

# HP & MicroSoft BASICS

v2.0 / 09 sep 98 / greg goebel

\* This document is intended as a short overview of the history of contemporary BASICs and of the syntactic differences between HP and Microsoft BASIC.

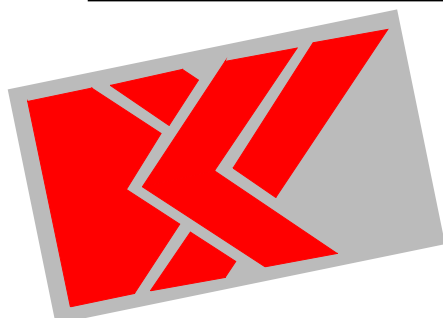
\* Revision history:

v1.0 / 03 jul 95 / gvg  
v1.1 / 17 oct 95 / gvg / Web update.  
v2.0 / 09 sep 98 / gvg / Condensation & simplification.

- 
- [1] OVERVIEW
  - [2] GENERAL DIFFERENCES
  - [3] CONTROL STRUCTURES
  - [4] FUNCTIONS & SUBPROGRAMS
  - [5] DATA DECLARATIONS
  - [6] MATH FUNCTIONS
  - [7] STRING FUNCTIONS
  - [8] TIME & DATE FUNCTIONS
  - [9] FILE-I/O STATEMENTS
  - [10] BASIC PLUS VERSUS VISUAL BASIC
- 

*Notes :*

*Ce document a été réalisé par le groupe responsable du support technique de la gamme des produits HP BASIC ou RMB (Rocky Mountain Basic) de la division MXD de Hewlett-Packard. Il est reproduit par BOURBAKY qui distribue l'ensemble des produits de cette gamme en France avec l'aimable autorisation de son auteur Greg Goebel.*



\*\*\*\*\*  
Cie : BOURBAKY  
BP 53  
13, Rue des Alpes  
07302 TOURNON Cedex - France  
Tel (Nat.) : 04 75 07 81 20      Tel (Int.) : +33 4 75 07 81 20  
Fax (Nat.) : 04 75 07 29 74      Fax (Int.) : +33 4 75 07 29 74  
  
web :      <<http://www.bourbaky.com>>  
e-mail : <[info@bourbaky.com](mailto:info@bourbaky.com)>  
\*\*\*\*\*

---

**[1] OVERVIEW**

\* The many dialects of the BASIC programming language are descendants of a simple computer language devised in the early 1960s at Dartmouth that was named Beginner's All-Purpose Symbolic Instruction Code -- BASIC. This early BASIC was very primitive command set, not too different BASICs that ran on early personal computers -- such as the Apple II, Commodore PET, and Tandy TRS-80.

\* As Hewlett-Packard (HP) developed controller computers in the mid 1970s, they first focused on a calculator-like language known as HPL. However, with the introduction of the 9845 controller in the late 1970s, HP introduced a BASIC language optimized for measurement and data analysis.

HP BASIC evolved down two main paths. One was followed by the Series-80 controllers, developed by Corvallis HP. This path began with the HP-85 and its BASIC, which became (with substantial additions) HP-86/87 BASIC, which in turn was ported to the ill-fated Integral PC as IPC Tech BASIC.

There were a number of related versions of this branch of BASIC, such as the BASICs for the HP-75 and HP-71 hand-held computers, and even a (very unsuccessful) port of IPC Tech BASIC to the Series-300 controller. But in the end all these BASICs died out.

The more successful path was followed by Fort Collins HP in their development of HP BASICs for the 9826, 9816, and 9836 controllers. This branch of BASICs was known as "Rocky Mountain BASIC" or RMB. The early RMBs evolved steadily into the HP BASIC versions for the Series-300 (S300) controllers, and were then ported to HP-UX. S300 HP BASIC was ported to the PC under DOS, using a separate coprocessor card, known as the Measurement Coprocessor, or MCP, that emulated an S300.

To keep HP BASIC up to date, HP introduced a set of software accessories named BASIC Plus to allow HP BASIC users to easily develop Windows-like user interfaces for HP BASIC programs, with pull-down menus, pop-up dialogs, bitmap-graphics panels, pushbuttons, sliders, and so on.

After the introduction of BASIC Plus, HP worked with Transera Corporation of Provo, Utah, to introduce a version of HP BASIC that ran under MS Windows, known as HP BASIC for Windows.

