Its purpose is to allow the operator to flag a VP Code file that it should gracefully terminate its operation.

During the keying in of an input line the ESC key is the same as F4, the previous command is executed without the Enter key being required. During Help, the ESC key is used to exit Help.

Down or Up => Retrieve history of previous operator entries:

The history of up to 20 previous operator entries are saved and can be retrieved by using the up and down arrow keys. Press one of these keys and the following help menu will pop-up:



```
┌──────── History of Previous Operator Entries ────────┐
│ z
│■restore(
      │■save(                                                   │
      │ run("b                                                  │
      │ vpcin(                                                  │
      │ b=456                                                   │
      │ c=789 d=321 e=789 f=888                                 │
      │■a = (12,345 + 2 * b * (c + d) / Sin(e + f)) ^ 2         │
      │ ]                                                       │
      └─> Press ESC, Enter, Up, Down, PgUp, PgDn, Ins, Del <────┘
--------------------------------------------------------------------------------
```

While this menu is up, use the Up, Down, PgUp, and PgDn keys to select a previous entry and then use the Enter key to accept it. The previous entry accepted will be put on the Command: prompt line, and then it can be edited before it is executed. The ESC key will remove the menu without changing the Command: prompt line. The Del key will delete the selected entry. The Ins key will toggle the locked status of an entry. When an entry is locked it cannot be deleted or scrolled off the top of the list. A small square ■ will be displayed to the left of a locked entry. At most 18 of up to 20 entries can be locked. This leaves room for at least the last 2 entries from the Command: line.

Infix operators -

Infix operators +, -, *, /, ^, @, #, %, \, &, |, <, =, >, <=, <>, and >= are the operators that appear between operands in an expression. Infix operators do not change the value of their operands, but produce a single result that can be used to further complete the evaluation of the expression that contains the infix operator. The infix operator precedence classes, from highest to lowest, are:

1) ^
2) *, /, @, #, %, \, &
3) +, -, |
4) <, =, >, <=, <>, =>

Operators of the same class are evaluated from left to right. Thus (2 * 10)^2 = 20^2, but 2 * 10^2 = 2 * 100. Also, A + B * C = A + (B * C).

A = X + Y => Set A to X plus Y:

Addition operator.

A = X - Y => Set A to X minus Y:

Subtraction operator.

A = X * Y => Set A to X times Y:

Multiplication operator.

A = X / Y => Set A to X divided by Y:

Division operator, error if y = 0.

A = X ^ Y => Set A to X to the power Y:

Exponential operator. This operator operates differently depending on whether Y is an exact integer. If Y is an exact integer, the peasants' method is used in which up to 2 * Log base 2 of Y multiplies of powers of X are done to compute the result. If Y is not an exact integer, the result is computed by Exp(Y * Ln(X)). An error message is generated in two cases: 1) X is < 0 and Y is not an integer. 2) X = 0 and Y is < 0. If X = 0 and Y = 0, an answer of 1.0 will be given.

A = Y @ X => Set A to ATan2(Y over X):

ArcTangent of Y over X operator. Used to find the Polar coordinates angle coordinate of the Cartesian coordinates (X, Y). If the degree mode is set, the answer, A, will be in the range -180 < A <= 180. If the radian mode is set, the answer will be in the range -Pi < A <= Pi. If both X and Y are zero, an answer of zero will be given.

A = X # Y => Set A to Mag(X' Y) = SqRt(Sq(X) + Sq(Y)):

Magnitude of (Y, X) operator. Used to find the Polar coordinates radius coordinate of the Cartesian coordinates (X, Y).

A = X % Y => Set A to Mod(X' Y) = X Modulo Y:

Modulo operator. X % Y = X - (Int(X/Y) * Y). Where Int(X/Y) is the integer part of X/Y. The sign of X % Y is equal to the sign of X. An error message is generated if Y = 0.

