Decimal BASIC Help

by Kazuo SHIRAISHI

English Translations obtained from
Google Translate (http://translate.google.com/)
by Benjamin Robert Tubb (23-25 Jan 2011)
brtubb@pdmusic.org
Revision 2 (30 Jan 2011)

About Decimal BASIC

Terms of Use and Copyright

> (Tentative) Decimal BASIC is a program that was created as an educational and research purposes
> BASIC book using the research results obtained, please be sure to publish
> Bugs (or bug), who always found, please contact
> Does not restrict redistribution. Can not restrict the publication of reference books and commentary
> Quality is guaranteed. Responsible for the execution result

[Author]
> SHIRAISHI Kazuo

[Contacts]
> kazuo.shiraishi@nifty.com
> Or
> Bunkyo University Faculty of Education,
> Shiraishi Kazuo 343-8511
> Ogishima Minami Koshigaya, 3337

Overview

Overview

Decimal BASIC is a programming language for computing.
It implements almost all of the core module, the graphics module, the module module, and the individual character input module of ISO Full BASIC.
This help is an overview. Precise description of the language specification is descrived in Detailed specification, which describes what differ from the ISO standard.

Sorry, this help is under translating, so that many pages remain written in Japanese.

System requirements and file organization

System Requirement and file organization

[System Requirement]

Machine
PC/AT Comatibles
OS
Windows95/98/Me/NT4.0/2000/XP/Vista
Internet Explorer 5.5 or later required.

[File organization]

The execution file is BASIC.EXE.
BASIC.CHM, BASIC.KW1, BASIC.KW2, BASIC.KWS, BASIC.KWF should be put in the folder where BASIC.EXE exists.
BASFILE.ICO is required for file association.
The other files do not affect on exectution of BASIC.EXE.

This system does not change Windows system, so that you can carry this system put in a floppy disk.
Refference

Once BASIC.EXE is executed, BASIC.INI is made on the folder where BASIC.EXE is put.
When a program is runned, BASIC.BAK should be made. This file shall be deleted the program finished running.

SETUP.BAT registers BASIC to the START Menu of Windows, and let BASIC.EXE use the Windows registry to save its settings.
Detail of SETUP.BAT

Supplementation: BASIC.HLP and BASIC.CNT, which are included in the previous version of Decimal BASIC, are no longer needed.

[Uninstall]

To Uninstall Decimal BASIC, only remove the installed files, provided that if SETUP.BAT was executed, execute UNSETUP.-BAT and remove the files.

SETUP.BAT

SETUP.BAT

SETUP.BAT
SETOP.BAT enters BASIC in the start menu,
associate the extention ".BAS" to BASIC.EXE, ask whether shortcut should be made on the desktop, and make BASIC.EXE save option settings in the Windows registry.

UNSETUP.BAT
UNSETUP.BAT removes the settings made by SETUP.BAT.

On Windows NT/2000/XP, SETUP.BAT should be launched with full administrator privileges.

Note
On associating, SETUP.BAT registers two actions of "Edit" and "Run".
In case of "Edit", if BASIC.EXE is executed by association, it starts with the file open.
In case of "Run", if BASIC.EXE is executed by association, it runs the program and then terminates.

The action when a file with extention ".BAS" is double-clicked probably is "Edit". If this differs from your desire, use the tool option of Explorer to change the setting.

Note2
The setting whether the registory be used or not is registered on the BASIC.INI placed where BASIC.EXE started. If BASIC.INI was deleted by mistake, Execute SETUP.BAT again.

Copyright

Terms of Use and Copyright

(Tentative) Decimal BASIC is a program that was created as an educational and research purposes.
BASIC book using the research results obtained, please be sure to publish.
Bugs (or bug), who found, please contact us without fail.
Does not restrict redistribution. Can not restrict the publication of reference books and commentary.
Quality is guaranteed. Responsible for the execution result.

[The author]
Shiraishi Kazuo
[Contact]
kazuo.shiraishi @ nifty.com
Bunkyo University Faculty of Education,
Shiraishi Kazuo 343-8511 or
Ogishima Minami Koshigaya, 3337

■ Web pages

Decimal BASIC Web pages

Decimal BASIC Web pages
@ Decimal BASIC Web
@http://hp.vector.co.jp/authors/VA008683//english/index.htm

We have Japanese Web board
@Decimal BASIC Web Board

Introduction

■ PRINT

Introduction to BASIC
PRINT statement

◇ PRINT statement
To display the results using the print statement.
You can write multiple expressions separated by commas.
2-3 +3 two cases to calculate and display the results.

```
10 PRINT 2+3, 2-3
20 END
```

PRINT statement is also used to display the string.
String around the "" enclose.
Cases the screen "Hello!" To display.

```
10 PRINT "Hello!"
20 END
```

Cases of "2 +3 =" and the numerical results show 2 +3.

```
10 PRINT "2+3=", 2+3
20 END
```

[Operation] F9 and the program is run press.
Run from the menu "Run" may be selected.
<Note>
The program may be written in uppercase or lowercase writing, not use double-byte characters.
In other words, "PRINT" is, "Print" can "print" it is good, "PRINT" is useless.
The state's standard of BASIC, print the corrected automatically capitalizes the lowercase words and other features.
<Reference> PRINT statement details

■ Calculation

Introduction to BASIC
Calculate

◇ addition, subtraction, multiplication and division

BASIC, addition, subtraction, multiplication, division, respectively ,+,-,* / expressed.

2 ÷ × 3 and 3 cases to compute the two programs
10 PRINT 3 * 2,3 / 2
20 END

◇ power

In BASIC, ab to represent a ^ b.

Program to compute the fifth power of 2 cases
10 PRINT 2 ^ 5
20 END

◇ Order of operations

BASIC, powers, multiplication and division are performed in the order of addition and subtraction operations.

The priority calculation is performed from left to right.
Example
10 PRINT 1 +3 * 2 ^ 3
20 PRINT 2 ^ 3 ^ 3
30 END
When you run,
25
512
Be. 3 ^ 2 ^ 3 (3 ^ 2) ^ 3 Note that interpreted.

◇ complex equation

BASIC uses parentheses to alter the order of operations () is used.

Parentheses may be used in many layers, all written in parentheses.
Example
10 PRINT ((2 +3) * 4) ^ 2
20 END

◇ square root

In BASIC, a positive square root of the SQR (a) as written.

Find the square root of two cases of positive
10 PRINT SQR (2)
20 END
<Reference> Built-in Functions

BASIC of the JIS is created with the goal that meets the specifications. Therefore, power calculations and SQR can be considered as if the error of addition, subtraction, multiplication and division functions.

BASIC irregular because of the power calculation error in the square, three squares a * a, a * a * a may be written as it is recommended that, in this power calculation using the BASIC There is no reason to challenge.

■ Variables

Introduction to BASIC

Variable

◇ variable
You can temporarily leave the internal storage of the result.
Variable that is used for that storage location.
Variable, a, b, c, x, y and identifying and named.
I let you assign a value to a variable using a statement.
Using a variable cases
10 LET a = 2 +3
20 PRINT a-1
30 END

◇ let statement
The assignment is to calculate the value on the right, assigned to the variable results are written on the left side.

For example, the variables before executing the next line in the program a value of 20 is a 10, 20 let the line after the statement, a value would be 11.
Example
10 LET a = 10
20 LET a = a +1
30 PRINT a
40 END

<Supplement>
20 let line statement, a +1 to calculate the value, and then, to assign a result.
The variable is assigned a new value, the value of the yuan is extinguished. <Reference> let sentence

◇ input sentence
By using input statements can assign values to variables when you run the program from the keyboard.
If you list more than two variables, writing them, separated by commas.
At run time, type a comma-separated numbers as needed.
Example
10 INPUT a, b
20 PRINT a + b
30 END

Using character ◇
In BASIC, PRINT, LET, INPUT function words such as capital, may be written in either case.
Variable names are case-sensitive, may use either of the case. Differences in the case of variable names are ignored. For example, A and a is the same variable.
<Caution> double-byte characters (LET, etc.) can not be used to describe the program.

■ FOR ... NEXT

Introduction to BASIC

FOR ~ NEXT

◇ FOR ~ NEXT (1)
Cases x = 0,0.1,0.2, ..., about 1 SQR (x) to calculate

10 FOR x = 0 TO 1 STEP 0.1
20 PRINT x, SQR (x)
30 NEXT x

40 END
When you run a line of 10 for sentence, first, x is assigned to 0, go to line 20.
When you run a line 30 next statement, x 10 rows back to 0.1 is added.
This is repeated until the x value is greater than one.
<Supplement> indent
20 print line that says to indent the statements.
As for writing on the readability.
This is called indent.
Indentation does not affect program execution.

◇ FOR ~ NEXT (2)
statements for one step may be omitted.
Cases x = 1,2,3, ..., for calculating 10 x ^ 2.

```
10 FOR x = 1 TO 10
20 PRINT x, x ^ 2
30 NEXT x
40 END
```

Details for ~ next

■ IF ... END IF

Introduction to BASIC

IF ~ END IF

◇ IF ~ END IF
When conditions are met - to run each line.
Two cases of real solutions of equations

```
10 INPUT a, b, c
20 LET D = b ^ 2-4 * a * c
30 IF D> = 0 THEN
40 PRINT (-b-SQR (D)) / (2 * a), (-b + SQR (D)) / (2 * a)
50 END IF
60 END
```

◇ IF ~ ELSE ~ END IF

Changing the sentence to run if conditions are met.
Two cases of real solutions of equations

```
10 INPUT a, b, c
20 LET D = b ^ 2-4 * a * c
30 IF D> = 0 THEN
40 PRINT (-b-SQR (D)) / (2 * a), (-b + SQR (D)) / (2 * a)
50 ELSE
60 PRINT "no solution"
70 END IF
80 END
```

◇ inequality
In BASIC, ≦ inequality, ≧, ≠, respectively, to substitute,<=,>=,><.

if ~ end if more

■ DO ... LOOP

Introduction to BASIC

DO ~ LOOP

◇ DO ~ LOOP
EXIT DO statement is executed before, DO LOOP to repeat the sentence and the line written between the lines.
Repeat until you enter the six cases of "2 × 3 is?" to prompt to display.

```
10 DO
20 PRINT "2 × 3 is?"
30 INPUT n
40 IF n = 6 THEN EXIT DO
50 LOOP
60 END
```

<Supplement> IF statement
As on 40 lines, IF THEN condition can be written, followed by only one simple statement. In this case, IF END do not write.

do ~ loop detail

DEF

Introduction to BASIC

DEF statement

◇ DEF statement
DEF can be defined as the use of statements and functions.
Example
100 DEF f (x) = 3 * x ^ 3-2 * x ^ 2 + x
110 PRINT f (1), f (2), f (3)
120 END

<more>

Quotients and remainders

Introduction to BASIC
quotient and remainder

◇ quotient of integer division
Integer a integer quotient when divided by b, INT (a / b) required by.
INT (x), x-represents the largest integer not exceeding.
if x is a positive number, INT (x) is equivalent to x decimal places rounded.

◇ remainder of integer division
Integer a integer remainder when divided by b, ab * INT (a / b) that is, this MOD (a, b) can be found at.

◇ expansion of the definition of division
Dividend and a positive number, and to seek an integer quotient, it is possible to extend the range of real integer division.
a, b and when real, b> 0, it
a = bq + r, ≦ 0 r <b
Determined q and r integers satisfying as the real one. In this case, q, r the quotient is divided by a to b, that too.

In BASIC, a quotient of a divided by b, the remainder INT (a / b), MOD (a, b) can be found at.

Cases when x is 1.8 liters of liquid divided liter bottle of stuff, ask for a volume of 1.8 obtained by liquid becomes irregular and the number of surplus liter bottle.
10 INPUT x
20 PRINT INT (x/1.8), MOD (x, 1.8)
30 END

■ Trigonometric functions

Introduction to BASIC
Trigonometric function

◇ OPTION ANGLE DEGREES
OPTION ANGLE DEGREES write at the beginning of the program, the size of the unit at the corner of trigonometric functions (degrees) would.
Example
10 OPTION ANGLE DEGREES
20 PRINT SIN (30), COS (30), TAN (30)
30 END

SIN (30), COS (30), TAN (30), respectively, sin30 °, cos30 °, tan30 ° would represent.

[Operation] shift key while pressing the F7 key is inserted OPTION ANGLE DEGREES.

◇ inverse cosine

angle θ and cosθ = x ACOS (x) can be determined by. However, the resulting angle, 0 ° 180 ° from a range.

The length of the sides to use the cosine theorem cases a, b, c ∠ A request from the size.

10 OPTION ANGLE DEGREES
20 INPUT a, b, c
30 PRINT ACOS ((b ^ 2 + c ^ 2-a ^ 2) / (2 * b * c))
40 END

Inverse sine function ◇, arc tangent

sinθ = x, tanθ = x value of θ and, respectively, ASIN (x), ATN (x) required by. The resulting angle, -90 ° 90 ° from a among.

■ Logarithmic functions

Introduction to BASIC
Logarithm

◇ log
BASIC is the logarithm to base 10 LOG10 (x), logarithm to base 2 LOG2 (x), e = 2.71828 ... the logarithm function LOG (x) has been provided.
If you need a logarithm to the base, otherwise the number may be as follows.
Logarithm to the base 8 when three cases.

```
10 DEF log (a, x) = LOG10 (x) / LOG10 (a)
20 PRINT log (3,8)
30 END
```

<Supplement> BASIC does, DEF has been allowed to define a function built-in functions with the same statement.
In that case, DEF definition of priority statement.

■ Curves and Graphs

Introduction to BASIC
Curves and Graphs

◇ coordinate
When drawing graphs of functions and curves, the drawing area to set the coordinate system.
coordinate the range of x from x1 to x2, y coordinates range from y1 to y2 when you assign to the drawing area,
SET WINDOW x1, x2, y1, y2
Run.

◇ draw axes
To draw axes
DRAW axes
Used.

◇ plot Statement
To draw the plot points and lines used in the statement.
PLOT LINES: x, y
When you run the point (x, y) is drawn to the point.
End; writing
PLOT LINES: x, y;
You, then run between the plot lines are tied in with the specified statement.
Without specifying the coordinates, just
PLOT LINES
And write only to cancel the work between the line segments between specified points before.