\* In the meantime, BASIC followed a different path of evolution on the PC, under different versions of Microsoft BASIC. The original version of Microsoft BASIC was very much like other dialects of BASIC available on PCs in the late 1970s. This BASIC was particularly important, however, because it was intimately associated with the IBM PC.

The IBM PCs all included a BASIC in ROM. If you didn't boot into the Disk Operating System (DOS), the PC came up into ROM BASIC. This ROM BASIC was rudimentary, and so an enhanced superset named BASICA was derived that used ROM BASIC as a kernel.

BASICA was popular but since nobody but IBM could use ROM BASIC in PCs, nobody but IBM could use it in their PCs. So Microsoft developed a compatible clone called GW ("gee-whiz") BASIC for other PCs. This was often shipped standard with DOS. Early HP Vectra PCs did not include it, but HP



did provide it separately under the name "Vectra BASIC".

Microsoft followed GW BASIC with a much cleaner version, known as QuickBASIC, which was very popular, since it was the first of the series to add modern programming-language features, a good development environment, and compile capability. It was so popular so popular that Microsoft even went so far as to create an extended Professional Development System (PDS) based on it.

In late versions of DOS, Microsoft also came up with a stripped-down version of QuickBASIC, known as QBASIC, that lacked compile capability and some other features. It was shipped standard with DOS.

When Windows 3 was introduced, Microsoft came up with a new Windows-oriented version of BASIC, known as Visual BASIC, that used QuickBASIC-like language syntax to build windowed application programs. Different versions of Visual BASIC are now available with extensions for advanced programmers.

Microsoft also came up with a souped-up version of QuickBASIC for DOS with Visual BASIC extensions, under the name Visual BASIC for DOS. They also devised a "macro" language called Visual BASIC for Applications to their popular software packages such as MS Word, Excel, and so on, and a stripped-down version of Visual BASIC known as VBScript that runs under Web browsers.

\* The strengths of HP BASIC relative to Microsoft BASIC are its math capabilities, particularly matrix and complex math; good capabilities for presenting and plotting data; excellent event handling capability; and convenient I/O capabilities.

Microsoft BASICs have their advantages as well. They have a better editing environment than HP BASIC, most allow compiled programs, have global constants, as well as a record mechanism.

Both types of BASIC have similar control constructs (like FOR-NEXT or SELECT-CASE) and roughly similar SUB and FUNCTION mechanisms. File-I/O is dissimilar, and graphics are totally unlike. While the widgets provided by BASIC Plus have roughly comparable equivalents in Visual BASIC, there is no syntactic equivalence between them.

The MS BASIC dialects include capability for serial I/O, but access to HPIB is only through add-on products, like the HP Standard Instrument Control Library (SICL) for Windows. These libraries do not operate at as high a level of abstraction as HP BASIC I/O statements, however, so converting I/O code can become laborious.

Further sections of this document perform a more detailed comparison between the languages. Printing and graphics are not covered in detail, however, as they differ not merely between HP and MS BASICs, but between DOS and Windows MS BASICs, and so are beyond the scope of this short document.



---

## [2] GENERAL DIFFERENCES

The first two things that you will notice when going from HP BASIC to MS BASIC are:

- \* MS BASIC does not use line numbers.
- \* A comment in HP BASIC is designated by a "!": ! This is a comment.
- \* A comment in MS BASIC is designated by a "'": ' This is a comment.

So the first two tasks when re-rewriting your HP BASIC program are to get rid of the line numbers, and change all the "!" marks to a "'". You can do this in any reasonable text editor.

The next thing you'll run into is variable names. While HP BASIC allows names of the format:

```
Data_val
```

-- that "\_" will give MS BASIC fits. You can change the variable name to:

```
DataVal
```

-- and it will work fine.

The next issue is data types. HP BASIC variables can be 16-bit integers, 64-bit reals, strings of characters, and arrays of all those. MS BASICs have more data types -- 16- and 32-bit integers, 32- and 64-bit reals, strings of characters, and arrays of these types.

This means that MS BASIC fortunately supports all the data types that HP BASIC does. However, by convention MS BASIC define variable types with a suffix character:

```
Int16var%      A 16-bit integer variable.
Int32var&      A 32-bit integer variable.
Real32var!     A 32-bit real variable.
Real64var#     A 64-bit real variable.
```

Leave off a suffix and you end up with a 32-bit real. That means that if you type the variable name:

```
MyVar%
```

-- which is a 16-bit integer, and later type in:

```
MyVar
```

-- by mistake, you have just entered an entirely different variable that is a 32-bit real. This can lead to some maddenly hard bugs to track down.