A = X \ Y => Set A to GCD(X' Y) = Greatest Common Divisor:

Greatest common divisor operator. Uses the oldest algorithm in the book, Euclid's algorithm (see Euclid's Elements, Book 7, Propositions 1 and 2). Only the integer parts of X and Y are used in the computation. For example, the GCD of 12 and 18 is 6.

A = X & Y => Set A to 1 if X and Y are not 0, else set A to 0:

Logical And operator. For all logical operations, 0.0 is considered False and all other values are considered True. When the result of a logical operation is True, the value 1.0 will be produced. When the result of a logical operation is False, the value 0.0 will be produced.

A = X | Y => Set A to 1 if X or Y, is not 0, else set A to 0:

Logical Or operator.

A = X < Y => Set A to 1 if X < Y, else set A to 0:

Numerical Less-than operator. For all numerical equivalence operators, the operands are considered as real numbers and the result is either 1.0 (True) or 0.0 (False).

A = X = Y => Set A to 1 if X = Y, else set A to 0:

Numerical Equal-to operator.

A = X > Y => Set A to 1 if X > Y, else set A to 0:

Numerical Greater-than operator.

A = X <= Y => Set A to 1 if X <= Y, else set A to 0:

Numerical Less-than-or-equal-to operator.

A = X <> Y => Set A to 1 if X <> Y, else set A to 0:

Numerical Not-equal-to operator.

A = X >= Y => Set A to 1 if X >= Y, else set A to 0:

Numerical Greater-than-or-equal-to operator.

Procedures -

Procedures are invoked by a statement starting with a procedure name followed by its argument. Arguments are numerical expressions that are evaluated before the procedure is performed. Procedures do not change the value of their arguments. For the procedures Write and WriteLn, arguments are optional and may be literal like: WriteLn("Now is the time"). For the procedure Next, arguments are not allowed.

AutoDisplay(X) => Set Auto display on if X <> 0, else off:

Same as the A primitive op code, but instead of being a toggle, sets Auto display on if X <> 0, and sets it off if X = 0.

ClearHist => Clear history of previous operator entries:

The history of up to 20 previous operator entries are saved and can be retrieved by using the up and down arrow keys. The ClearHist procedure removes all operator entries currently saved and makes this memory available to the calculator. Even though no argument is needed for this and some other procedures, it is usually better to use the parentheses, e.g., ClearHist() or ClearHist(to prevent unexpected results if the procedure name is misspelled.

Diag(X) => Set diagnostic mode on or off:

The diagnostic mode is turned on if X <> 0 and is turned off if X = 0. When the diagnostic mode is on, all command line executions will be timed by the computer clock and the time spent executing the command will be displayed. The timing data is displayed as:

```
   T = xxx.xx  DT = x.xx sec.  Start execution
        .
        .  <Command output, if any>
        .
   T = xxx.xx  DT = xx.xx sec.  End of execution
```

The DT value on the End of execution line is the time spent executing the command. The DT on the Start execution line is the time spent waiting for the operator to compose the command line. The T values are the total running time since the program was started and can only be reset by terminating and reentering the program from DOS.

EchoScreen(X) => Echo screen to printer/Log file, on or off:

Same as the H primitive op code, but instead of being a toggle, sets Echo screen to printer/Log file on if X <> 0, and sets it off if X = 0.

InputLines(X) => Set number of input lines (1, 2, 3, or 4):

Sets the number of lines in the command input field to X. This allows control of the length of the input field to 70, 150, 230, or 250 characters. The integer part of X is used, X larger than 4 implies 4, smaller than 1 implies 1. The length of the input field will always be large enough to hold the previous command line for the F3 function.

LogFile(X) => Substitute Log file for printer, on or off:

Same as the @ primitive op code, but instead of being a toggle, sets Substitute Log file for printer on if X <> 0, and sets it off if X = 0.

LX => LT => Restore LastTop to top of the list:

This sets the current active item equal to the value of the item named LastTop. If LastTop does not exist, it is created with a value of zero. Normally, before each command line is executed the value of the current active item is saved on the list in an item named LastTop. If a command line is entered that changes the value of the current active item, it can be restored to its previous value if the LX or LT procedure is performed immediately. This procedure should be entered on a command line by its self to prevent LastTop from being changed before it is retrieved.