Example

```
10 DEF f (x) = x ^ 3-x
20 SET WINDOW -4,4, -4,4
30 DRAW axes
40 FOR x =- 4 TO 4 STEP 0.01
50 PLOT LINES: x, f (x);
60 NEXT x
70 END
```

<Supplement>
BASIC does this, LINES: allowing for the omission of.
For example, a line of cases over 50
50 PLOT x, f (x)
May be written.

■ Graphics

Introduction to BASIC
Basic Graphics

◇ Set the coordinate system
set window to set the coordinates for the statement.
Format

SET WINDOW left, right, bottom, top
left left x-coordinate
right right x-coordinate
bottom bottom y coordinate
top top y coordinate
Example
SET WINDOW -4,4,0,8
x-coordinate range -4 ~ 4, y 0 coordinate a range of 8 - to set.
<Note>
When properly draw a shape,
right-left = top - bottom
So as to define the scope.

Draw the axes ◇.
When there is a need to draw the axis is defined to use the picture embedded axes.
Also, when you draw a grid instead of the axis is defined using the picture of the embedded grid.
Format
DRAW axes
DRAW grid
If you want a value other than an interval scale, we use the following form.
DRAW axes (x, y)
DRAW grid (x, y)
x, y are numeric expressions.

Curve ◇.
The curve order PLOT LINES.

Satoshi Hara curve
There are devices called XY plotter.

The curve drawn by the pen.
Move the pen on paper. Then lines are drawn with pen on paper facing.
Penn State has risen (the paper away from the state) does not draw anything you can move the pen.
I hope that 描点 pen
描点 is on paper that is in contact with the pen
That it is not turned away from the paper 描点.

The form and meaning

PLOT LINES: x, y point (x, y) 描点 to move, turn off the 描点.
PLOT LINES: x, y; point (x, y) 描点 to move, to turn on 描点.
Turn off 描点 PLOT LINES.
The state just after the execution of the program 描点 is always off.

◇ cases
(A) Draw a graph of the function.

```
10 DEF f (x) = x ^ 3 +2 * x +1
20 SET WINDOW -5,5, -5,5
30 DRAW grid
40 FOR x =- 5 TO 5 STEP 0.01
50 PLOT LINES: x, f (x);
60 NEXT x
70 END
```

(2) multiple curves.

```
10 SET WINDOW -5,5, -5,5
20 FOR x =- 5 TO 5 STEP 0.01
30 PLOT LINES: x, sin (x);
40 NEXT x
50 PLOT LINES
60 FOR x =- 5 TO 5 STEP 0.01
70 PLOT LINES: x, cos (x);
80 NEXT x
90 END
```

Connected to the two curves Shimawanai 2 ◇, 50 PLOT LINES lines of writing.

(3) parametric

```
10 option angle degrees
20 SET WINDOW -4,4, -4,4
30 DRAW axes
40 FOR t = 0 TO 360
50 PLOT LINES: 3 * cos (t), 2 * sin (t);
60 NEXT t
70 END
```

(4) Polar

```
10 SET WINDOW -2,2, -2,2
20 DRAW axes
30 FOR t = 0 TO 2 * pi STEP pi/180
40 LET r = 1 + cos (t)
50 PLOT LINES: r * cos (t), r * sin (t);
60 NEXT t
70 END
```

To convert rectangular coordinates to polar ◇ will be as follows.

```
LET r = SQR (x ^ 2 + y ^ 2)
LET t = ANGLE (x, y)
```

■ Exception Handling

Introduction to BASIC
Exception handling state

◇ WHEN ~ END WHEN syntax
When drawing graphs and functions, and have not defined the middle point, they stopped with an error in the program there.
BASIC, such as division of zero exception error occurs at run time.
Exception handling and use will be able to continue the program even when an error occurs, such as zero divide.

Cases y = 1 / x draw a graph of

```
10 SET WINDOW -4,4, -4,4
20 FOR x =- 4 TO 4 STEP 0.01
30 WHEN EXCEPTION IN
```

```
40 PLOT LINES: x, 1 / x;
50 USE
60 PLOT LINES
70 END WHEN
80 NEXT x
90 END
```

In a typical program, for it was the statement x to a value of 0
PLOT LINES: x, 1 / x
When you run, the program stops with an error there.
As the above example, the statement may result in errors
PLOT LINES: x, 1 / x
WHEN EXCEPTION IN USE and the writing and enclosed in and when the error occurred, USE END WHEN line and proceed to execute the statement was written between the lines.
The 60 lines in the above example, x = LINES PLOT intent is to ensure that there would be tied around the curve of 0.

<Exception Details>

■ Arrays

Introduction to BASIC
Array

◇ sequence
Multiple variables at once, it was arranged so that you can see by the number of individual variables called indices. Each variable is called a subscripted variable.
In order to use an array, it is necessary to declare the array name and subscript range. The declaration of an array using the DIM statement.

```
10 DIM A (4)
```

Written declaration to the beginning of the program, such as after, A (1), A (2), A (3), A (4) be able to use the four subscripted variables.
Subscripted variables can be specified by the index number formula. For example,

```
20 LET N = 2
30 LET A (N) = 10
```

How could such.

When you declare an array
DIM A (0 TO 10)
You can also specify a lower and upper limits of the range of indices and so on.

10-point increments determine the distribution of test scores of a scale of 100 cases. When you enter a negative end.

```
100 DIM m (0 TO 10)
110 FOR i = 0 TO 10
120 LET m (i) = 0
130 NEXT i
140 DO
150 INPUT x
160 IF x <0 THEN EXIT DO
170 LET i = INT (x/10)
180 LET m (i) = m (i) +1
190 LOOP
200 PRINT "one point", "person"
210 FOR i = 0 TO 10
220 PRINT i * 10, m (i)
230 NEXT i
240 END
```

■ Sample programs

Introduction to BASIC
Using the sample program

You can load a sample program to use to read the File menu.
If you have understood the meaning of function words, together with the cursor to the left and press F1 to display the corresponding help.
You can choose to step from the Run menu, F10 can be performed in sequence by pressing Hazime Aya.
If you want to interrupt the program execution is suspended from the Run menu or select, Ctrl key while pressing the B press.

Function folder contains examples of using built-in functions.
statemen folder, BASIC contains examples of statements.
sample folder, BASIC contains application examples.

■ PDF version of INTRODUCTION to Full BASIC

Introduction to BASIC
PDF version of "JIS Full BASIC Getting Started "

(Tentative) Decimal BASIC is, PDF version of "JIS Full BASIC the introduction"(tutorial.pdf) are bundled.

tutorial.pdf to view and print Adobe Acrobat Reader is required. The Reader Acrobat, Adobe's site
http://www.adobe.co.jp/product/acrobat/readstep.html
Please get it from.
The paper size is B5.

References

Introduction to BASIC
Reference book

1. BASIC Application of Mathematics

"Mathematics and Computers 1" ISBN4-627-03580-2
"Mathematics and Computers 2" ISBN4-627-03590-X
Morikita published work by Sato Hiroshi Shiraishi Kazuo Takahashi Masanobu

"Using Computer Graphics" by Nobu Shiraishi Katagiri Shigeru 和夫
Tokyo Denki University Publication Bureau ISBN4-501-52590-8

"Playing simple computer programming" by Kimura Yoshio
Kodansha Blue Backs ISBN4-06-257398-9

"Open the PC world of wonders" by Iidaka Shigeru
Iwanamisyoten ISBN4-00-500483-0

2. BASIC Programming

"Full BASIC algorithm by 通論" by Satoshi Masao Takeiti Masato I Moriguti Shigekazu
University of Tokyo Press ISBN4-13-062129-7

3. JIS Full BASIC standards Aya Tamotsu

"Computer programming languages Full BASIC JIS X 3003"
Japan Standards Association

4. BASIC formation process of

"Back to BASIC" JGKemeny & TEKurtz Author, translated by Matsuda Takeo
Satoru Satoru Press
It is hard to find publisher for extinction. Library, bookstore, etc. Please look.

Manipulation

■ On help

Using Help

Press the F1 key to display context sensitive help screens.
○ F1 program while editing, press
Word of the selected state (s) have, as a key search word.
If there is no word in the selected state, as a key word search for the current cursor position.

Place the cursor on the menu item displays the corresponding help ○ F1 is pressed.

<Note>
Instructions on attaching □ important.
Beginner □ □ □
□ □ Star
Advanced □
Please use as a rough guide.

Moving Cursors

Operations

Move the cursor ○
Home to the top of the line
End to end
Ctrl-Home to the document head
Ctrl-End Document End

Insert mode and Overwrite mode switching ○
Insert key

Menu Selections ○ (if operated from a keyboard)
Main Menu Selections
you have to press alt text in the hold down brackets.
Select sub-menu
To invert the arrow keys move the piece and press Enter.
May be key in parentheses Kakareta character (alt required)
Cancel menu selection
Esc key

○ character selection
Shift and characters you can choose key while pressing the arrow keys. Selected character, the Edit menu "Cut", "copy", "delete", "Search", "replace" the help "search" subject to.

○ move between regions in the dialog box
Tab key

Function keys

Function Keys

I've assigned the following functions: a function key.
F1 Help
F2 function word
Find Next F3

shift-F5 ~ shift-F8
User-defined (optional - function key) to insert text

F9 Run
Stepping F10

File Menu

File menu

Tadashi Arata
The current program is to create programs Please choose a different time.
Open
Load the program that is stored on disk.
Read to the end
Add to the program and save it as an external library function definitions.
Close
Close the frontmost program.
Save
Save to update the current record.
Save as Name
Name and save.
Print
Print the program.
If you close without clicking Ok, you can just change the settings of the printer without printing.
BASIC End
BASIC close.

<Notes>
Press Alt + F File menu, you can also press the Start key.
<Supplement>
Programs and library files will be saved as a text file. Windows word processing format and read in common, such as Notepad.)
<Note>
NoName be saved in a file name that contains a spell that is not possible.

Run Menu

Run menu

Run
This command is executed normally.

Stepping
Enters debug state after each single statement.
<Reference> Using the Debugging Window

Break (Ctrl + B)
Suspends execution of the program ready for debugging.
After that, it is possible to continue running the program.

If you choose to freeze or suspend Troubleshooting Steps

Window Menu

View Menu

Text Output
Graphics
Trace Log
If you want to use the window is not displayed.
Purposes can also be used to switch the window to display on top.

Toolbar
Each click on the toolbar display, hidden alternates.
Not appear correctly on the toolbar icon Trouble

Button Frame
View button on the toolbar frame / hide.

Tile
Cascade
Used to adjust the display of child windows.

Help Menu

Help Menu

Search (S)
Word (s) and select the state to execute the search as the key word.
If there is no word in the selected state, as a key word search for the current cursor position.
(F1 is the same press)

Edit menu

Cut Copy Paste

Edit Menu

Cut
Copy
Paste
Delete
Undo
Select all

"Cut", "copy", "delete" before you run
You must select a character to advance.
How to choose a character (one of the following)
(1) Hold down the left mouse button, trace.
(2) Shift key while pressing the arrow keys.
(3) "Select all" to run.

"Cut" is executed, the selected characters are deleted, Windows will move to the clipboard.
"Copy" to run the selected character is copied into Windows clipboard.
"Paste" to run, Windows will insert the data in the clipboard.
"Delete" is executed, the selected characters are deleted. Clipboard does not change.
"Undo" when you run, run before the "cut", "Paste", "Delete" to cancel the undo operation.

"Cut" and "paste" and use, and move.
"Copy" and "paste" and use, you can copy.

"Copy", "Paste" to use editing features, you can to get data to and from other software.

Search Replace

Edit Menu

Search
Substitution

Usually, the search from the beginning of the text, select the text to state that contains at least one end of the line search, or select Replace to find the range.
<Note> Modify the text, move the cursor to "Find Next" to run will cause deviation in the search.

If you check the Match whole word, BASIC Search by word in accordance with the grammar. However, the point will recognize as a word separator. And notes, DATA statement, IMAGE line and is not supported.

Search term is found to close the dialog.

Insert Keywords

Edit Menu

Function word insertion

When working with a mouse
A simple tip is displayed when you click the mouse.
Imprinted with the words double-click feature.
Help will be displayed by clicking the Help button.

When the keyboard
Start the F2 key.
Statement with the arrow keys, function, please select one of the characters.
Press the Tab key once.
Since then, the objective function words Find any of the following methods.
(1) until the appearance of function words to be inserted, press the key several times to the first letter of the desired function words.
(2) If the word is unknown function, press the space bar until you see the genre you want.
(3) arrow keys to scroll.
When you see the desired function words, Enter words are imprinted with the function key.
If you want to help screen for the selected function words, F1 key please.
Press ESC to exit.
<Notes>
Function words that appear in the list is intended only to pick up the language commonly used functions.

<Reference>
Customizing the Insert function words

<Trouble>
F2 will be killed by pressing

- **Add line numbers**

Edit Menu

Adding line numbers
Adds line numbers. You can specify the initial value and increment.
There is already a line containing the line number, line after line numbers from the wave to its value. However, its value is not less than the previous row number of the line, adding line numbers will fail.

Remove line numbers
Remove the line numbers.
However, GOTO line number references are not removed from the sentences.

- **Change case**

Change Case Change
Change the casing of a particular word based on pre-registered list. However, string constants, comments, data elements (DATA part of the contents of the statement) and is unchanged.
Customizing the casing changes

- **Commentize**

Of annotation
Exclamation point at the beginning of each line of the selection (!) Is inserted.

Annotation de-
Remove the exclamation point at the beginning of each line of the selection.

- **Wordwrap**

Edit Menu

Word Wrap
When you open a program file is not displayed.

Option Menu

Syntax

Options Menu

Grammar

Standard (JIS Full BASIC)
Admit implicit array declaration.
If there is no option base statement is a lower bound of the array.

Impose a variable declaration
BASIC, IDH_DECLARE statement DECLARE NUMERIC DECLARE STRING statement can declare all the variables used.
If you check the force variable declaration error in the translation function names and variable names are not declared. This feature is effective to reduce bugs caused by typos in variable names.

Old JIS-compatible Basic BASIC
Implicitly recognizes the array declaration.
If there is no option base statement, the lower bound of the array is 0.