### [ 3 ] CONTROL STRUCTURES

\* MS BASIC control structures are equivalent to and syntactically similar to those of HP BASIC. The FOR loop:

```
FOR N=1 TO 20
  <statements>
NEXT N
```

-- is basically the same in both languages. However, there is an optional EXIT FOR statement in MS BASIC that allow you to bail out of the loop (you have to use a GOTO in HP BASIC).

The IF-THEN-ELSE construct is also basically the same in both languages:

```
IF Val=Target THEN
  <statements>
ELSE
  <statements>
END IF
```

However, nesting an IF-THEN-ELSE block is different in the two languages. In HP BASIC it is done as follows:

```
IF Val=Target1 THEN
  <statements>
ELSE
  IF Val=Target2 THEN
    <statements>
  ELSE
    <statements>
  END IF
END IF
```

You could do the same thing in MS BASIC, but MS BASIC also allows the cleaner syntax:

```
IF Val=Target1 THEN
  <statements>
ELSEIF Val=Target2 THEN
  <statements>
ELSE
  <statements>
END IF
```

The SELECT-CASE expression is similar in both languages as well; in HP BASIC it's written as follows:

```
SELECT Testval
CASE < Val1
  <statements>
CASE Val2, Val9
  <statements>
```



```
CASE Val3 TO Val8
  <statements>
CASE ELSE
  <statements>
END SELECT
```

The syntax for MS BASIC is only slightly different:

```
SELECT CASE Testval ' Note difference here.
CASE IS < Val1      ' Note difference here.
  <statements>
CASE Val2, Val9
  <statements>
CASE Val3 TO Val8
  <statements>
CASE ELSE
  <statements>
END SELECT
```

HP BASIC has three other looping constructs:

```
LOOP
  <statements>
EXIT IF <condition>
  <statements>
END LOOP
```

```
WHILE <condition>
  <statements>
END WHILE
```

```
REPEAT
  <statements>
UNTIL <condition>
```

MS BASIC replaces all three of these constructs with one construct of more flexible syntax:

```
DO
  <statements>
  IF <condition> THEN EXIT DO
  <statements>
LOOP
```

```
DO WHILE <condition>
  <statements>
LOOP
```

```
DO
  <statements>
LOOP UNTIL <condition>
```



**[4] FUNCTIONS & SUBPROGRAMS**

\* Both HP BASIC and MS BASIC support the GOTO and GOSUB-RETURN control constructs. As MS BASIC does not support line numbers, they have to use a label for a jump target. Labels are pretty much defined the same way in both languages:

```
GOTO Mylabel
...
Mylabel:
...
```

The old ON-GOTO and ON-GOSUB commands work the same in MS BASIC as they do in HP BASIC, but their use is discouraged (in favor of SELECT-CASE) in both languages.

Both languages support SUBs and FUNCTIONS, but the syntax is different. In HP BASIC, a SUB is defined as follows:

```
SUB Mysub (<parameters>
  <statements>
  SUBEXIT
  <statements>
SUBEND
```

-- where the parameters are defined as follows:

```
(INTEGER Int1, Int2, REAL A(*))
```

MS BASIC stands this syntax a little on its head:

```
SUB Mysub (<parameters>
  <statements>
  EXIT SUB
  <statements>
END SUB
```

-- where the parameters are defined as follows:

```
(Int1 AS INTEGER, Int2 AS INTEGER, A() AS DOUBLE)
```

\* Functions are also defined differently in the two languages. In HP BASIC a function is defined as follows:

```
DEF FNMyFunc(<parameters>)
  <statements>
  RETURN <value>
```



```
<statements>  
FNEND
```

In MS BASIC this is defined as:

```
FUNCTION MyFunc(<parameters>) AS TYPE <type>  
  <statements>  
  MyFunc = <value>  
EXIT FUNCTION  
  <statements>  
FNEND
```

Parameters are defined as for SUBs.