The value of the current active item, Top, is not saved in LastTop if:

1) The command line is: LX
2) The command line is: LT
3) The command line is: LastTop=
4) The command line is: empty, i.e, <Enter> only
5) The SaveTop option is turned off by SaveTop(0).

Next => Move to next item on the list (no argument):

This changes which item on the list is the active item from the current active item to the next item from top to bottom. If X is the active item, which is always at the bottom of the list, the top item will become the active item. A command line with the single command Next followed by several F4 function keys will move through the whole list one item at a time. Note, this procedure does not take an argument.

ReadN(F) => Read file F = "ccc...c", F is optional:

This will use the argument F = "ccc...c" as a file name and read this file as a VPCalc formatted number and assign it to the item with the name stored in the file. This is the name it had when it was written. It is assumed that the file was created by the W command or the WriteN(F) procedure. See the W command for the format of file names. If no argument is given, and a comment has not been entered, the file name NoName.VPN is used. As files are input, the most significant bit of each byte read is set to zero to allow the files to be created or modified by a text editor that uses these upper bits as flag bits.

The ReadN proc differs from the J command in that the J command does not use the name stored in the file, but assigns the value read to the current active item. The ReadN command will not change the current active item unless an = sign is not found in the file or the name found is the same as the current active item.

Restore/Save => Restore or Save Configuration, History, & List:

Save will write the entry history file Hist.VPT like the] command, write the configuration file Config.VPC like the > command, write each items on the list to a separate file (Save0000.VPN, Save0001.VPN, ...), and write a VPCalc code file Restore.VPN that can be run by VPCalc to restore all of the saved items.

Restore will read the entry history file Hist.VPT like the [command, read the configuration file Config.VPC like the < command, and run the Restore.VPN restore file to read in each items that was on the list at save time. Restore does not clear the entry history or the list before it executes, so they may grow larger than they were at save time.

Run(F) => Run VPCalc code from file F, F is optional

This will use the argument F = "ccc...c" as a file name and read and run this file as a VPCalc code file. If no argument is given, defaults are like ReadN(F). To see examples of how VPCalc primitives, procedures, and functions are used, inspect the ---.VPC files (type or print). It will be noted that they are in plain DOS text.

SaveTop(X) => Set "save top value in LastTop" on or off:

This sets the "save top value in LastTop" option on if X <> 0, and sets it off if X = 0.

ScientificN(X) => Force scientific notation on iff X <> 0:

Normally numbers with less than 14 significant digits to the left and less than 14 to the right of the decimal point are displayed in fixed notation (e.g., 12.34). If the ScientificN(X) procedure is executed with X <> 0, all numbers will be displayed in scientific notation (e.g. 1.234 E+1 [14]). The normal method is restored after the ScientificN(X) procedure is executed with X = 0.

SetD(X) => Set max decimal digits in display:

The SetD(X) procedure sets the maximum number of decimal digits to display to the evaluated value of X. If this is set larger than the number of digits set by the M command minus the number of digits set by the T command, the smaller value will be used to determine the number of digits to display. This maximum only applies when the display is in scientific notation. The values set by the M and T commands are always carried as a multiple of seven (7), but the value set by the SetD(X) procedure can be any integer >= two (2). If this maximum is in effect, the last digit will not be rounded.

SetMax(X) => Set max decimal digits allowed in mantissa:

The SetMax(X) procedure sets the max decimal digits allowed in the mantissa of any value to the evaluated value of X. If X is not a multiple of 7, then the next higher multiple of 7 is used.

VPCIn(F) => Enter file name F = "ccc...c" for J command:

This establishes the file name of the VPCalc code file that will be read by the next J command. If the ("<filename>") is missing, the file name input with the last " comment command will be used.