Microsoft BASIC compatible
Implicitly recognizes the array declaration.
If there is no option base statement, the lower bound of the array is 0.
LET statements, or other multi-optional and enables the Microsoft grammar. Microsoft enabled grammar, almost, is the extent of the AutoCorrect menu. However, when using the variable name of the BASIC language is a function of LET is required.
Ignore the specified numeric variable of type double precision (about 16 binary digits) is.
AND, OR, NOT logical operations and results of the comparison operators such as ,<,<=, Makoto Tokino -1, 0 if false will be a number (but a comparison of a string expression to write the statement PRINT and especially not supported.)
Disable the built-in functions PI. PI is able to run the program for use as a variable.
MOD operator \ Microsoft interprets the operator sense.
INPUT "A ="; A INPUT statement in the form of work according to Microsoft BASIC.
Be able to use LPRINT statement.
Pixel coordinate system is introduced with the origin at the start of run left end.
You can use the following form rendering instructions.
Below the lower part of writing a numeric expression.
If the parameter is also optional.
SCREEN n

n is one of 2,3,11,12,87.
Black White 0 1 640 2 × 400
640 3 × 400 ... 0 Black 1 Blue 2 Red 7 White, 7 foreground
Black White 0 1 640 11 × 480
640 12 × 2 Green Blue Black 0 1 White 480 ... 7, 7 foreground
640 87 × 2 Green Blue Black 0 1 White 400 ... 7, 7 foreground
If you ignore the values listed above.
CLS n
Regardless of the value of n, CLEAR statement is executed.
WINDOW (x1, y2) - (x2, y1)
COLOR c
COLOR,,, c
PSET (a, b)
PSET (a, b), c
LINE - (x, y)
LINE (x1, y1) - (x2, y2)
LINE (x1, y1) - (x2, y2), c, B
LINE (x1, y1) - (x2, y2), c, BF
PAINT (x, y), c, d
CIRCLE (x, y), r, c,,, s
CIRCLE (x, y), r, c,,, s, F, t
Start angle, end angle can not write
Other works in principle, every statement JIS statements follow the corresponding action.
String handling and compatibility of files.
[Memo]
If you want graphics compatible with N88-BASIC, the beginning of the program SCREEN 2 (black) or SCREEN 3 (color) to run.

- **Numeric values**

Options Menu

Option number
Specify the number of treatment option arithmetic when writing a sentence.

Digit 15 (JIS) mode
Variable holds the number is 15 decimal digits.
Numerical calculation formula is as follows.
Result of addition, subtraction, multiplication and division are rounded to the digits 19 to 27 decimal digits.
Powers and functions of the force results are rounded to 17 decimal digits.

(However, in column number 17 is even)
If you assign a numeric variable results of a numeric expression, rounded to 15 digits.
Results are rounded to the 15 digits.
If you check the number of digits to display ○, as it displays the results.

Column mode 1000
1000 decimal digits or less significant figures have the exact value.
Results are rounded to the decimal digits of 1000 or 1008 digits.
Rounding is truncated.
Trigonometric, exponential, logarithmic and transcendental functions will not work.
Power index
-9223372036854775807 To 9223372036854775807 (= 263-1)
Limited to integers.
Exponent is not an integer, an error at run time.
Transcendental functions ○ (17 digits) to use a check,
Trigonometric, exponential, and logarithmic and transcendental functions are available and the results of those 17 digits of precision (the tail is even) will be.
The operation of power is not an integer in the range when the exponent is the precision of 17 digits (trailing even) is.

Approximately 16-digit binary mode
The smallest positive number is about 5.0E-324, the maximum number is approximately 1.7E308.
It may be expressed in decimal accurate, and faster.
-9007199254740992 - 9007199254740992 (= 253) integer can be represented exactly.
In the form of an integer power of 10, 1 ~ 1E36 can be represented accurately to. On the other hand, 0.1,0.01, such as the decimal fraction, in most cases can not be represented accurately. If you can not accurately represent the number of programs written in the approximate number representation possible. For example, the 0.1 is slightly larger than the number is approximately the true value. Displays the result back to decimal again.
Results are usually rounded to 15 digits.
If you check the number of digits to display ○, 19 positions will show up.

Complex number mode
Plurality of binary numbers in two (approximately 16 characters) are represented by pairs.
To view and print imaginary statement and print two numbers in parentheses.
Power operations, when the base is positive, the imaginary can be exponential.
Also, when the index is an integer from -2147483647 to 2147483647, the imaginary can be a bottom.
ABS (x), SQR (x), EXP (x), LOG (x) is extended to a complex domain.
As a function of the complex mode only, COMPLEX (x, y), RE (x), IM (x), CONJ (x), ARG (x) are available.
SCALE and SHIFT functions of the shape deformation is extended to the complex.
Using the above in a non imaginary, an error occurs at run time.

Mode of Satoshi Tamotsu
Multi-digit calculations can be rational.
Displays the result in improper fraction form.
However, input or DATA statements can not use this form.
Power operations, the power index is limited to integers from -2147483647 to 2147483647.
Trigonometric, exponential, logarithmic and transcendental functions, such as SQR function, PI function is used.
Transcendental functions ○ (17 digits) to use a check,
Trigonometric, logarithmic and exponential functions and transcendental functions SQR, PI function is used and the results of these 17 digits (even at the end) (PI functions around 1000 digits) are accurate.
<Reference>
OPTION ARITHMETIC Statement
Complex rational

- **Graphics**

Options Menu

Graphics

Image output format
The bitmap image format suitable for widespread use. The result can be saved as GIF or JPEG in.
Metafile (specify size) is selected to create a meta file size specified.
Metafile (printer) to create a meta file size of the printable area of the paper specified in the printer settings.
Printer (direct) is selected, the drawing instructions are sent directly to the printer.

Chosen a metafile or printer, please make sure the printer settings before running the program.
Metafile (printer) or printer (direct) When selected, the slide bar to add the margin, the margin of the default printer, you can even specify the additional margin. The print position "on the received" and specifies that when choosing a vertical form, DEVICE VIEWPORT sets the initial value of the upper area of the square.

Reference
The format of the output image "metafile" or "printer (direct)" when a, a printer driver to specify a PostScript printer you can create EPS files. EPS files are recorded in the form of drawing steps.

Steps for Creating EPS

<Reference> Graphics Metafile Printer

<Note>
To use the metafile, which must be installed on any printer. Even if you are not actually connected to a printer, please leave at least one printer driver installed.

- **Compatibilities**

Options Menu

Compatibility

Grammar
JIS when you want to change the statement to conform the provisions of the error.

Operation
- Format character
How to interpret the character as you want the provisions of JIS format, please change this option.

* Invalid arguments related to the shape SET statement
SET WINODW example, when you try to set the coordinates of the statement has zero width or height, shape-related parameters set the behavior of the statement is incorrect according to the provisions of JIS If you want to change this option.
However, if you change this option, because it will continue to ignore any errors, less likely to notice errors in the program.

- String processing unit
BASIC This process corresponds to a string of Chinese characters. Therefore, assuming the program was created to be run as a byte string processing unit may be damaged. If so, please change this option.
Change the behavior of the instruction by setting this option,
CHARACTER INPUT statement, substring processing, POS functions, ORD function, CHR $ function, LEN function.
"Running as a character string processing unit" is the extended instruction
OPTION CHARACTER KANJI
To
"Running as a byte string processing unit" is the extended instruction
OPTION CHARACTER BYTE
Are supported.
OPTION CHARACTER program units written statement specifying OPTION CHARACTER statement takes precedence.
<Reference> Chinese

· DEF of arguments
"Also used by reference" is selected, and when the variable argument DEF statement will be passed by reference, the program will run a little faster.

The variable own · FOR ~ NEXT
FOR ~ NEXT BASIC does not see the limits and holds a variable increment. Decide where to put these two variables.

A dash dotted line width
BASIC is to draw a dotted line of the dashed line when the width of a cosmetic pen is used, the line width greater than 1 when the geometric pen is used. Therefore, when a line width varies with the design of a line drawn when two or more. When to avoid it, please change your settings.
However, Windows95/98/Me does change settings is useless.

Draw a line
Windows should draw a line, start painting, painted in the end leaves. Full BASIC PLOT LINES This behavior does not match the action, and this is the Windows BASIC line drawing instructions on the reverse. Changing this option can be a positive change in the order line drawing. However, the format of the output image "bitmap (screen)", then this is specified regardless of the reverse.

Drawing to the screen
In that Winodows Vista Aero is disabled, a pane on the screen does not reflect PLOT POINTS drawing.
When this problem occurs, "Vista Non-Aero fault tolerance," Please check.

<Operations>
How to work with the keyboard
Grammar with the arrow keys, please choose one of the operation.
Please go to the Tab key to the desired area.
Please change the up and down arrow keys.
Press Enter to finish.

- **Auto correct on compiling**

Options Menu

AutoCorrect
Modify the specified items during translation.

Microsoft BASIC syntax
Specifies whether to automatically correct the grammar for programs written in Microsoft.
The setting for this field will be stored until the next start.

This matches the BASIC variable names in function words, LET automatic insertion does not work.
¥ mod operator and the operator and ask how to fix at actually occurred in the program.
%,!,# Variable name at the end of the automatic correction is not covered.
You can also modify the syntax should be treated as mathematical or logical operation to compare.

Old Basic BASIC
Old standard basic programs written in BASIC Full BASIC Please check if you want to modify the program. (Details)
Systems such as Microsoft BASIC N88BASIC check is required even if you want to modify programs written in automatically.
This item setting is not saved.
(Important) JIS Full BASIC if you run a program written in the
"OPTION BASE 0 to insert a" Please remove the check.

Angle
BASIC setting the standard unit is the size of the angle in radians,
OPTION ANGLE DEGREES
Angular size of the unit of time to write a (degrees) can be changed.
If you set this option, programs that use the trigonometric function
OPTION ANGLE DEGREES are inserted automatically.

If the state is pressed, the program insert the OPTION ANGLE DEGREES, at the corner units of the size of trigonometric functions.
(Important) already in the program have a statement OPTION ANGLE, this setting is ignored.
In addition, Microsoft in compatibility mode settings are ignored here.
This item setting is not saved.

Automatic indenting
Automatic indentation during translation (indented) Specifies whether.
However, the grammar is not compatible with Microsoft when the auto-indentation.
The setting for this field will be stored until the next start.

Keyboard Operation
Tab key to move between regions
Up and down arrow keys to select the item (as highlighted)
Check state is switched by pressing the spacebar
Press Enter to exit (ESC key to cancel)

- **Auto formatting on key input**

Options Menu

AutoFormat input

Function word to uppercase when you enter from the keyboard.
DATA statement and quotation marks, REM, and whether the sentence is being considered but, if the same spelling words using the word functions as the variable names and procedure names to uppercase.
Beginners ON, OFF recommend that advanced users to keep.
The setting for this field will be stored until the next start.

<Note>
The ability to transform words and BASIC.KW1 BASIC.KW2 are defined. More

- **Font**

Options Menu

Font
Specifies the font used for screen display. Printing font is specified in the Print menu.
<Note>
Windows2000/XP may not change the way you have specified. If so, once again please start from the end.

- **Function keys**

Options Menu

Function Keys

When you press the function key in the editor will insert the string you set here.
This setting will be saved until the next start.

- **Option settings**

Options Menu

Option Settings - Initialization
Restores the default settings in the options menu at the next reboot.
Also returns to the initial window position.

Text output window

- **File menu**

File menu (text)

Close
Close the window.
Save
Save that are displayed at the top.
Save as Name
Name and save.
Print
Print.
BASIC End
BASIC close.

- **Edit menu**

Edit Menu (Text)

Replace Search
Find and Replace window specifications differ editing program.
Although you can not specify the scope of the search, you can specify the search start position.

When looking at the definition of the word per word, Windows follow the specifications.

Write Protect
If the write protection is checked, keyboard input is disabled.
To change the check, please click on the mouse.

- **Option menu**

Options Menu (Text)

Margin
Specifies the default margin.
Please specify the number of 24 or more.

You can specify the maximum number depends on the number of points of Windows fonts and specifications. Please select the number that will fit on one line to run the following program display.

10 ASK MARGIN a
20 PRINT REPEAT $ ("W", a)
30 END
When you type anything other than the numerical value is not changed.

"Do not erased before execution"means that,
The next time you run the program does not erase the current contents.

- **Graphics Window**

1. File menu
2. File menu (s network graphics)

   Open
   You can import files that are stored in bitmap format. If you want to be yelling image reading operation of the program is in the options menu, please select Do not leave before running clear.
   When you load the image coordinate system, coordinate x 0 left, right x 1 coordinate, y 0 coordinate of the bottom, the top y-coordinate is reset to 1. In addition, SET WINDOW statement when the new left, right, bottom, coordinate system is introduced based on the top.

   Save
   24-bit bitmap and save it as a name that appears at the top of the window.

   Save Name
   The save format BMP, GIF, JPEG can choose from among.
3. Edit menu

Edit Menu (Graphics)

When you run the copy, you can pass the image to word processing or image processing software. In that case, the software receive images of the Edit menu "Paste"Please run.
You can also paste images from other software to the contrary.

5. Option Menu

5.1. Size

Options menu (s network graphics)
The size of the bitmap (Graphics)
You can also choose the size exceeds the screen resolution.
When not on the preferred size of the extended instruction program
Please run the SET BITMAP SIZE.

The setting will be saved until the next start.

5.2. Color index

Options menu (s network graphics)
Color index (Graphics)
SET LINE COLOR, SET POINT COLOR color index used when specifying colors in such statements, correspondence and determines the actual color.
If you want to assign a color other than the menu, the program please use SET COLOR MIX.
More colors corresponding to each number, the sample program
STATEMEN \ COLORS.BAS
Please check run.

5.3. Font

Options menu (s network graphics)

Text Font
plot text specifies the font to draw text characters.
Already drawn character font is not changed.

5.4. Erase before run

Options menu (s network graphics)

Erased before execution
Once selected, draw a picture superimposed on an existing one.

■ Debug

Debugging

Debugging (debug window)

Trace
Check to trace and record the value of variables and their values have changed, and ran lines.
If you want to run only a trace, then please select Continue.
When trace logging is not displayed, please choose a trace log from the View menu.
<Note> With a large number of variables, the trace is time-consuming. If so, please uncheck the trace.
<Note> If you run a program trace, trace the use of statements. By using a trace statement tracing for each program unit on / off can be set.