**[ 5 ] DATA DECLARATIONS**

\* Data type declarations differ slightly in both languages. In HP BASIC you would declare:

```
REAL Myvar1, Myvar2
INTEGER Somevar1, Somevar2
```

In MS BASIC it's:

```
DIM Myvar1 AS DOUBLE, Myvar2 AS DOUBLE
DIM Somevar1 AS INTEGER, Somevar2 AS INTEGER
```

In HP BASIC you declare arrays as follows:

```
DIM My_array(1:30)
INTEGER Int_array(1:4,1:100)
```

In MS BASIC this would be done as follows:

```
DIM MyArray (1 TO 30) AS DOUBLE
DIM IntArray (1 TO 4,1 To 100) AS INTEGER
```

You can also dimension arrays of strings:

```
DIM D(2,2) AS STRING
```

This creates an array of 32K-long strings; you can also specify a string of shorter length, if it so pleases you:

```
DIM D(2,2) AS STRING * 6
```

Where MS BASIC steals an advantage from HP BASIC is in common variable declarations. In HP BASIC you have to define a COM block to set up common variables:

```
COM /MyVars/ INTEGER Int1, Int2, REAL Reall
```

-- and then this block has to be copied in any subprogram or function that uses it. In MS BASIC you can set up true global variables:

```
COMMON SHARED Int1 AS INTEGER, Int2 AS INTEGER, Reall AS REAL
```

Data can be initialized in HP BASIC and the DOS versions of MS BASIC with



the DATA ... READ statement:

```
DATA 1,2,3
READ Var1,Var2,Var3
```

However, Visual BASIC for Windows does not support DATA-READ. Unlike HP BASIC, you can also declare constants in MS BASIC:

```
CONST YES = -1
CONST NO = 0
```

These constants are global -- available in any part of the MS BASIC program -- and so can be used to simplify some constructs that are inconvenient in HP BASIC.

You can also define data structures (at varying levels of complexity, depending on the particular MS BASIC dialect), which also allows you to work around some limitations of HP BASIC:

```
TYPE Vector
  I AS SINGLE
  J AS SINGLE
  K AS SINGLE
END Vector
```



**[ 6 ] MATH FUNCTIONS**

\* Math functions in HP BASIC and MS BASIC are similar, though HP BASIC has a larger set. Both support the same fundamental functions:

+        -        \*        /        ^

MS BASIC also supports an integer division operator:

\

Under HP BASIC this operation is performed by the DIV function. The following table matches the other familiar math functions in the two languages:

HP BASIC	MSB	description
ABS(N)	ABS(N)	Absolute value.
EXP(N)	EXP(N)	Exponentiation (e^N).
FRACT(N)	--	Use (V - INT(V)).
INT(N)	FIX(N)	Integer portion of floating-point number.
LGT(N)	--	Base-10 log -- use (LOG(N)/2.3025851).
LOG(N)	LOG(N)	Natural log.
MOD(N)	MOD(N)	Modulo (remainder) operation.
PI	--	Define CONST PI = 3.141592654.
SGN(N)	SGN(N)	Signum function (-1 if < 0, 0 if = 0, 1 if > 1).
SQR(N)	SQR(N)	Square root.
ACS(N)	--	Arccosine; use (ATN(N/SQR(1 - N*N))).
ASN(N)	--	Arcsin; use (ATN(N/SQR(1 - N*N*)) + PI/2).
ATN(N)	ATN(N)	Arctan.
COS(N)	COS(N)	Cosine.
SIN(N)	SIN(N)	Sine.
TAN(N)	TAN(N)	Tangent.

Note that trig functions in MS BASIC always expect and return values in radians. If you have a program that you can't easily convert from degrees to radians, you can define a function to perform the conversion, or define a set of functions to perform the trig operations in degrees:

```
FUNCTION DSIN (N AS DOUBLE) AS DOUBLE
  RETURN SIN(PI*N/180)
END FUNCTION
```



In contrast, HP BASIC allows you to set the trig mode:

```
DEG    ! Set degrees mode.  
RAD    ! Set radians mode.
```

Random number functions are the same in both languages. You use RANDOMIZE to seed the random-number generator and RND to get a random number in the range 0 to 1.

Both languages use the same comparison operators:

```
=      <      >      <>      =>      =<
```

-- and the same logical operators:

```
NOT     AND     OR     XOR
```

A TRUE value in HP BASIC is 1, while a FALSE is 0; in MS BASIC TRUE is -1, while FALSE is 0. Note that in HP BASIC the bitwise logical operators are different functions -- BINAND, BINIOR, BINEOR, BINCMP -- from the logical operators, while in MS BASIC logical operators can also be used for bitwise operations.