VPLOut(F) => Enter file name F = "ccc...c" for @ command:

This establishes the file name of the VPCalc log file that will be opened by the next @ command that opens a file. If the ("<filename>") is missing, the file name input with the last " comment command will be used.

VPNIn(F) => Enter file name F = "ccc...c" for I command:

This establishes the file name of the VPCalc number file that will be read by the next I command. If the ("<filename>") is missing, the file name input with the last " comment command will be used.

VPNOut(F) => Enter file name F = "ccc...c" for W command:

This establishes the file name of the VPCalc number file that will be written by the next W command. If the ("<filename>") is missing, the file name input with the last " comment command will be used.

Write(X) => Output X, (X may be "ccc...c", X is optional):

The Write(X) procedure outputs the evaluated value of X to the console. The H and @ commands and the EchoScreen(X) and LogFile(X) procedures can be used to echo this output to the printer or the Log file.

WriteLn(X) => Write(X) and a line feed:

The WriteLn(X) procedure is the same as the Write(X) procedure except that the output generated is followed by an end-of-line indicator.

WriteN(F) => Write X to file F = "ccc...c", F is optional):

This will use the argument F = "ccc...c" as a file name and write the current active item as a VPCalc formatted number exactly like the W command. See the W command for the format of file names. If no argument is given, and a comment has not been entered, the file name NoName.VPN is used.

Functions -

Functions are used on the right hand side of an equation or assignment statement. Functions do not change the value of their arguments, but produce a single result that can be used to further complete the evaluation of the expression that contains the function reference. If a statement starts with a function reference like a procedure, then the function is evaluated and this value is assigned to the current active item.

Abs(X) = AbsoluteValue(X):

Absolute value function = |X|.

ACos(X) = ArcCoSine(X):

Inverse of Trigonometric CoSine function, error if |X| > 1. If the degree mode is set, the answer, A, will be in the range 0 <= A <= 180. If the radian mode is set, the answer will be in the range 0 <= A <= Pi.

ACosH(X) = ArcHyperbolicCoSine(X):

The positive inverse of Hyperbolic CoSine function, error if X < 1.

ASin(X) = ArcSin(X):

Inverse of Trigonometric Sine function, error if |X| > 1. If the degree mode is set, the answer, A, will be in the range -90 <= A <= 90. If the radian mode is set, the answer will be in the range -Pi/2 <= A <= Pi/2.

ASinH(X) = ArcHyperbolicSine(X):

Inverse of Hyperbolic Sine function.

ATan(X) = ArcTangent(X):

Inverse of Trigonometric Tangent function. If the degree mode is set, the answer, A, will be in the range -90 <= A <= 90. If the radian mode is set, the answer will be in the range -Pi/2 <= A <= Pi/2.

ATan2(Y' X) = ArcTangent(Y over X):

Trigonometric ArcTangent function. Used to find the Polar coordinates angle coordinate of the Cartesian coordinates (X, Y). If the degree mode is set, the answer, A, will be in the range -180 < A <= 180. If the radian mode is set, the answer will be in the range -Pi < A <= Pi. If both X and Y are zero, an answer of zero will be given.

ATanH(X) = ArcHyperbolicTangent(X):

Inverse of Hyperbolic Tangent function, error if|X| >= 1.

Cos(X) = CoSine(X):

Trigonometric CoSine function, error if |X| is very large.

CosH(X) = HyperbolicCoSine(X):

Hyperbolic CoSine function.

Exp(X) = eToThePower(X):

Evaluates to e raised to the X power, where e is the base of the natural logarithms. The item Ln10 = Ln(10) is put on the list when needed by the exponential functions. If Ln10 is not on the list, the file Ln10.VPN is read-in and Ln10 added to the list. If the file Ln10.VPN is not found, Ln10 is computed.

ExpL(X) = eToThePower(X) - 1:

Evaluates to one less than e raised to the X power, where e is the base of the natural logarithms. This function is needed when an expression contains Exp(X) - 1 and X can take on small values. ExpL(X) is accurate for small X.