Breakpoint
In the Debug window, you can set a breakpoint.
Move the cursor to the line you want to set a breakpoint
Burekupointobotan please click with your mouse.
The line changes color when a breakpoint is underlined.
Now, if you choose to continue, then hit the break point.
Clear Breakpoint
When the cursor is on the line breakpoint is set,
Clicking Burekupointobotan,
Clear the breakpoint.
<Notes>
Breakpoints can be set only during execution.
If you want to set a breakpoint,
Select a step, please submit the debug window.
<Reference> If you set a breakpoint in the program, Break the statement uses.

History (Back)
History (Next)

Window resizing
The debug window can change the size.
When there are many variables and, when you want an array variable in detail, please resize.
How to Change Size
Move the mouse pointer near the border of the window, holding the left button while moving the mouse arrow changes to both the shape of the mouse pointer.

Content View
Of the contents to be displayed does not display characters beyond 1024.
(Windows95 is a measure for stable operation.)

Foundation

Lines

Lines

BASIC has a concept of lines.

A line consists of three parts, the line number, the statement, and the tail comment.
Example.
```
10 LET A=10   ! Substitutes 10 for the variable A
20 PRINT 2*A
30 END
```

Line Number
The numbers 10, 20, 30 written at the left end of lines are line numbers.
Line numbers may be not continual, but the line number of a line must be greater than that of a line above.
A line number must be written from the left end of a line, no extra spaces preceding it is allowed.
A line number is an unsigned integer.

Statements
An instruction statement is simply called a statement. The following of the above example are statements.
```
LET A=10
PRINT 2*A
END
```

Tail comments
The portion subsequent to an exclamation mark (!) is a tail comment.
Comments do not affect execution of a program.

Supplement
Omission of line numbers
On this BASIC, line numbers can be omitted.
Because line numbers become obstacles for rewriting, it is recommended that line numbers be omitted unless accordance with the standard is the purpose.
Supplementation or deletion of line numbers can be performed from the edit menu.

Line continuation

Line continuation

As the standard prescribes that the upper limit of the length of a line is 132, syntax for writing a statement across two or more lines is provided.
For example,
10 PRINT f(t)*COS(t), f(t)*SIN(t)
20 END
can be written as
10 PRINT f(t)*COS(t), &
& f(t)*SIN(t)
20 END

This is called a line continuation.
Any place where grammatically spaces can be put can be replaced by a line continuation.
A line continuation is the part from & to & in the next line.
& must be written at the beginning of a continued line. Any space character can not be put before.

As this BASIC allows a line of length much more than 132, line continuations are not necessary, however this syntax may be useful on the case of printing on a paper.

REM

REM statement

REM statements are used for writing comments or remarks.
Example.
10 REM This is a comment.
20 END

（Note）REM statements can be replaced by tail comments.

■ Indentifiers and Reserved words

Identifiers and Reserved Words □□

Identifiers
Identifiers are the name of variables, functions or subprograms.
An Identifier is an alphabet or an alphabet succeeded by alphabets, numerals or underscores.
A string identifier ends with $.
Identifiers of BASIC are case-insensitive. For example, 'A' and 'a' are identical.
Underscores can be included as angle_A.
(Note)
User-defined function names need not start with FN.
The maximum length of identifiers is 31 in the standard.

Reserved Words
The following words are prohibited to be used as identifiers by the standard.
PI,RND,MAXNUM,DATE,TIME,EXLINE,EXTYPE,CON,IDN,ZER,
TRANSFORM, NOT,ELSE,PRINT,REM, DATE$,NUL$,TIME$
(Note.)
Any word except the above can be used as a name of any variable or function.
For example, a user can define a function whose name is identical with a supplied function.

■ Main Program and External Procedures

Main program and External Procedures

A BASIC program consists of the main program and some or no external procedures.
The main program is the part from the beginning to the END line.
External procedures are written succeeding the main program.
An external procedure is an external subprogram, an external function definition, or an external picture definition.

Example
On the following program, lines from 100 to 130 compose the main program, and lines from 200 to 260 compose an external function definition.

```
100 DECLARE EXTERNAL FUNCTION fact
110 INPUT n
120 PRINT fact(n)
130 END
200 EXTERNAL FUNCTION fact(n)
210 IF n=1 THEN
220    LET fact=1
230 ELSE
240    LET fact=n*fact(n-1)
250 END IF
260 END FUNCTION
```

Program Unit
The main program or an external procedure is called a program unit. A program unit is the scope of an identifier or a OPTION statement.

Program Units

■ Program Characters

Program Characters

The characters that can be used for description of programs are provided by the standard on Full BASIC.
Any TAB or multi-byte space character can not be used as a substitute for a space.
Keywords and identifiers are case-insensitive.

Numbers

Numbers

Numbers

Numerical Constants
A Numerical constant without exponent part is written using numerals, a decimal point(.), and a plus sign or minus sign. The figure 0 preceding the decimal point can be omitted.
Example. '0.12' can be written as '.12'

（Note) Omitting 0's succeeding the decimal point, a numeric can be written as '23.', while Full BASIC does not distinguish '23.' from '23'.

Numerical constants with exponent parts
Example.
2.3E4，.23E5，and 23E3 stand for 23000.
2.3E-2，.23E-1，and 23E-3 stand for 0.023.
(Note) 'E' can be written in lower case. No space character can be written just before or just after 'E'.

Four arithmetic operations and power operation
Addition, subtraction, multiplication and division are denoted by +, -, *, / respectively.
ab, a raised to b, is denoted by a^b.
The priority is power operation first, multiplication and division second, and addition and subtraction third. Multiplication and division have the same priority. Addition and subtraction have the same priority. Operations with the same priority are applied from left. For example, 2^3^4 is evaluated as (2^3)^4.
Parentheses can be used for changing the priority. All parentheses are round parentheses even if they are nested.
0^0 is defined to be 1.

(Note) While minus signs are used for changing sign, those minus signs have the same priority with subtraction. Note that this is different from Microsoft BASIC.

(Note) Every non-integral power of a negative number causes an exception. For example, (-32)^0.2 causes an error, although it is -2 mathematically.

Numeric Expressions
The result of an arithmetic operation has precision greater than 16 digits.
If the true value of an operation can be represented by a decimal floating point of at most 16 digits, the result of the operation is accurate, that is, it has no error.
For example, the result of SQR(49) is accurately 7. The result of 7^2 is accurately 49.
(note)
The accuracy of the result of continued operations can not be guaranteed.
For example, it is uncertain whether the result of SQR(7)^2 becomes 7.

Refer to Option menu Numbers and Precision.

LET

Assignment

LET statement
LET Numeric_Variable = Numeric_Expression
Examples.
LET A=2+3
calculates the expression on the right hand and assigns the value to the variable on the left hand.
LET A=B
assigns the value of B to A.
LET A=A+1
add 1 to A.

(Note)
'LET' cannot be omitted, because a LET statement is not an equation.

Numeric Variables
A numeric variable has the precision of 15 digits.
A result of an arithmetic operation, which has precision of over 16 digits, is rounded to 15 digits when it is assigned to a variable.
Example. The result of the following program never becomes 0.
10 LET A=1/3
20 PRINT 1/3-A
30 END

OPTION ANGLE

OPTION ANGLE

OPTION ANGLE RADIANS
lets the supplied trigonometric functions use radian measure.

OPTION ANGLE DEGREES
lets the supplied trigonometric functions use degree measure.

If none of the above is written, the unit of angles is assumed to be radians.
The scope of an OPTION statement is the program unit that contains it.
Thus OPTION ANGLE statement must be written on every program unit which uses trigonometric functions.
An OPTION statement is a declarative statement, which is not an object of the control.

■ OPTION ARITHMETIC

OPTION ARITHMETIC

OPTION ARITHMETIC DECIMAL
lets all numbers be decimal floating point.

OPTION ARITHMETIC NATIVE
lets all numbers be double precision floating point.
The precision is about 16 digits.
Although decimal fractions may have errors, computation is fast.

The scope of OPTION ARITHMETIC statement is the Program unit where it is written.
Program units that do not deliver or receive numbers may have different OPTION ARITHMETIC.
An OPTION statement is a declarative statement, which is not an object of the control.

Original Enhancements

OPTION ARITHMETIC DECIMAL_HIGH
Decimal 1000 digits operation. (Transcendental functions have the precision of 17 digits(last digit even).)

OPTION ARITHMETIC COMPLEX
Complex numbers

OPTION ARITHMETIC RATIONAL
Rational numbers of huge digits. (irrational functions have the precision of 17 digits(last digit even).)

(Note)
If a program unit has no OPTION ARITHMETIC statement, number manipulation obeys the designate on the Option Menu - Precision

DECLARE NUMERIC

DECLARE NUMERIC

DECLARE NUMERIC
declares numeric variables.
Example.
DECLARE NUMERIC A,B,M(10)
declares simple variables A and B, and a 1-dimensional array M.

The way how to declare arrays is same as DIM statements.
Example.
1-dimensional arrays A(10), B(2 TO 7)
2-dimensional arrays C(10,10), D(0 TO 2, 0 TO 5)
3-dimensional arrays E(4,4,2), F(3,3,2 TO 3)
Note that bounds of indices must be constants in the declaration.

DECLARE statements are declarative statements, which are not objects of the control.

When Option menu- Syntax - "Compel All Variables Declared" is checked, if a non-declared variable is found, compiling stops. This is useful to avid a bug caused by a typo.

Refer to DECLARE STRING

■ Supplied Functions

■ Numeric Functions (general)

Supplied Functions (numeric general)

ABS(x)	The absolute value of x
SQR(x)	The nonnegative square root of x
INT(x)	The largest integer not greater than x
	Example. INT(5.4)=5, INT(5)=5, INT(-4.4)=-5
MOD(x,y)	x-y*INT(x/y) x modulo y
	Example. MOD(5,3)=2, MOD(-5,3)=1, MOD(3.14,1)=0.14
ROUND(x,n)	INT(x*10^n+0.5)/10^n The value of x rounded to n digits of decimal place
CEIL(x)	-INT(-x) The smallest integer not less than x
	Example. CEIL(5.4)=6, CEIL(5)=5, CEIL(-4.4)=-4
SGN(x)	the sign of x
	SGN(x)=1 if x>0,
	SGN(0)=0,
	SGN(x)=-1 if x<0
IP(x)	SGN(x)*INT(ABS(x)) The integer part of x
	Example. IP(5.4)=5, IP(5)=5, IP(-4.4)=-4
FP(x)	x-IP(x) The fractional part of x
REMAINDER(x,y)	x-y*IP(x/y)
	Example. REMAINDER(5,3)=2, REMAINDER(-5,3)=-2
TRUNCATE(x,n)	IP(x*10^n)/10^n

<Note> SQR-functions can not be used on the RATIONAL operating mode, though INTSQR-functions can be used.

■ Exponential and Logarithmic functions

Supplied Functions (Exponential and Logarithmic) □□□

EXP(x) The exponential of x, ex, where e=2.71828...
LOG(x) The natural logarithm of x
LOG2(x) The logarithm of x base 2
LOG10(x)The common logarithm of x

<Note> These functions can not be used on the 1000-digit operation mode nor on the RATIONAL operation mode.
Refer to Option-Menu Numbers

- **Trigonometric Functions**

Supplied Functions (Trigonometric)

SIN(x)	The sine of x
COS(x)	The cosine of x
TAN(x)	The tangent of x
CSC(x)	The cosecant of x , 1/sin(x)
SEC(x)	The secant of x , 1/cos(x)
COT(x)	The cotangent of x , 1/tan(x)
ASIN(x)	The arcsine of x, the angle θ such that $\sin\theta$=x, where $-\pi/2 \leqq \theta \leqq \pi/2$A
COS(x)	The arccosine of x , the angle θ such that $\cos\theta$ = x, where $0 \leqq \theta \leqq \pi$
ATN(x)	The arctangent of x , the angle θ such that $\tan\theta$ = x, where $-\pi/2 < \theta < \pi/2$
ANGLE(x,y)	The angle between the positive x-axis and the vector joining the origin to (x,y). where $-\pi <$ ANGLE(x,y)$\leqq\pi$

<Note> If OPTION ANGLE DEGREES is in effect, the angles about these functions are in degrees.

<Note> These functions can not be used in the 1000-digit operation mode nor the RATIONAL operation mode. See Precision options menu

<Supplement>
PI The ratio of the circumference of a circle to its diameter π=3.141592....

- **Hyperbolic functions**

Supplied functions (Hyperbolic)

SINH(x)	The hyperbolic sine of x
COSH(x)	The hyperbolic cosine of x
TANH(x)	The hyperbolic tangent of x

<Note> These functions can not be used on the 1000-digit operation mode nor the RATIONAL operation mode. See Precision options menu

■ Randon Numbers

Random Numbers□□

RND-function
RND The pseudorandom number, where 0≦RND < 1
< Note > RND is a reserved word of Full BASIC.

randomize statement
RANDOMIZE
Sets the starting point of the pseudorandom numbers unpredictable.

[Note]
The sequence of the pseudorandom numbers is global over the whole program.

Example. A program which generates 20 random integers within 1 to 6.

```
10 RANDOMIZE
20 FOR t=1 TO 20
30   LET n=1+INT(6*RND)
40   PRINT n
50 NEXT t
60 END
```

If RANDOMIZE in line 10 lacks, this program generates the same sequence for every time.

[Supplement]
The pseudorandom numbers generated by RND-functions are those generated by Mersenne Twister method.
They have precision of 52 bits on the binary or the complex mode, 50-bits on the decimal, 1000-digits decimal, or the RATIONAL operation mode.
The algorithm depends on mt19937ar.c by Matsumoto and Nishimura, 2002

■ MAXNUM and EPS

MAXNUM, EPS

MAXNUM function
MAXNUM
The largest manipulable number.
MAXNUM is 1E99 on the decimal operation mode.
MAXNUM is about 1.8E308 on the binary operation mode.
(Note)
MAXNUM is undefined on the rational operation mode. (Compiler stops if any.)
(Note)
MAXNUM is a reserved word.