Note that MS BASIC does not support complex nor matrix math -- you have to build libraries of functions to perform these tasks.



---

## [7] STRING FUNCTIONS

\* String operations in MS BASIC are very different from string operations in HP BASIC. For the most fundamental example, string concatenation is performed under HP BASIC with the "&" character:

```
Cmdstr$ = "VALUE =" & VAL$(My_var)
```

In MS BASIC, you use the "+" character for string concatenation:

```
CmdStr$ = "VALUE =" + STR$(MyVar)
```

As these two contrasting examples show, the HP BASIC VAL\$ statement (which converts a numeric variable into a string representation) translates into the equivalent MS BASIC STR\$ statement. In general, string functions in HP BASIC and MS BASIC have this kind of equivalence: they have functions that operate in an almost identical fashion but have, for the most part, different names, as the table below shows:

---

HP BASIC	MSB	function
CHR\$(N)	CHR\$(N%)	Convert ASCII code into character.
IVAL\$(N,8)	OCT\$(N%)	Convert number to octal string.
IVAL\$(N,16)	HEX\$(N%)	Convert number to hex string.
LEN(S\$)	LEN(S\$)	Get length of string.
LWC\$(U\$)	LCASE\$(U\$)	Convert string to lower case.
NUM(C\$)	ASC(C\$)	Convert character to ASCII code.
POS(S\$,S\$)	INSTR(T\$,S\$)	Get position of substring.
RPT\$(S\$,N)	STRING\$(N,S\$)	Repeat string.
TRIM\$(S\$)	LTRIM\$(RTRIM\$(S\$))	Trim leading and trailing spaces.
UPC\$(L\$)	UCASE\$(L\$)	Convert string to upper case.
VAL(S\$)	VAL(S\$)	Convert numeric string to value.
VAL\$(N)	STR\$(N%)	Convert value to numeric string.

---

Note that MS BASIC also has a string function similar to RPT\$ that only repeats string characters:

```
PRINT SPACE$(80)
```

-- prints 80 spaces. Beware that it is an easy mistake to enter this as "SPACES\$", but the form is singular, not plural: "SPACE\$".

\* One major way in which HP BASIC and MS BASIC string handling differs is in dealing with substrings. In HP BASIC, you can access substrings using



subscripts; for example:

```
PRINT S${3,6}
```

-- prints the substring of S\$ starting at character 3 and ending at character 6; while:

```
PRINT S${3;6}
```

-- prints the 6 characters following character 3. In MS BASIC, you have to use a function, MID\$, to get a substring:

```
PRINT MID$(S$, 3, 6)
```

-- prints the 6 characters following character 3. A variation on this syntax is also used to replace substrings; for example:

```
MID$(S$, 6, 1) = Ch$
```

-- replaces character 6 of S\$ with the character in Ch\$. MS BASIC also has two other functions related to MID\$, one to access substrings from the start of the string:

```
PRINT LEFT$(S$, N%)
```

-- and one to access substrings from the end of the string:

```
PRINT RIGHT$(S$, N%)
```

This last is actually somewhat handy, since in HP BASIC you would have to do this operation as follows:

```
PRINT S${LEN(S$)-N;N}
```



**[ 8 ] TIME & DATE FUNCTIONS**

\* HP BASIC time and date functions center around a function named TIMEDATE, which gives the number of seconds since some ancient date. TIMEDATE can be used to provide time and date as follows:

```
PRINT TIME$(TIMEDATE)
PRINT DATE$(TIMEDATE)
```

MS BASIC has similar but separate functions: TIMER gives the number of seconds since midnight, TIME\$ gives the current time, and DATE\$ gives the current date.

HP BASIC allows a program to delay for a specified period using the WAIT statement:

```
WAIT 10
```

This causes a ten-second wait. The equivalent MS BASIC statement (for DOS) is SLEEP:

```
SLEEP 10
```

This is not supported under Visual BASIC for Windows. Note that there is an MS BASIC statement named WAIT, but it performs an entirely different function from the HP BASIC statement named WAIT; the MS BASIC WAIT function waits for a value to be input from an interface port.

For the most part, MS BASICs have coarser timing resolution than HP BASIC.