Fac(X) = Factorial of Int(X):

Factorial function = 1 * 2 * 3 * ... * X. Only the integer portion of X is used in the calculation.

Frac(X) = FractionalPart(X):

Fractional part function. Frac(X) = X - Int(X).

GCD(X' Y) = Greatest Common Divisor:

Greatest common divisor function. Uses the oldest algorithm in the book, Euclid's algorithm (see Euclid's Elements, Book 7, Propositions 1 and 2). Only the integer parts of X and Y are used in the computation. For example, the GCD of 12 and 18 is 6.

Int(X) = IntegerPart(X):

Integer part function. For X >= 0, Int(X) is the largest integer less than or equal to X. Int(-X) = -Int(X);

Inv(X) = 1 / X:

Inverse or reciprocal function, 1.0 divided by X, error if X = 0.

Ln(X) = NaturalLog(X):

Evaluates to the Log base e of X, where e is the base of the natural logarithms, error if X <= 0.

LnL(X) = NaturalLog(X + 1):

Evaluates to the Log base e of (X + 1), where e is the base of the natural logarithms, error if X <= -1. This function is needed when an expression contains Ln(X + 1) and X can take on a value near zero. LnL(X) is accurate for values of X near zero.

Log(X) = LogBase10(X):

Evaluates to the Log base 10 of X, error if X <= 0.

Lop(X) = ReducePrecision(X):

This function evaluates to X with its least significant super-digit removed. If rounding is turned on, the removed super-digit is used to round into the new least significant super-digit. In VPCalc numbers are normalized from both sides. If a calculation results in a number with some trailing zero bytes, these bytes are removed by reducing the count of the number of bytes in the mantissa, and memory is reallocated. The Lop function can result in many bytes being removed if removing one byte results in many trailing zeros.

Mag(X' Y) = SqRt(Sq(X), Sq(Y)):

Magnitude of (Y, X) function Used to find the Polar coordinates radius coordinate of the Cartesian coordinates (X, Y).

Mod(X' Y) = X - (Int(X/Y) * Y):

Modulo function. Mod(X' Y) = X - (Int(X/Y) * Y). Where Int(X/Y) is the integer part of X/Y. The sign of Mod(X' Y) is equal to the sign of X. An error message is generated if Y = 0.

PowM(X' Y) = (X to the power Y) Mod FMB:

The Exponential function with modulo arithmetic. This function operates differently depending on whether Y is an exact integer. If Y is an exact integer, the peasants' method is used in which up to 2 * Log base 2 of Y multiplies of powers of X are done to compute the result. The Modulo process is performed after each multiply to prevent the intermediate results from becoming large. If Y is not an exact integer, the result is computed by Exp(Y * Ln(X)) Mod FMB. If FMB is not on the list, it is added to the list with a value of zero. If FMB is zero, the Modulo is not performed.

An error message is generated in two cases: 1) X is < 0 and Y is not an integer. 2) X = 0 and Y is < 0. If X = 0 and Y = 0, an answer of 1.0 will be given.

RN(X) = RandomNumber(Seed=X):

Random number function. RN(X) evaluates to a random number between zero and 1.0. This number will never have more than 35 decimal digits. Theoretically the random number generator will cycle after 10 ** 35 numbers, but the earth will not last that long. The fractional part of the argument of the function is taken as the seed of the random number generator. For a consecutive set of random numbers, the argument X should be the previous random number generated. The items RNA and RNC are put on the list by the random number function. The equation used is: RN(X) = (RNA * X * 10^35 + RNC) mod (10^35) / 10^35, where RNA and RNC are 35 digit integers.

Sin(X) = Sine(X):

Trigonometric Sine function, error if |X| is very large.

SinH(X) = HyperbolicSine(X)

Hyperbolic Sine function.

Sq(X) = X Squared:

The square function, X times X.

SqRt(X) = SquareRoot(X):

The positive square root function, error if X < 0.

Tan(X) = Tangent(X):

Trigonometric Tangent function, error if |X| is very large. It is also an error if X is equivalent to plus or minus 90 degrees.