EPS functions
EPS(x)
EPS(x) is the larger of the differences of x and the numbers adjacent to x, provided that if it is smaller than EPS(0), EPS(x) is EPS(0).
EPS(x) means the resolution around x.
Decimal BASIC defines EPS(x) with respect to the precision of numeric variables, while the definition on the standard is vague.
As the precision of variables is 15 digits, EPS(x)=1E-14 for a number x not less than 1 and less than 10.
EPS(0) is the smallest positive number. EPS(0) is 1E-99 on the decimal mode of Decimal BASIC. If a number of absolute less than EPS(0) is assigned to a variable, the value of the variable becomes 0.
The EPS function can be expressed using a logarithmic function as follows.
EPS(x)=MAX(10^(INT(LOG10(ABS(x))-14)),1E-99) for x≠0

(Note) If arithmetic option is NATIVE,
EPS(0)=2^(-1074)□4.94E-324
EPS(x)= MAX(2^(INT(LOG2(ABS(x))-52)),EPS(0)) for x≠0

The EPS function is undefined on the rational operation mode.

■ Numeric Constrants and Others

Supplied Functions（Constant or else）

Constant
PI
An approximate value of the ratio of the circumference of a circle to its diameter π=3.141592....
<Note> PI is a reserved word.

Conversion of the unit of angles
DEG(x)　x*180/π
RAD(x)　x*π/180

Others
MAX(a,b)　The larger of a and b
MIN(a,b)　The smaller of a and b

Date and Time

Supplied Functions (Date or Time)

TIME The time elapsed since the previous midnight, expressed in seconds.

DATE The current date in decimal form YYDDD, where YY are the last two digits of the year and DDD is the ordinal number of the current day of the year.
Example. The value of DATE on Jan. 1, 2011 is 11001.

DATE$ The date in the string representation "yyyymmdd"
TIME$ The time of the day in the form "hh:mm:ss"<Note> These function names are reserved words.

<Usage.>
```
10 LET t0=time
    . . . . . .
    . . . . . .
80 PRINT time-t0;" seconds elapsed."
90 END
```

[Notice] If the date changes during the execution, the result shall not be correct.

<Supplement>Functions that return the year, month or day
```
DEF year = INT(VAL(date$)/10000)
DEF month = MOD(INT(VAL(date$)/100),100)
DEF day = MOD(VAL(date$),100)
```

- **Original Enhancement**

Supplied Functions (Original Enhancement)

FACT(x) The factorial of x
PERM(n,r) The number of permutations
COMB(n,r) The number of combinations (the binary coefficient)ROUND(x) The value of x rounded to an integer.
ROUND(x) coincides with ROUND(x,0) on The DECIMAL or the 1000-digit or the RATIONAL operation mode.
ROUND(x) is the value of x rounded half even on the BINARY or the COMPLEX operation mode.

Strings

Strings

Strings

String Constants
String Constants are expressed being enclosed in double quotation marks " and ".
Example
PRINT "BASIC"

If a double quotation mark is included in a string constant, it must be repeated twice.
Example.
10 PRINT "A""BC"""
20 END
Output
A"B"

String identifiers
String variables and string functions are named by identifiers that end with $.
Example.
10 LET A$="ABC"
20 PRINT A$
30 END

String concatenation
The string concatenation operator is &.
Example.
10 LET a$="123"
20 LET b$="456"
30 PRINT a$ & b$
40 END
Output
123456

String LET-statement
Example.
10 LET s$="123"
20 LET s$=s$ & "4"
30 PRINT s$
40 END
Output
1234

■ Substring

Substring

1. A String variable can have a substring qualifier.
When s$ is a string variable, and m,n are numeric expressions, s$(m:n) stands for the substring of s$ from m-th character to n-th character.
For example, if a$="12345", a$(2:4) is "234".

2. A string variable with a substring qualifier can be written on the left hand side of a LET statement.
Example.
LET s$(m:n)=a$
The substring of s$ from m-th character to n-th character is replaced with a$.
The length of a$ can be different from the length of the substring to be replaced.
Example.
10 LET A$="1234567"
20 LET A$(2:6)="ABC"
30 PRINT A$
40 END
Output
1ABC7

Insertion of a string to a string variable
LET A$(n+1:n)=b$
inserts b$ at the n-th character of A$.
Example.
10 LET A$="1234567"
20 LET LET A$(4:3)="ABC"
30 PRINT A$
40 END
Output</P
123ABC4567

Deletion of a substring from a string variable
LET A$(m:n)=""
deletes m-th character to n-th character from A$.
Example.
10 LET A$="1234567"

```
20 LET LET A$(4:6)=""
30 PRINT A$
40 END
Output</P
 1237
```

(Note)
String variables with substring qualifiers can be written in a INPUT statement or a READ statement.

A substring qualifier can not be applied to a string expression.
A function that picks out a substring from a string can be defined as follows.
DEF SUBSTR$(a$,m,n)=a$(m:n)
SUBSTR$ is pre-defined on Decimal BASIC, so the DEF statement above need not be written.

■ String Functions

String Functions

Let a$, b$ be string expressions, a, m be numeric expressions below.

Functions of which the result is a number

LEN(a$)	The length of s$, the number of characters a$ contains.
POS(a$,b$)	The position of first occurrence of b$ within a$.
	If no occurrence, the value is 0.
POS(a$,b$,m)	The position of first occurrence of b$ within the m-th or latter characters of a$.
VAL(a$)	The value of a numeric a$ represents.
	Leading and trailing spaces are ignored.
	If a$ does not represent a number correctly, VAL(a$) causes an exception.
ORD(a$)	If a$ consists of a single character, the value is the character code of it.
	And ORD("CR")=13, ORD("LF")=10, ORD("HT")=9, etc.

Functions of which the result is a string

REPEAT$(a$,m)	The string consisting of m copies of a$
STR$(a)	The string a is displayed as.
	No leading or trailing spaces are included.
USING$(a$,x)	The string representing x using a$ as a format item.
	Format is common with PRINT USING.

a$ can not hold extra characters except a format.

CHR$(m)	The character whose character code is m.
LCASE$(a$)	The string on which upper case letters are converted to the lower case.
UCASE$(a$)	The string on which lower case letters are converted to the upper case.
LTRIM$(a$)	The string from which all leading spaces are removed.
RTRIM$(a$)	The string from which all trailing spaces are removed.
DATE$	The date of the form "yyyymmdd"
TIME$	The time of the form "hh:mm:ss"

(Supplementary)
On the complex mode or the rational mode, if a number is converted to a string using STR$, and converted back to a numeric using VAL, it may be changed.

■ Bit Patter manipulation

Bit Pattern

Bit Pattern Functions

BVAL(a$,2)	The non-negative integer whose binary representation is a$.
BVAL(a$,16)	The non-negative integer whose hexadecimal representation is a$.
BSTR$(n,2)	Binary representation of a non-negative number n.
BSTR$(n,16)	hexadecimal representation of a non-negative number n. The argument of BSTR$ is rounded to an integer. A negative argument of BSTR$ causes an exception of extype 4203.

(Note) BVAL and BSTR$ do not have satisfactory precision on the 1000-digit decimal mode and on the rational mode.

■ DECLARE STRING

DECLARE STRING

DECLARE STRING
Declares string simple variables or string arrays.
Example.
DECLARE STRING A$,B$,S$(10)

(Note) String identifiers must end with $.

Refer to DECLARE NUMERIC

■ Original enhancement

Original Enhancement

Let a$ be a string expression , m, n be numeric expressions below.

Functions whose result is a string
SUBSTR$(a$,m,n) Substring of a$ from m-th character to n-th character.
MID$(a$,m,n) n characters of a$ from m-th character.
LEFT$(a$,n) The first n characters of a$
RIGHT$(a$,n) The last n characters of a$
The above functions are defined as follows.
DEF SUBSTR$(a$,m,n)=a$(m:n)
DEF MID$(a$,m,n)=a$(m:m+n-1)
DEF LEFT$(a$,n)=a$(1:n)
DEF RIGHT$(a$,n)=a$(LEN(a$)-n+1:LEN(a$))

Functions whose result is numeric.
BLEN(a$) The length of a$ in bytes.
This function is prepared for handling multi-byte characters.

■ Multi-byte Characters

Chinese characters

BASIC Full has been standardized using the assumption that the character code number from 0 to 127. Sono Tame, JIS has not only written about things that may be treated after the second 128 characters.

This BASIC, Microsoft code inside the Kanji Kanji (Shift-JIS) expresses, in the Kanji code using the Microsoft file input and output. String comparison is done based on the internal representation in bytes.
Also, PRINT statements, and position control of the PRINT USING digit display is used at length in bytes of the internal representation format control statement.

POS location and character of the substring function, CHR $ function ORD function, LEN function, CHARACTER INPUT statement in the handling of Chinese characters in the book there is in BASIC mode 2. One mode is treated as a single kanji character, and one mode of dealing with a string of bytes as a unit.

The two modes will be determined during translation OPTION CHARACTER statement. Specified to be treated as a single Chinese character OPTION statement
OPTION CHARACTER KANJI
And the processing is performed in a string of bytes OPTION statement
OPTION CHARACTER BYTE
Be. OPTION CHARACTER OPTION range is similar to other statements, the program unit. However, if you write any of these are treated as a mode kanji characters.

Treated as kanji character mode, CHR $ function ORD function based on the JIS Kanji code to handle.

Input and Output

■ PRINT

PRINT

PRINT
Displays the values of numerical expressions or string expressions.
Two or more items can be written punctuated by commas or semicolons.
A comma makes the place proceed to the next zone.
A semicolon does not so.
Example
PRINT "2+3="; 2+3, "2-3="; 2-3
Output
2+3= 5 2-3=-1

If TAB(n) is written as an item, The place is made to proceed to the n-th column.
Usually it is punctuated by a semicolon.
Example
PRINT 2; TAB(10); 2^10
Output
2 1024

■ PRINT USING

PRINT USING

PRINT USING

PRINT USING format_string : numeric_expression
Displays the value of the expression using the format.

(1) The form without a exponent part
Example.
PRINT USING "----%.####": a
displays the value of a with the integer places including the sign 5 digits, fractional places 4 digits.
" 12.5000" for a= 12.5
" -12.5000" for a=-12.5
" 0.0000" for a= 0
Usage
(i) Integer part.
Put a queue of an arbitrary number of either of +'s or -'s,
without a break put a queue of an arbitrary number of either of %'s or #'s.
Meanings
- Replaced by numerals, a minus sign, or spaces.
+ Replaced by numerals, a sign, or spaces.
% Replaced by numerals.
# Replaced by numerals or spaces.
(ii) Fractional part
Put a queue of an arbitrary number of #'s.
(Note) A format for a negative number must have +'s or -'s.
(Note) If Option menu - compatibility - behavior - "No preceding space generated" is checked, # can be replaced by a minus sign, that is, a format consisting of only #'s can be used for a negative number.

(2) The form with a exponent part
Example.
PRINT USING "-%.#####^^^^":a
displays the value of a with the sign part 1 digit, the integer part 1 digit, the fractional part 5 digits, and exponent part 4 digits.
" 1.25000E+01" for a= 12.5
"-1.25000E+01" for a=-12.5
" 0.00000E+00" for a= 0

Usage
(i) Integer part and fractional part
Same as the form without an exponent part.
(ii) Exponent part
At least 3 ^'s shows the exponent part (including "E").

Notes
1. A format string is a string expression, which can contain string variables or string functions.
Example
PRINT USING "." & REPEAT$("#",15) : 1/7

2. A format string can contain two or more formats.
Example.
PRINT USING "### #.############": n, 1/n
On this case, the expressions must be punctuated by commas (not semicolons).

3. A format item can contain commas.
Example
PRINT USING "###,###,###,###,###": 2^32
Output 4,294,967,296
Note that commas are part of a format item.

4. When a format string contains non-format characters, they act as punctuations of format items and are outputted just like they are.
Example
PRINT USING "A= ### B= ###": A, B
(Note) The format characters are the following 11 characters.
$ % * + , - . < > ^

5. Formatting can be used for strings.
Example
PRINT USING "<#####": s$
Usage
(1) The first character of a format can be a "<" or a ">"
"<" left adjusts a string. ">" right adjusts a string.
Lack of "<" or ">" center adjusts.
(2) "<" or ">" is part of digits.

6. The format string can be assigned by a IMAGE line.

Example
50 IMAGE: ##### #.#########
60 PRINT USING 50: N, 1/N

Refer to USING$ function Non-fatal exceptions

(Note) The above is not the whole. For more information, see the ANSI Standard Full BASIC, JIS Full BASIC or ECMA BASIC standard.

■ Zone Width and Margin

MARGIN, ZONEWIDTH

SET MARGIN n
determines the width of the output, where n is a numeric expression.
When the printed output exceeds the margin if it continues, a new line shall be inserted.
Example. Output does not exceed 80 column.
```
SET MARGIN 80
FOR i=1 TO 1000
  PRINT i;
NEXT i
```

SET ZONEWIDTH n
determines the zone width, where n is a numeric expression.
The zone width is the width which an item occupies.
Example.
```
SET ZONEWIDTH 8
FOR i=1 TO 10
  PRINT i,
NEXT i
```

(Note.) The unit of the width is bytes. Multi-byte characters occupies two or more columns.

■ INPUT

INPUT

INPUT variable_1, variable_2, …, variable_n
assigns data entered from the keyboard to variables.

When an INPUT statement has two or more variables, the values are written punctuated by commas in the input reply.
Example.
On execution of
INPUT a, b
enter the data as
  2 , 3

An input reply for an INPUT statement can consist of two or more lines.
In such a case, any line except the last must be written ending with a comma.
However no item can be written divided into more than one line.
Example. For
INPUT a, b
an input reply can be as
2,
3

If an item is assigned to a string variable, the item should be enclosed in the double quotes, otherwise leading or trailing spaces shall be eliminated.
Example.
If
" Sure ?"
is entered for
INPUT s$
s$ shall be Sure ?.

Note.
When an input line whose last character except spaces is a comma, it has a continuation line.
Therefore, the null string cannot be input without quoted.

References.
If you want to input characters without quoted, use LINE INPUT.
Refer to Non-fatal exceptions

INPUT PROMPT

INPUT PROMPT

INPUT PROMPT String_Expression : varable_1, variable_2 , …
displays the string and then wait for the user to enter data.
Example 1.
INPUT PROMPT "a,b=":a,b
Example 2.
LET Prompt$="a,b="
INPUT PROMPT prompt$:a,b

INPUT TIMEOUT

INPUT TIMEOUT

Let t be a numeric expression.
INPUT TIMEOUT t : variable, variable , …
INPUT TIMEOUT t ,PROMPT Format_String : variable , variable , …
If no input reply is given within t seconds, an exception of extype 8401 is raised.