---

**[9] FILE-I/O STATEMENTS**

\* File-I/O in MS BASIC is very different from HP BASIC, and a detailed discussion is way beyond the scope of this document. We'll focus on the most useful case: writing and reading a text file.

Under HP BASIC, you create and open a text file as follows:

```
CREATE "MYFILE.TXT",1
ASSIGN @F TO "MYFILE.TXT";FORMAT ON
```

The "@F" is a "handle" that can be then used to access the file:

```
OUTPUT @F; "This is a test!"
OUTPUT @F; V1,V2,V3
```

When done, the file is then closed:

```
ASSIGN @F TO *
```

To read the file, you simply ASSIGN to it again:

```
ASSIGN @F TO "MYFILE.TXT";FORMAT ON
```

-- and then read from it with ENTER:

```
ENTER @F;S$
ENTER @F;V1,V2,V3
```

HP BASIC can determine the end of a file by perform an ON END jump:

```
ON END @F GOTO Endofit
```

\* Under MS BASIC you create a file simply by opening it:

```
OPEN "MYFILE.TXT" FOR OUTPUT AS #1
```

You then PRINT to it using "#1" as a handle:

```
PRINT #1,"This is a test!"
PRINT #1,V1,V2,V3
```

-- and close it as follows:



```
CLOSE #1
```

You can then open it for reading with:

```
OPEN "MYFILE.TXT" FOR INPUT AS #1
```

-- and read from it with LINE INPUT and INPUT:

```
LINE INPUT #1,S$  
INPUT #1,V1,V2,V3
```

MS BASIC has a variation on INPUT called INPUT\$ that reads a specified number of characters from a file:

```
S$ = INPUT$(6,#1)
```

MS BASIC can determine the end of a file by checking with the EOF statement:

```
DO WHILE NOT EOF(1)  
  LINE INPUT #1,S$  
LOOP
```

\* Note that you can test for the existence of a file under HP BASIC by trying to ASSIGN to the file and checking for an error:

```
CLEAR ERROR  
ON ERROR GOTO Nofile  
ASSIGN @F TO "MYFILE.TXT";FORMAT ON  
Nofile: !  
OFF ERROR  
IF ERRN<>0 THEN PRINT "File doesn't exist!"
```

You can do pretty much the same thing under MS BASIC, except that you must open the file for INPUT for this to work, even if you intend to write to it. If you open it for OUTPUT, you'll simply create the file and no error will occur:

```
ErrFlag% = 0  
ON ERROR GOTO Errtrap  
OPEN "MYFILE.TXT" FOR INPUT AS #1  
CLOSE #1  
ON ERROR GOTO 0  
IF ErrFlag%<>0 THEN PRINT "File doesn't exist!"  
...  
Errtrap:
```



---

```
ErrFlag%=1  
RESUME NEXT
```

Note how the error handling is performed in the two languages.

\* HP BASIC and MS BASIC have roughly equivalent file-system control statements:

---

HP BASIC	MSB	function
MSI	CHDIR	Change directory.
CREATE DIR	MKDIR	Make a directory.
PURGE	RMDIR	Remove a directory.
CAT	FILES	List files.
RENAME	NAME	Rename files.
PURGE	KILL	Delete files.

---



---

**[10] BASIC PLUS VERSUS VISUAL BASIC**

\* The BASIC Plus user interface tools are programmed entirely differently from Visual BASIC components, but there is a partial equivalence between the two sets. BASIC Plus includes the following pop-up dialogues:

---

INFORMATION	Displays information, user responds with mouse button.
ERROR	Displays error message, user responds mouse button.
QUESTION	Displays question, user responds with mouse button.
WARNING	Displays warning, user responds mouse button.
NUMBER	Allows input of numeric data.
KEYPAD	Similar to NUMBER, but provides a calculator keypad.
STRING	Prompts user for text string.
COMBO	Allows entry by text input or from list.
LIST	Allows user to select from a list of items.
FILE	Allows user to select a file from a directory.