TanH(X) = HyperbolicTangent(X):

Hyperbolic Tangent function.

ToDeg(X) = RadiansToDegrees(X):

Converts radians to degrees. Evaluates to X multiplied by 180/Pi.

ToRad(X) = DegreesToRadians(X):

Converts degrees to radians. Evaluates to X multiplied by Pi/180.

-X = Negative of X, 0 - X:

Negative inverse of X. The -, +, and ! functions do not require the parentheses so they also can be considered as unary or monadic operators.

+X = Positive of X, 0 + X:

The identity operator, +X = X.

!X = Not X, 0 -> 1 else 0:

Logical Not operator. Not X (!!X) will leave 0.0 alone and will change all other values to 1.0 (True).

If, GoTo, GoUpTo, Label, Continuation lines -

The following commands are primarily for use in VPCalc code files, but can be used from the Command: prompt line.

If Command:

The If command is the first word of an If statement. The syntax of the If statement is:

```
   If <expression> Then <statements> Else <statements>
```

The expression following the If is evaluated and if it is True, i.e., not zero, all statements on the same line following the next Else are deleted and execution continues with the statements following the Then. If the expression evaluates to zero (False), all statements following the expression up to the next Else are deleted and execution continues with the statements following the next Else. The Then key word is optional, the Then <statements> is optional and the Else <statements> is optional.

The equivalent of a case statement can be constructed for example like:

```
   If A=1 B=3 Else If A=2 B=5 Else If A=3 B=7 Else B=0
```

If A is an integer, this is equivalent to:

```
   B=0 If (1 <= A) & (A <= 3) Then B=2*A+1
```

GoTo Command:

The GoTo <label> command will skip all statements following the GoTo until <label>: is found and then start executing the statements following the <label>:. If the GoTo command is in a VPCalc code file, lines of input also will be skipped until the <label>: is found or until an end-of-file. If the line containing the GoTo is from the Command: prompt line, only statements on the current line will be skipped. It is not an error if the <label>: is not found, but a GoTo end-of-file or end-of-line will be performed in this case.

GoUpTo Command:

The GoUpTo <label> command will skip all statements following the start of the current line until <label>: is found and then start executing the statements following the <label>:. If the <label>: is not found on the current line and the GoUpTo command is in a VPCalc code file, the file will be reset to the first line of the file and lines of input will be skipped until the <label>: is found or until an end-of-file. If the line containing the GoUpTo is from the Command: prompt line, only statements on the current line will be skipped. It is not an error if the <label>: is not found, but a GoTo end-of-file or end-of-line will be performed in this case.

Labels:

A label is a name followed by a colon (:). When encountered as a command, a label is a no-op. When searching for where to go from a GoTo <label> or from a GoUpTo <label> command, the <label>: is used to determine where to restart execution. If duplicate labels are on a command line or in a code file, the first one encountered is the one that is effective.

Continuation lines:

Continuation lines are indicated by the last non-blank character of the line being a + or - character. A + says, this line is to be continued by adding the next line, but a blank character should be included between them if it is needed to separate fields. A - says, this line is to be continued by adding the next line, but no blank character should be included between them.

Batch Commands (Echo, @Echo, Pause, and Rem) -

The following commands are primarily for use in VPCalc code files, but can be used from the Command: prompt line.

Echo Command:

Normally, commands from a VPCalc code file are displayed on the screen as they are executed. This can be turned off by the Echo off command and turned on by the Echo on command. If something other than or more than on or off follow the word Echo, it is considered a message and is output to the screen.

@Echo Command:

The @Echo command is the same as the Echo command except that, if it is the first command on a line, it is executed before the command line is echoed to the screen. Thus, an @Echo off at the beginning of a line will do an Echo off without the command being echoed to the screen.

Pause Command:

The pause command will output the following message to the screen and wait for operator input of any key:

```
   Strike a key when ready . . . _
```

Everything on the line following the word Pause is considered a remark and is skipped. VPCalc code file processing can be interrupted by pressing the ESC key and can be terminated by pressing the ESC key twice.