Example
```
100 LET prompt$="If no reply is given within 3 seconds, it is assumed that 1 is entered."
110 WHEN EXCEPTION IN
120   INPUT TIMEOUT 3 ,PROMPT prompt$:n
130 USE
140   SELECT CASE EXTYPE
150   CASE 8401
160     LET n=1
170   CASE 8103
180     LET prompt$="Re-input"
190     RETRY
200   CASE ELSE
210     EXIT HANDLER
220   END SELECT
230 END WHEN
240 PRINT n
250 END
```

INPUT ELAPSED

INPUT ELAPSED

Let s be a numeric variable.
INPUT ELAPSED s: variable , variable , …
INPUT ELAPSED s ,TIMEOUT t : variable , variable , …
The time in seconds that has elapsed until an input reply is given is assigned to s.

LINE INPUT

LINE INPUT

LINE INPUT s$
assigns the whole input line to a string variable s$.

(Supplement)
A LINE INPUT statement allows a PROMPT clause, a TIMEOUT clause, or a ELAPSED clause to be included.
Example.
LINE INPUT PROMPT "Enter favorite words.":s$

CHARACTER INPUT

CHARACTER INPUT

CHARACTER INPUT s$
waits for a key input and then assigns the character to a string variable s$
Enter key, BackSpace key, or ESC key stands for a character.
Enter stands for a character whose ordinal is 13,
Ctrl-Enter stands for a character whose ordinal is 10,
BackSpace stands for a character whose ordinal is 8,
ESC stands for a character whose ordinal is 27.
Any other key that does not stand for any character such as a function key or an array key is ignored.

Supplement.
A PROMPT clause, a TIMEOUT clause or an ELAPSED clause can be included in a CHARACTER INPUT statement as in a INPUT statement.

Example.
CHARACTER INPUT PROMPT "Press the number that you selected.":s$

In addition, a CLEAR clause or a NOWAIT clause can be written in a CHARACTER INPUT statement.
Example.
CHARACTER INPUT CLEAR:s$
All characters that lie in the typeahead buffer are removed before key input.
Example.
CHARACTER INPUT NOWAIT:s$
do not wait for a key input. s$ remains the former value if the typeahead buffer is empty.

When plural clauses are included, they must be separated by a comma or commas. The order is not definite.
Examples.
CHARACTER INPUT ELAPSED n ,CLEAR: s$
CHARACTER INPUT CLEAR, ELAPSED n: s$

The number of characters in the typeahead buffer can be known via the following statement, where n is a numeric variable.
ASK CHARACTER PENDING n

See also SET ECHO

SET ECHO

SET ECHO

SET ECHO "ON"
INPUT statements shall echo the input replies.
SET ECHO "OFF"
INPUT statements shall not echo the input replies.
"ON" or "OFF" can be indicated by a string expresion.
When a program starts, the echo is set to "ON".

READ and DATA

READ, DATA

READ
assigns the data written in the DATA statements to the variables.

DATA
DATA Statements describes the data that shall be read by the READ statements.
The data items are written in DATA statements punctuated by commas.
The data may extent over more than one DATA statement.
In that case, unlike a input reply, no comma is put on the tail.

Example 1.
```
10 DATA 34,78,13,54
20 READ a,b,c,d
```

Example 2.
```
10 DATA 34,78
15 DATA 13,54
20 READ a,b,c,d
```

Example 3.
```
10 DATA 34,78,13,54
20 READ a,b
25 READ c,d
```

Data items that is to be assigned to string variables are written enclosed in double quotation marks.
Example.
```
10 DATA "Tanaka",168
20 READ N$,L
```

On some conditions, data items can be written unquoted. For detail, see the Full BASIC standard.
Example
```
10 DATA "I am a boy." , ABC
20 READ A$,B$
30 PRINT A$,B$
40 END
```

All the items written in the DATA statements of a program unit forms a single sequence of data.
Data items are read in written order in the program unit.

RESTORE
let the READ statement that shall be executed next read from the first item.

RESTORE line_number
The READ statement that shall be executed next shall read from the item located in the DATA statement that is labeled line_number .

Note.
The scope of DATA statements is the program unit.
Data is independent for each program unit.

■ READ IF MISSING

READ IF MISSING

READ IF MISSING THEN EXIT DO: variables
READ IF MISSING THEN EXIT FOR: variables
READ IF MISSING THEN line_number: variables

A READ statement can contain the instruction that is to be executed on the exhaustion of data.
The instruction is one of EXIT DO, EXIT FOR, or a line number.

Example.
```
10 DATA 1,2,3
20 DO
30    READ IF MISSING THEN EXIT DO: A
40    PRINT A,A^2
50 LOOP
60 END
```

Original Enhancement (keyboard)

Proprietary extensions (keyboard)

Functions
GetKeyState (n)
The Win32API GetKeyState
Check the status of key
Argument is the virtual key codes
Key is pressed when the result is negative, non-negative when not pressed.
[Reference]
The main key of virtual key codes
Enter 13
Shift 16
Ctrl 17
Alt 18
Blank 32
37 ←
38 ↑
39 →
40 ↓
0-48 9-57
A ~ Z 65 ~ 90
Num 0 96 9-105
Key code numbers and letters corresponding letter key (upper case) equal to the value of ORD function.

Investigate cases of virtual key codes

```
DO
  FOR i = 8 TO 239
    IF GetKeyState (i) <0 THEN PRINT i
  NEXT i
LOOP
END
```

Warning
To use this command with caution. When I came to the foreground BASIC-party applications, is a key input to the application.
This instruction is not intended to keystrokes only just peek at the key state.

Arrays

Arrays

Arrays

OPTION BASE 0
designates 0 as the default lower bound.
OPTION BASE 1
designates 1 as the default lower bound.

If a program unit contains no OPTION BASE statement, the default lower bound is assumed to be 1.
A program unit can contain at most one OPTION BASE statement.
If a OPTION BASE is written in a program unit, it must precede the first DIM statement.

A DIM statement declares arrays with the names, dimensions, and the lower and upper bounds of indices.
If a declaration does not contain the lower bound, it is assumed to be the option base.
Examples.
DIM A(2 TO 10)
Declares a 1-dimensional array A with the lower bound 2, the upper bound 10.
DIM A(10)
Declares a 1-dimensional array A with the upper bound 10.
DIM A(4,5)
Declares a 2-dimensional array A with the upper bound of the first dimension 4, the upper bound of the second dimension 5.
DIM A(0 TO 10, 2 TO 11)
Declares a 2-dimentional array A with the range of the first subscript 0 to 10, and the range of the second subscript 2 to 11.

Dimensions of arrays are at most 3.
OPTION statements and DIM statements are declarative statements, which are effective only written, and not objects of the control.

If a fraction is specified for the subscript, it shall be rounded to an integer.
Example.

```
 10 DIM a(1 TO 10)
 20 LET k=1.2
 30 LET a(k)=2*k
```

makes a(1) be 2.4.

(Note) Differences from Minimal BASIC.
(1) The default option base is 1.
(2) No array can be written without declaration.
(Note) Differences from some dialect.
(1) A simple variable and an array cannot have the same name.
(2) Only one declaration for an array can be written,
that is, an array does not vary in dimension.

Refer to Enhanced syntax for DIM statement

■ Array functions

Array functions

Let M be an array, i be a numeric expression below.
LBOUND(M,i) The lower bound of the i-th dimension.
UBOUND(M,i) The upper bound of the i-th dimension.
SIZE(M,i) The number of the i-th subscripts.
SIZE(M) The number of current elements.
MAXSIZE(M) The upper limit of the number of elements.
Example 1.
DIM A(8,9)
yields
lbound(A,1)=1, lbound(a,2)=1, ubound(a,1)=8, ubound(a,2)=9
size(A,1)=8, size(A,2)=9, size(A)=72, maxsize(A)=72
if no OPTION BASE exists.

Example 2.
10 DIM a(10)
20 MAT a=ZER(5)
yields SIZE(a)=5 , MAXSIZE(a)=10.
if no OPTION BASE exists.

If M is 1-dimensional, the following abbreviations are allowed.
LBOUND(M) The lower bound of the subscript of M.UBOUND(M) The upper bound of the subscript of M.
DET-function
Let M be a 2-dimensional array.
DET(A) The determinant of A
If A is not square, an exception shall be caused on execution.

DOT-function
Let A,B be 1-dimensional arrays
DOT(A,B) The dot (inner) product of A and B.
If the size of A and B are different, an exception shall be caused on execution.

MAT statements

MAT statements

Full BASIC provides MAT statements, which perform matrix operations.

Let A, B and C be arrays, m, n and p be numeric expressions below.

MAT A = B
substitutes B for A, where A and B have the same dimension.

MAT A = ZER
substitutes 0 for all elements of A.

MAT A = ZER(m) when A is 1-dimensional
MAT A = ZER(m, n) when A is 2-dimensional
MAT A = ZER(m, n, p) when A is 3-dimensional
changes the upper bounds of A and substitutes 0 for all elements.
Note that the dimension of an array do not change.

MAT A = CON
substitutes 1 for all elements of A.

MAT A = CON(m)
MAT A = CON(m, n)
MAT A = CON(m, n, p)
changes the upper bounds of A and substitutes 1 for all elements.

MAT A=B+C
MAT A=B-C
MAT A=B*C
Substitutes the sum, difference or product of B and C for A.
In case of the product, dimensions of A, B and C must be one of 1-2-1, 1-1-2, 2-2-2.

MAT A=(numeric_expression)*B
Scalar multiplication.
If numeric_expression is a constant, a function reference or a variable, parentheses can be omitted.
Example. Let x be a numeric variable.

MAT A=x*A
MAT A=(2*x)*B
MAT A=ABS(x+3)*B

MAT A=(numeric_expression)*CON
The value of numeric_expression is assigned to all elements of A.

Let A and B be 2-dimensional arrays.

MAT A = IDN
substitutes the identity matrix for A.
If A is not square, an exception of 6004 shall be raised.

MAT A = IDN(n)
substitutes n×n identity matrix for A.

MAT A=INV(B)
substitutes the inverse matrix of B for A.

MAT A=TRN(B)
substitutes the transposed matrix of B for A.

Cross Product (Original Enhancement)
Let A, B and C be 1-dimensional arrays.
MAT A=CROSS(B,C)
Substitutes the outer product of B and C for A.
If the size of B and C are not 3, an exception of extype 6001 shall be raised.

Notes.

Although a MAT statement may change the size of an array, the lower bounds of an array does not change, and only the upper bounds are adjusted for the new size.
For example, when an array such as A(1)=1，A(2)=2，A(3)=3 is substituted for an array B with the lower bound 0, B becomes as B(0)=1，B(1)=2，B(2)=3.
That is, MAT B=A means
LET B(0)=A(1)
LET B(1)=A(2)
LET B(2)=A(3)

Note.
MAT A=ZER(m TO n) does not make the lower bound of A into m.
This statement keeps the lower bound of A and changes only the size of A to n-m+1.
If you want to change the lower bound of an array, use a MAT READ statement or an original enhancement MAT REDIM.

Note.
If a MAT statement yields that the number of elements of an array exceeds the number of elements first declared, an exception of extye 5001 shall be raised.

Note.
Addition, subtraction, or multiplication of only two matrices can be performed.
Although MAT A=B*C*D can be compiled when A, B, C and D are all 2-dimensional, this MAT statement is a transform-assignment and an exception shall be raised unless the size of every matrix is 4× 4. Moreover, even if the size of every matrix is 4×4, the precision of result shall be that of the graphics subsystem (currently binary double precision).

■ String MAT statements

String MAT Statements

Let A$ and B$ be string arrays.

MAT A$=nul$
assigns the empty string to all elements of A$

Let s$ be a string expression.
MAT A$=(s$) & nul$
assigns s$ to all elements of A$
When s$ is a string variable, a string constant, or a string function reference, the parentheses can be omitted.

MAT A$(m:n)=(s$) & nul$
replace the m-th character to the n-th character of A$ with s$.

Le A$ be 1-dimensional and the lower bound of it be equal to option base.
MAT A$=nul$(n)
changes the upper bound of A$ to n, and assigns the empty string to all elements.

Le A$ be 2-dimensional and the lower bounds of it be equal to option base.
MAT A$=nul$(m,n)
changes the upper bounds of A$ to m and n, and assigns the empty string to all elements.

Let A$ and B$ have the same dimension and the same lower bounds.
MAT A$=B$
substitutes B$ for A$.

Let A$ and B$ have the same size.
MAT A$(m:n)=B$
Replace the m-th character to n-th character of A$ with B$.

Let B$ and C$ have the same size and the same dimension as A$.
MAT A$=B$ & C$
Substitutes an array that is obtained by concatenating each element of B$ and each element of C$ for A$.

Let A$ and B$ have the same dimension, s$ be a string expression.
MAT A$=(s$) & B$
Substitutes an array that is obtained by concatenating s$ and each element of B$ for A$.
MAT A$=B$ & (s$)
Substitutes an array that is obtained by concatenating each element of B$ and s$ for A$.

■ MAT READ

MAT READ

MAT READ array, array, …, array
assigns data to arrays in order of latter index changing fast.
Example.
10 DATA 1,2,3,4
20 DIM A(2,2)
30 MAT READ A
yields A(1,1)=1, A(1,2)=2, A(2,1)=3, A(2,2)=4.

Let A be an array and m and n be a numeric expression.
MAT READ A(n)
changes the lower bound to option base and the upper bound to n and then assigns data.

MAT READ A(m TO n)
changes the lower bound to m and the upper bound to n and then assigns data.

Example
10 DATA 1,2,3,4,5,6
20 DIM a(10)
30 LET m=-2
40 LET n=3
50 MAT READ a(m TO n)
yields a(-2)=1, a(-1)=2, …, a(3)=6.

2 or 3-dimensional arrays have similar MAT statements.

Note.
Handling of the lower bounds of indices is different from that of MAT statements.

■ MAT INPUT

MAT INPUT

MAT INPUT Array
assigns the data entered from the keyboard to the array.
The elements are written separated by commas in the input line.
When an input line ends with a comma, the input reply is regarded to continue to the next line.