---

-- and the following widgets:

---

PANEL	Provides a "panel" to place other widgets.
SEPARATOR	Allows line to be drawn to separate regions on PANEL.
PUSHBUTTON	Provides "pushbutton" that can be clicked by mouse.
TOGGLEBUTTON	Similar to PUSHBUTTON.
RADIOBUTTON	Similar to PUSHBUTTON.
LABEL	Allows labels to be placed on user interface.
NUMBER	Allows input of numeric data.
KEYPAD	Similar to NUMBER, but provides a calculator keypad.
STRING	Allows string to be entered from user interface.
COMBO	Allows string entry or selection from a list of items.
LIST	Allows selection from a list of items.
FILE	Allows selection of disk volume, directory, or file.
BAR	Single-bar bargraph.
BARS	Multiple-bar bargraph.
METER	Meter.
LIMITS	Similar to a METER but the needle moves along a bar.
PRINTER	Allows output of multiple lines of text.

---



---

SLIDER	Allows numeric input with mouse using slider.
SCROLLBAR	Simplified version of SLIDER, used for scrolling.
XYGRAPH	Allows drawing of X-Y graphs.
STRIPCHART	Allows drawing of stripcharts.
HPGLVIEW	Allows viewing of HPGL files.
BITMAP	Allows display of .BMP or .XWD bitmap files.
PULLDOWN MENU	Used for building pulldown menu systems.
CASCADE MENU	Used for building lower-level pulldown menus.
MENU BUTTON	Provides menu selection in pulldown or cascade menus.
MENU TOGGLE	Similar to MENU BUTTON.
MENU SEPARATOR	Allows line to be drawn in menus as item separator.
SYSTEM	Used to build user interfaces & widget arrays.

---

From the point of view of the default set of dialogs and widgets, VBWIN is roughly comparability to BASIC Plus, with some things missing and some things added. VBWIN has similar capabilities for dialogs:

- \* The "MsgBox" dialog allows the program to display a message (using a variety of formats); this is much like the BASIC Plus INFORMATION dialog and its sisters.
- \* The "InputBox" dialog allows the program to get text input from the user; this is much like the BASIC Plus STRING dialog.
- \* You can also specify the standard dialog boxes provided by Windows for a VBWIN program: "Open", "Save As", "Print", "Color", "Font".

The "Open" and "Save As" dialogs are comparable to the BASIC Plus FILE dialog.

- \* Finally, you can construct custom dialogs using the available VBWIN tools.

The default control set is as follows:

---

Picture Box Image	Used for displaying images; roughly equivalent to BASIC Plus display object.
Command Button	Same as BASIC Plus BUTTON widget.
Option Button	Similar to BASIC Plus RADIO BUTTON widget.
Check Box	Similar to BASIC Plus RADIO BUTTON widget.
List Box	Same as BASIC Plus LIST widget.
Combo Box	Same as BASIC Plus COMBO widget.

---



Drive List Box	Equivalent to BASIC Plus FILE widget.
File List Box	
Directory List Box	
Label	Same as BASIC Plus LABEL widget.
Text Box	Same as BASIC Plus STRING widget (no editor).
Vertical Scroll Bar	Same as BASIC Plus SCROLLBAR widgets.
Horizontal Scroll Bar	
Timer	Roughly similar to BASIC Plus CLOCK widget.

---

There are no menu controls in VBWIN as such; you create menus -- totally equivalent to those in BASIC Plus -- using a tool.

Other elements in the default set do not have BASIC Plus equivalents:

---

Frame	Draws frame around controls.
Grid	Defines spreadsheet-like grid.
Shape	Allows creation of various block shapes on the VBWIN display.
Line	Allows drawing lines on the BASIC Plus display.
Data	Provides database access.
OLE	OLE control.

---

All controls in VBWIN have attributes just as they do in BASIC Plus; but the syntax for the two sets are totally dissimilar. As noted before, you have to rebuild the user interface from scratch when moving from BASIC Plus to Visual BASIC.

Visual BASIC does not have by default the graphics tools available to HP BASIC Plus. However, it is possible to obtain additional tools for Visual BASIC, known as ActiveX Controls, that could in principle add these capabilities.



\*\*\*\*\*  
Cie : BOURBAKY  
BP 53  
13, Rue des Alpes  
07302 TOURNON Cedex - France  
Tel (Nat.) : 04 75 07 81 20      Tel (Int.) : +33 4 75 07 81 20  
Fax (Nat.) : 04 75 07 29 74      Fax (Int.) : +33 4 75 07 29 74  
  
web :      <<http://www.bourbaky.com>>  
e-mail : <[info@bourbaky.com](mailto:info@bourbaky.com)>  
\*\*\*\*\*

---

## **HP & MicroSoft BASICS**

---

v2.0 / 09 sep 98 / greg goebel