Rem Command:

The syntax of the Rem command is Rem <remark>. It is a no-op command and everything on the line following the word Rem is skipped.

Transcendental Function Evaluation -

All transcendental functions, Sin(X), Cos(X), ASin(X), ACos(X), Tan(X), ATan(X), Exp(X), ExpL(X), Ln(X), LnL(X), Log(X), SinH(X), CosH(X), TanH(X), ASinH(X), ACosH(X), ATanH(X), and ATan2(Y' X), when they are evaluated, ends up using one of the four basic transcendental functions, Sin(X), ATan(X), ExpL(X), and LnL(X). The methods used by these four functions are quite similar: 1) For F(X), reduce the given argument X to a related argument f. 2) Further reduce f, NN times in a recursive loop to produce an argument g much smaller than f. 3) Evaluate the Taylor series for the argument g. 4) Reconstruct F(f) from F(g) by a recursive process executed NN times. 5) Reconstruct the desired function value F(X) from F(f).

The number NN in steps 2) and 4) is computed by a heuristic equation of the form NN = a + b * SqRt(M) where a and b are constants and M is the current max decimal digits in a mantissa. The best value of NN is the value that produces the smallest total execution time. After step 4) a best value of NN is computed and output by estimating a value of NN that would have made the running time of step 3) equal the sum of the running time of steps 2) and 4). Best NN = NN * SqRt(T3 / (T2 + T4)). Where Tn is the time to execute step n). This equation is based on T2 and T4 being proportional to NN and T3 being inversely proportional to NN.

If the operator wants to control the value on NN, he can enter a value on the list for item MSinNN, MATanNN, MExpLNN, MLnLNN to control the value used for NN in the Sin(X), ATan(X), ExpL(X), and LnL(X) functions respectively.

The recursive method used to reduce the argument for Sin(X) is based on the equation: Sin(X) = Sin(X/3) * (3 - 4 * Sq(Sin(X/3))). In step 2) f is divided by 3, NN times to produce g. In step 4) the recursion: S = S * (3 - 4 * Sq(S)), is performed NN times, where S is initially the value of Sin(g) produced in step 3) and the final value is Sin(f).

The recursive method used to reduce the argument for ATan(X) is based on the equation: Tan(X/2) = Tan(X) / (1 + SqRt(1+Sq(Tan(X))). In step 2) the recursion: T = T / (1 + SqRt(1 + Sq(T))), is performed NN times, where T is initially the value of f from step 1) and the final value of T is the value of g for step 3). In step 4) the angle value, A = ATan(g), produced in step 3) is multiplied by 2, NN times to produce ATan(f).

The recursive method used to reduce the argument for ExpL(X) is based on the equation: Exp(X) = Sq(Exp(X/2). In step 2) f is divided by 2, NN times to produce g. In step 4) the recursion: A = A * (2 + A), is performed NN times, where A is initially the value of ExpL(g) produced in step 3) and the final value is ExpL(f). The recursion A = A * (2 + A) is equivalent to, but more accurate than, the recursion E = Sq(E), where E = A + 1;

The recursive method used to reduce the argument for LnL(Y) is based on the equation: Ln(X) = 2 * Ln(SqRt(X)). In step 2) the recursion: Y = Y / (1 + SqRt(1 + Y)), is performed NN times, where Y is initially the value of f from step 1) and the final value of Y is the value of g for step 3). In step 4) the log value L = LnL(g) produced in step 3) is multiplied by 2, NN times to produce LnL(f). The recursion Y = Y / (1 + SqRt(1 + Y)) is equivalent to, but more accurate than, the recursion X = SqRt(X), where Y = X - 1;

If the diagnostic mode is on, the values computed for NN in the four subroutines MSin, MATan, MExpL, and MLnL are displayed, for example, as:

```
    MExpL: NN = 22.299
    MExpL: NN = 21
    Best   NN = 21.935 +/- 1.633
```

In this example the MExpL subroutine estimated NN to be 22.299. A value of NN = 21 was actually used (this is not 22 because the number being worked on was less than 3, the base number used to generate the heuristic equation). Based on the actual timing of the run, the best value for NN is computed to be 21.935. Due to the uncertainty of the timing, the Best NN could be off by + or - 1.633.