MAT INPUT array(n)
changes the upper bound to n, and then inputs, where n is a numeric expression.

MAT INPUT array(m TO n)
changes the lower bound and the upper bound, and then inputs, where m and n is numeric expressions.
Similar type of MAT INPUT statements are available for 2 or 3-dimensional arrays.

Let A be a 1-dimensional array.
MAT INPUT A(?)
changes the upper bound according to the number of entered numeric values.
The lower bound do not change.
This type of a MAT INPUT statement can apply only to a 1-dimensional array.

MAT LINE INPUT string_array
inputs the lines as many as the elements of the array.

■ MAT PRINT

MAT PRINT

A MAT PRINT statement outputs all element of arrays.

When A is a 1-dimensional array,
MAT PRINT A
is practically equivalent to

```
30 FOR I=LBOUND(A) TO UBOUND(A)
40     PRINT A(I),
50 NEXT I
60 PRINT
```

Provided that MAT PRINT first outputs a new line if the line on which the output starts is not empty.

When A is a 2-dimensional array,
MAT PRINT A
is practically equivalent to

```
20 FOR I=LBOUND(A,1) TO UBOUND(A,1)
30   FOR J=LBOUND(A,2) TO UBOUND(A,2)
40     PRINT A(I,J),
50   NEXT J
60   PRINT
70 NEXT I
```

When A is a 3-dimensional array,
MAT PRINT A
is practically equivalent to

```
10 FOR I=LBOUND(A,1) TO UBOUND(A,1)
20   FOR J=LBOUND(A,2) TO UBOUND(A,2)
30     FOR K=LBOUND(A,3) TO UBOUND(A,3)
40       PRINT A(I,J,K),
50     NEXT K
60     PRINT
70   NEXT J
80   PRINT
90 NEXT I
```

When a semicolon is written succeeding the array as follows, it is practically equivalent to what generated by replacing the comma in line 40 to a semicolon on the above.
MAT PRINT A;

two or more arrays can be written separated by commas or semicolons as
MAT PRINT A;B,C

■ MAT PRINT USING

MAT PRINT USING

MAT PRINT USING format_string: array
A MAT PRINT USING statement handles the elements as if they are arranged in line regardless of the dimension.
The order is that of latter index changing fast.
When the format items are exhausted, a new line is outputted and the format items are used again from the first.
Example.

```
100 DIM A(2,3)
110 MAT READ A
120 DATA 1,2,3,4,5,6
130 MAT PRINT USING " # # # # # #":A
140 MAT PRINT USING " # # #":A
150 MAT PRINT USING " #":A
160 END
```

yields

```
1 2 3 4 5 6
1 2 3
4 5 6
1
2
3
4
5
6
```

Note.
A 2-dimensional array can be outputted finely when the format is given the items of the number of the second indices.
Use a repeat$ function to compose the format string consisting of the needed number of items.
Example.

```
DIM A(3,4)
MAT PRINT USING REPEAT$("  #.####", 4):A
```

Refer to PRINT USING

■ MAT REDIM (original enhancement)

MAT REDIM (original Enhancement)

MAT REDIM
re-defines the lower bounds and the upper bounds of indices of arrays.
Any MAT REDIM statements do not initialize the elements of arrays
Example.

```
 OPTION BASE 1
 DIM A(10)
 MAT A=ZER(0 TO 4)
```

makes the range of indices of A from 1 to 5, while

```
 MAT REDIM A(0 TO 4)
```

makes the range of indices of A from 0 to 4.

Note.
Since this statement is not contained in the standard, it is recommend that this statement should be used only when the lower bounds are needed to be changed.
To change the upper bounds, use

```
MAT A=ZER(n)
MAT A=ZER(m,n)
```

etc.

[Supplement]
If you want to change the lower bounds within the Full BASIC standard, write such an external subprogram as shown below.

```
EXTERNAL SUB REDIM(a(),m,n)
DATA 1          ! dummy data
MAT READ a(m TO m)
MAT a=ZER(m TO n)
END sub
```

Refer to MAT statements

■ Imperative DIM (original enhancement)

Imperative DIM (an enhanced DIM)

This BASIC allows a DIM statement to have numeric expressions in the bound arguments, which is not allowed in the Full BASIC standard.
Example.
DIM A(N),B(2*N)
When a numeric expression except a constant is written as the bound argument, no array elements are arranged until the DIM statement is executed.
If a size of greater than 0 has been assigned once, the size can not be enlarged hereafter.
Note.
An enhanced DIM statement has both features of declarative and imperative.
No internal procedure can include this type of DIM statements.

Graphics

■ Coordinate

Coordinates

BASIC manipulates graphics with problem coordinates□Cwhich are not depend on hardware pixels, but users can set up for problem situation.

SET WINDOW
SET WINDOW statements introduce the coordinates on the pane.

SET WINDOW left , right , bottom , top
@left , right , bottom , and top are numerical expressions.
@The coordinates on which the left-hand side x-coordinate is left, right-hand side coordinate is right, the bottom coordite is bottom, and the top-end coordinate is top shall be introduced.

The shapes of drawing panes of BASIC are principally squares. Thus, the ratio of the virtical length to the horizontal length is 1:1 when the coordinates are set on which right-left =top - bottom.

The left-hand side coordinate and the bottom coordite are 0, the right-hand side coordinate and the top coordinate are 1 unless any SET WINDOW statement is executed.

Example.
SET WINDOW -4,4,0,8
The x-coordinates shall be from -4 to 4□Cthe y-coordinates from 0 to 8.

ASK WINDOW

ASK WINDOW x1, x2, y1, y2
The numerical variables x1, x2, y1 and y2 shall be substituted by the current coordinates.

Color indeces

Color Index

Graphics statements manipulate colors indirectly through color indices, which are nonnegative integers in a certain range.
An arbitrary color can be assigned to any color index.
Color index 0 is a special one, which is the background color.
Graphics statements use color index 1 unless any color index has been specified.

This BASIC allows color indices of from 0 to 255.
Each color index has a pre-defied color as follows.
0 White, 1 Black, 2 Blue, 3 Green, 4 Red, 5 Cyan, 6 Yellow, 7 Magenta, 8 Grey, ... , 15 Silver, ...

Statements

SET COLOR MIX (Color_Index) red, green, blue
assigns the color specified by three numeric values that are between 0 and 1 to Color_Index, where Color_Index is a numeric expression.
Example.
SET COLOR MIX(15) 0,0,1
assigns blue to the color index 15.

ASK MAX COLOR m
Assigns the largest color index to a numeric variable m .
On this version, this value is 255.

ASK COLOR MIX(color_index) R, G, B
The red, green, blue strength of the color assigned to color_index is assigned to three numeric variables R, G, B.
Examle.
If the color of index 2 is blue,
ASK COLOR MIX(2) R,G,B
assigns 0, 0, 1 to R, G, B, respectively .

Original enhancement

SET … COLOR have the form of
SET … COLOR "RED"
Color names are as follows.
"WHITE", "BLACK", "BLUE", "GREEN", "RED", "CYAN", "MAGENTA", "YELLOW", "GRAY", "SILVER"
[Note]
Quotation marks must not be omitted.
For example, SET LINE COLOR RED means setting line color through a numeric variable RED.

Example 2

```
 LET color$="BLUE"
 SET POINT COLOR color$
```

PLOT POINTS

PLOT POINTS

PLOT POINTS: x , y

draws a mark at (x,y), where x,y are numeric expressions.

Two or more points can be written separated by semicolons as follows.
PLOT POINTS: x0, y0; x1, y1; x2, y2

SET POINT STYLE numeric_expression
indicates the mark style.
The mark styles are · for 1, + for 2, * for 3, ○ for 4 and × for 5.
The default point style is 3.

SET POINT COLOR color_index
sets the color index that PLOT TEXT statements use, where color_index is a numeric expression.
The default point color is 1.

ASK POINT COLOR n
assigns current point color to a numeric variable n.

ASK POINT STYLE n
assigns the current point style to a numeric variable n.

ASK MAX POINT STYLE n
assigns the largest point style to a numeric variable n.
On this version, 5 is always assigned to n.

■ PLOT LINES

Plotting Lines

The behavior of a PLOT LINES statement is described using the notion of 'beams'.
The state of a beam is 'on' or 'off'. If the beam moves when it is on, it draws a line.
The beam is set to be off when a program starts.

PLOT LINES: x,y
moves the beam to (x,y), and let it off.

PLOT LINES: x,y ;
moves the beam to (x,y), and let it on.

PLOT LINES
let the beam off.

PLOT LINES: x1, y1 ; x2, y2 ; ··· ; xn , yn
draws the polygonal line through the points specified, and then let the beam off.

[supplement]
The beam shall be off on the following.
(1) Before executing of any other PLOT or GRAPH statement than PLOT LINES.
(2) Before executing of any GET, LOCATE, MAT PLOT, MAT GRAPH, MAT GET, or MAT LOCARE statement.
(3) Before executing of SET WINDOW, SET VIEWPORT, SET DEVICE WINDOW, or SET DEVICE VIEWPORT.
(4) Before and after the invocation of a PICTURE definition.

SET LINE COLOR numeric_expression
sets the color index that is used for line plotting.
The default line color is 1.

SET LINE STYLE numeric_expression
sets the line style that is used for line plotitng.
The line styles are
1 for solid, 2 for dashed, 3 for dotted, 4 for dashed-dotted.
The default line style is 1.

ASK LINE COLOR Numeric_Variable

assigns the current line color.

ASK LINE STYLE Numeric_Variable
assigns the current line style.

ASK MAX LINE STYLE Numeric_Variable
assigns the largest line style, 4 on this version.

Original Enhancements

SET LINE WIDTH numeric_expression
assigns the line width multiplier to the normal width.
On Windows 95/98/Me, if line width is 2 or more, the line style is ignored.

PLOT BEZIER: x1, y1 ; x2, y2 ; x3, y3; x4, y4
draws a Bézier curve of terminal points (x1, y1) , (x4, y4), and control points (x2 , y2), (x3, y3).

SET BEAM MODE "IMMORTAL"
The beam shall remain on the execution of any GET, LOCATE, GRAPH or PLOT statement except PLOT LINES, or on any invocation of a PICTURE definition.

SET BEAM MODE "RIGOROUS"
The beam shall be off according to the Full BASIC standard.

■ PLOT AREA (filled polygon)

PLOT AREA (Filled Polygon)

PLOT AREA: x1, ,y1 ; x2 ,y2 ; ……; xn ,yn
draws a filled polygon of which vertices are (x1 ,y1),(x2 ,y2),……,(xn ,yn) and (x1 , y1) .

SET AREA COLOR numeric_expression
specifies the filling color
The defalut area color is 1.

ASK AREA COLOR Numeric_Variable
assigns the current area color.

[Note]

A point is defined to be internal if any ray beginning that point crosses the boundary an odd number of times.

Enhancements to the Standard

SET AREA STYLE string_expression
The value of the string_expression must be one of followngs.
"HOLLOW", "SOLID", "HATCH"

SET AREA STYLE INDEX numeric_expression
assigns the hatch pattern when area style "HATCH" specifed. Area styles are
1: Horizontal 2：Vertical 3:Slanted(135°) 4:slanted(45°) 5:cross 6:slanted cross

[Note]
Hatch patterns are defined on the physical coordinates.
On this version, if a transformation such as the polygon crosses a line at infinity is assigned, the area style shall be ignored.

ASK AREA STYLE String_variable
assigns the current area style.

ASK AREA STYLE INDEX Numeric_expression
assigns the current area style index.

■ PLOT TEXT

PLOT TEXT

PLOT TEXT ,AT x,y : s$
draws a label consisting of s$ at the point (x,y) ,where s$ is a string expression, and x,y are numeric expressions.
Examples
PLOT TEXT ,AT 0, 0.2: "BASIC"
PLOT TEXT ,AT x, 0: STR$(x)

PLOT TEXT ,AT x,y ,USING Format$: exp1, exp2, ...
draws a label of which literals are those of exp1, exp2, ...
Format$ is a string expression of which value is a format string.
Both numerical expressions and string expressions can be written for exp1, exp2, ...
Example
PLOT TEXT ,AT x,y ,USING "#.### #.###":a,b

SET TEXT HEIGHT a
specifies the text height on the problem coordinate, where a is a numeric expression.

SET TEXT COLOR n
specifies the text foreground color, where n is a numeric expression.
The default text color is 1.

SET TEXT JUSTIFY h$, v$
designates the base point for texts.
h$ is one of "LEFT" , "CENTER" , or "RIGHT".
v$ is one of "TOP","CAP","HALF","BASE",or "BOTTOM"
Example
SET TEXT JUSTIFY "right" , "top"
The default text justify is "LEFT","BOTTOM".

SET TEXT ANGLE a
designates the angle of the text, where a is a numeric expression.
The unit of angle depends on OPTION ANGLE.

ASK TEXT HEIGHT Numeric_Variable
assigns the text height on the problem coordinate

ASK TEXT COLOR numeric_Variable
assigns the current text color

ASK TEXT JUSTIFY String_Variable1 , String_Variable2
assigns the current text justification.

ASK TEXT ANGLE numeric_variable
assigns the current text angle.

Original Enhancement

PLOT LABEL ,AT x,y : String_Expression
PLOT LABEL ,AT x,y ,USING Format_String : expression, expression , …, expression
draw the label.
Only the base point is transformed.
The shape, size and direction are not transformed.

PLOT LETTERS ,AT x,y : String_Expression
PLOT LETTERS ,AT x,y ,USING Format_String : expression, expression , …, expression
draw the label.
The shape is preserved.
Only the base point and the size and direction are transformed.

SET TEXT FONT FontName$, size
specifies the text font name and the size on points.
If the first parameter is null, only the size is changed.
If size is zero, The former size is preserved.
Examples.
SET TEXT FONT "Courier New",12
SET TEXT FONT "Times New Roman Italic" ,0
SET TEXT FONT "",9

SET TEXT BACKGROUND "OPAQUE"
Background of text is to be filled with the color of color index 0.
SET TEXT BACKGROUND "TRANSPARENT"
Background of text is not to be painted.

ASK TEXT WIDTH(s$) n
assigns the width of string expression s$ in the problem coordinates to a numeric variable n.