The Taylor series used for Sin(X) is:
Sin(X) = X - X^3 / 3! + X^5 / 5! ...

The Taylor series used for ATan(X) is:
ATan(X) = X - X^3 / 3 + X^5 / 5 ...

The Taylor series used for ExpL(X) is:

```
ExpL(X) = X + X^2 / 2! + X^3 / 3! ...
```

The Taylor series used for LnL(Y) is:

```
LnL(y) = Ln(1+y) = Ln((1+z)/(1-z)) = 2 * (z + z^3/3 + z^5/5 ...)
Where x = 1+y = (1+z) / (1-z),
      y = x-1 = 2 * z / (1-z),
      z = (x-1) / (x+1) = y / (2+z).
```

Other equations used to produce the transcendental functions:

Cos(X) = Sin(X + Pi/2).

Tan(X) = Sin(X) / SqRt(1 - Sq(Sin(x)), and change sign of Tan(X) if in 2nd or 3rd quadrant, but error if X is equivalent to plus or minus 90 degrees.

ASin(S) = ATan2(S, SqRt(1 - Sq(S)), but error if |S| > 1.

ACos(C) = ATan2(SqRt(1 - Sq(C), C), but error if |C| > 1.

Log(X) = Ln(X) / Ln(10), but error if X <= 0.

```
For X >= 0.1, SinH(X) = (Y - 1/Y) / 2, where Y = Exp(X),
for X <  0.1, SinH(X) = Y / (2 * SqRt(Y+1)), where Y = ExpL(2*X),
and SinH(-X) = -SinH(X).
```

CosH(X) = (Y + 1/Y) / 2, where Y = Exp(|X|).

```
TanH(X) = Y / (Y + 2), where Y = ExpL(2 * X),
and TanH(-X) = -TanH(X).
```

```
For X >= 0.1, ASinH(X) = Ln(X + SqRt(1 + Sq(X))),
for X <  0.1, ASinH(X) = LnL(X + Sq(X) / SqRt(1 + Sq(X))),
and ASinH(-X) = -ASinH(X).
```

ACosH(X) = Ln(X + SqRt(Sq(X) - 1)), but error if X < 1.

```
ATanH(X) = LnL(2 * X / (1 - X)), but error if |X| >= 1,
and ATanH(-X) = -ATanH(X).
```

Error reports -

There are many different error reports like

```
   Cannot divide by zero, continuing...
```

that are a result of directly or indirectly requesting an operation that cannot be performed. Another type of error is the syntax error, where a command cannot be interpreted. The syntax errors are:

```
   Error in function's argument: <name>(|<string>
   Error in function's 2nd argument: <name>(...'|<string>
   Exponent expected: ^<sign>|<string>
   Expression expected: (IF |<string>
Expression expected: (|<string>
Expression expected: <name>(|<string>
Factor expected: <op>|<string>
Input line continuation too long: |<string>
Simple Expression expected: <op>|<string>
Term expected: <op>|<string>
Unknown function: |<name>(<string>
Unknown operation, Command line discarded: |<string>
( expected: <name>|<string>
"File name expected: <name>(|<string>
```

The vertical bar | always shows the start of the string of characters that cannot be interpreted.

The end -

Report any errors by sending me a letter or call me at my home voice phone (408) 741-0406 evenings or weekends.

Harry J. Smith
19628 Via Monte Dr.
Saratoga, CA 95070

-Harry

--
| Harry J. Smith, 19628 Via Monte Dr., Saratoga, CA 95070-4522, USA
| Home Phone: 1 408 741-0406
| E-mail: hjsmithh@sbcglobal.net
| Web site: http://www.geocities.com/hjsmithh/
--