Incompatibility with the standard
(1) When no transformation is applied, text figures and text angles are managed with respect to the physical coordinates.
(2) The default text height is determined by the font menu of the graphics window.
(3) If Graphics object is a metafile, PLOT TEXT always draw texts with respect to the physical coordinates.

■ MAT PLOT

MAT PLOT

MAT PLOT Statements indicate the points with one 2-dimensional array or two 1-dimensional arrays.

Let x and y be two 1-dimensional arrays with same upper and lower bounds.
MAT PLOT POINTS:x,y
MAT PLOT LINES:x,y
MAT PLOT AREA:x,y

Let M be a 2-dimensional array.
MAT PLOT POINTS:M
MAT PLOT LINES:M
MAT PLOT AREA:M

If the lower and upper bounds of the first dimension are 1 and n,
MAT PLOT POINTS:M
means
PLOT POINTS:M(1,1),M(1,2); M(2,1),M(2,2); ... ; M(n,1),M(n,2)

MAT PLOT POINTS,LIMIT n: x,y
MAT PLOT LINES,LIMIT n: x,y
MAT PLOT AREA,LIMIT n: x,y
MAT PLOT POINTS,LIMIT n: M
MAT PLOT LINES,LIMIT n: M
MAT PLOT AREA,LIMIT n: M
With a numeric expression n, the first n elements are used.

■ MAT PLOT CELLS

MAT PLOT CELLS

MAT PLOT CELLS fills a rectangular area with the colors assigned by a 2-dimensional array.

Let M be a 2-dimensional array.
MAT PLOT CELLS, IN x1 , y1 ; x2 , y2 : M
(x1 , y1) and (x2 , y2) are 2 diagonal vertices of the rectangular.
The point (x1 , y1) is the base point.

Example
```
10 DIM A(2,3)
20 DATA 1,2,3
30 DATA 4,5,6
40 MAT READ A
50 SET WINDOW 0,4,0,4
60 MAT PLOT CELLS,IN 0,0; 2,3 :A
70 END
```
A rectangle whose diagonal vertices are (0,0) and (2,3) is divided into 2 vertically, and 3 horizontally, and each segments are painted with the color assigned with the array A.

■ ASK PIXEL

ASK PIXEL

ASK PIXEL VALUE (x,y) a
assigns the color index of the point(x,y) to a numeric variable a.
If that point is invalid, -1 is assigned.

ASK PIXEL SIZE (x1 , y1 ; x2 , y2) a,b
assigns the numbers of horizontal and vertical pixels of the rectangle of which diagonal vertices is (x1, y1) and (x2, y2) to numeric variables a, b.
The vertices or sides of the rectangle are counted in.

Let M be a 2-dimensional array.
ASK PIXEL ARRAY (x,y) M
assigns the color index of each pixel on the rectangle of which the left-top vertex is (x,y) and the width is the number of first dimension of M and the height is the number of second dimension of M to M.
The value corresponding to an invalid pixel is -1.
[Reference] MAT PLOT CELLS

Example.

```
100 DIM M(501,501)
110 ASK PIXEL SIZE(-0.5,0.5; 0.5,-0.5) A,B
120 MAT M=ZER(A,B)
130 ASK PIXEL ARRAY (-0.5,0.5) M
140 GET POINT: X,Y
150 MAT PLOT CELLS, IN X-1,Y+1;X+1,Y-1 :M
```

This program shall obtain the color indices of the pixels on the interior of the square of side length 1 and of center the origin, and draw the square enlarged two times around the point indicated by a mouse click.

GET POINT

GET POINT

GET POINT: x, y
assigns the coordinates of the point indicated by the mouse click to numeric variables x and y.
Progression of the program is to be suspended until a mouse click.
The beam is to be off.

Refer to PLOT LINES

Let A,B be 1-dimensional arrays, M be a 2-dimensional array.
MAT GET POINT A,B
MAT GET POINT M
obtains the coordinates of the points of number of the size of the array.

Let n be a numeric expression.
MAT GET POINT A(n),B(n)
MAT GET POINT M(n,2)
Before execution, arrays are re-sized.

MAT GET POINT: A(?),B(?)
MAT GET POINT: M(?,)
While the left button of the mouse is pressed, the obtained coordinates are substituted for the arrays.
The size of the arrays are adjusted by the number of the points obtained.
If the amount of the arrays has reached the upper bound, the execution ends.

CLEAR

CLEAR

CLEAR
fills the image with the background color.
The background color is the color of index 0 on the regular draw mode.

■ VIEWPORT, DEVICE WINDOW, DEVICE VIEWPORT

VIEWPORT , DEVICE WINDOW , DEVICE VIEWPORT

Let left, right, bottom, top be numeric expressions.
SET VIEWPORT left, right, bottom, top
SET DEVICE WINDOW left, right, bottom, top
SET DEVICE VIEWPORT left, right, bottom, top

When SET DEVICE VIEWPORT is executed, a device viewport shall be set on the graphics device.
The initial device viewport is the whole graphics device.
The unit of device viewport is a meter.

The SET DEVICE WINDOW statement sets the largest rectangle similar to the specified rectangle inside of the device viewport.
The initial device window is 0,1,0,1.
SET DEVICE VIEWPORT statements and SET DEVICE WINDOW statements erase the image.

The SET VIEWPORT statement assigns the rectangle where graphics statements work with respect to the device window coordinates.
The initial value of the viewport is 0,1,0,1. The four parameters of a SET DEVICE WINDOW or a SET VIEWORT is not less than 0 and not greater than 1.

SET VIEWPORT statements can be used for dividing the drawing pane into two or more sections.
SET VIEWPORT statements and SET WINDOW statements do not erase the image.

Example
Two graphs are drawn on the right upper section and the left lower section.

```
100 OPTION ANGLE DEGREES
110 SET VIEWPORT 0.5, 1, 0.5, 1
120 SET WINDOW -4,4,-4,4
130 DRAW GRID
140 FOR x=-4 TO 4 STEP 0.01
150    PLOT LINES:x,x^3-3*x;
160 NEXT x
170 SET VIEWPORT 0,0.5 , 0,0.5
180 SET WINDOW -180,540,-4,4
190 DRAW GRID(90,1)
200 FOR x=-180 TO 540
```

```
210    PLOT LINES:x,SIN(x);
220 NEXT x
230 END
```

SET CLIP "ON"
SET CLIP "OFF"
If the clip is on, graphics is output on the intersection of the viewport and the device window.
If the clip is off, graphics is output on the device window.
The initial clip is on.

Let w, h be numeric variables, u$ a string variable.
ASK DEVICE SIZE w, h, u$
assigns the width and the height of the device to w and h, "METERS" to u$.
The unit of width and height is a meter.

Let l, r ,b ,t be numeric variables.
ASK DEVICE VIEWPORT l, r, b, t
ASK DEVICE WINDOW l, r, b, t
ASK VIEWPORT l, r, b, t
If the device is not square, an ASK DEVICE VIEWPORT statement can be used to inquire the actual aspect. This is the case the device is a printer.
Refer to Printers and metafiles

■ LOCATE CHOICE, LOCATE VALUE

LOCATE CHOICE, LOCATE VALUE

LOCATE CHOICE : n
displays 8 buttons and assigns the index of the pushed button to a numeric variable n.
LOCATE CHOICE(m) : n
displays m buttons and assigns the index of the pushed button to a numeric variable n, where m is an numeric expression.
The number of buttons is at most 8.

LOCATE VALUE ,RANGE a TO b : x
displays a slide bar, gets a value between a and b and assigns it to a numeric variable x.
LOCATE VALUE ,RANGE a TO b ,AT initial_value : x
A LOCATE-VALUE statement can contain a directive on the initial position on the slide bar.

Original Enhancement
LOCATE CHOICE(1-dimensional-string-array) : n
displays the value of the indexed variables on the buttons, assigns the index of the pushed button to a numeric variable n.
When the lower bound of the array is 1, the index shall coincide with the index on the array.
The number of buttons displayed is the size of the array, provided that it must be at most 8.

AXES and GRID (original enhancement)

Original Enhancement

DRAW GRID
draws a grid with gaps of 1 unit.
DRAW GRID(p,q)
draws a grid with horizontal gaps p, vertical gaps q.
DRAW AXES
draws x-axis and y-axis with ticks
DRAW AXES(p,q)
draws x-axis with ticks of interval p, y-axis with ticks of interval q.

DRAW GRID0
DRAW AXES0
Draws a grid or axes without numeric labels.

Supplied DRAW statements mentioned above use the color of index 15.

■ CIRCLE and DISK (original enhancement)

DRAW CIRCLE, DRAW DISK (original enhancement)

DRAW circle
draws a circle of radius 1 and of center the origin with the line color.
The beam shall be set off.

DRAW disk
draws a circle and fill the interior of it with the area color.
The beam shall remain.

These can be transformed. For example,
if you want to draw a circle of radius r, center (x,y), write
DRAW circle WITH SCALE(r)*SHIFT(x,y)

Refer to picture definition

<Note>
CIRCLE and DISK are not reserved words.
Thus a user can define a picture of its name "CIRCLE" or "DISK".

<Note>
If the circle crosses a line at infinity as a result of transformation, The DRAW statements above do not draw proper figures.
On such a case, the following picture definition is effective.

```
1000 EXTERNAL PICTURE circle
1010 OPTION ANGLE DEGREES
1020 FOR t=0 TO 360
1030    PLOT LINES : COS(t),SIN(t);
1040 NEXT t
1050 END PICTURE
2000 EXTERNAL PICTURE disk
2010 OPTION ANGLE DEGREES
2020 OPTION BASE 0
2030 DIM x(359),y(359)
2040 FOR t=0 TO 359
2050    LET x(t)=COS(t)
2060    LET y(t)=SIN(t)
2070 NEXT t
2080 MAT PLOT AREA : x,y
2090 END PICTURE
```

■ MOUSE POLL (original enhancement)

Real-time mouse input (Original Enhancement)

MOUSE POLL x, y, left, right
assigns the current mouse position to numeric variables x, y,
and the current mouse state to numeric variables left, right,
1 for button pressed, 0 for not pressed.

■ PIXELX, PIXELY, PROBLEMX, PROBLEMY (original enhancements)

Pixel Coordinates (original enhancement)

Pixel Coordinates are the coordinates whose origin is left-bottom end.

Supplied Functions (Original Enhancement)

Let x, y be numeric expressions below.

PIXELX(x) Pixel coordinate for problem coordinate x
PIXELY(y) Pixel coordinate for problem coordinate y

PROBLEMX(x) Problem coordinate for pixel coordinate x
PROBLEMY(y) Problem coordinate for pixel coordinate y

Example
To manipulate all pixels

```
SET WINDOW left, right, bottom, top
FOR i=0 TO PIXELX(right)
  FOR j=0 TO PIXELY(top)
    LET x=PROBLEMX(i)
    LET y=PROBLEMY(j)
    ……
  NEXT j
NEXT i
```

[supplement]
The above functions can be defined in Full BASIC as follows.

```
EXTERNAL FUNCTION PIXELX(x)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
LET PIXELX=ROUND((px-1)*(x-left)/(right-left),0)
END FUNCTION

EXTERNAL FUNCTION PIXELY(y)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
```

```
LET PIXELY=ROUND((py-1)*(y-bottom)/(top-bottom),0)
END FUNCTION

EXTERNAL FUNCTION WORLDX(x)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
LET WORLDX=(right-left)*x/(px-1)+left
END FUNCTION

EXTERNAL FUNCTION WORLDY(y)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
LET WORLDY=(top-bottom)*y/(py-1)+bottom
END FUNCTION
```

■ SET BITMAP SIZE (original enhancement)

SET BITMAP SIZE (original enhancement)

SET BITMAP SIZE width, height
sets the number of pixels of the bitmap, where width, height are numeric expressions.
This statement can be executed only on the bitmap graphics mode.
The problem coordinates are set to 0 at left end, 1 at right end, 0 at bottom end, 1 at top end.
This statement is equivalent to the following Full BASIC codes except that it does not initialize the image.

```
IF width=height THEN
  SET DEVICE WINDOW 0,1,0,(height-1)/(width-1)
  SET VIEWPORT 0,1,0,(height-1)/(width-1)
ELSE
  SET DEVICE WINDOW 0,(width-1)/(height-1),0.1
  SET VIEWPORT 0,(width-1)/(height-1),0.1
END IF
SET WINDOW 0,1,0,1
```

■ FLOOD and PAINT (original enhancements)

Filling (Original enhancement)

Let x and y be numeric expressions.

FLOOD x,y
fills the area where any point can be connected to the point (x, y) through the points of the same color with the current area color.

PAINT x,y
fills the area of which boundary consists of the points of the line color and that contains the point (x, y) with current area color.

<Note>
These two commands are not avilable on Linux or MAC.
FLOOD is compatible with that of True BASIC.
PAINT is functionally compatible with that of Microsoft BASIC.

Refer to PLOT AREA.

■ GLOAD and GSAVE (original enhancements)

Loading or saving Images (original enhancement)

FileName$ below is a string expression .

GLOAD FileName$
loads the image to the graphics pane from the file.
Available image formats are BMP, GIF and JPEG.
The pane size is adjusted with the size of the image, and the problem coordinates are set to 0 for left end, 1 for right end, 0 for bottom end, 1 for top end.
Example
GLOAD "C:\WINDOWS\winnt.bmp"

[Note]
SET WINDOW 0,PIXELX(1),0,PIXELY(1)
sets the pixel-wise problem coordinates.

[Note]
GLOAD statement is valid only on the Bitmap Graphics Mode.

GSAVE FileName$
saves the whole image to the file.
If the graphics mode is the bitmap mode, FileName$ must contain the extension ".BMP", ".GIF", or ".JPG".
If the graphics mode is the metafile mode, FileName$ must contain the extension ".EMF".
If the graphics mode is the metafile mode, execution of GSAVE clears the image.

[supplement]
If the format is JPEG, compression rate can be assigned as following.
Example
GSAVE "A:SAMPLE1.JPG", "33%"

■ SET COLOR MODE (original enhancement)

SET COLOR MODE (original enhancement)

SET COLOR MODE "NATIVE"
transfers to the mode of using 24-bit integers for color indices.
On this mode, a color index is the 24-bit integer of which the lower 8-bit indicates red, the middle 8-bit green, the upper 8-bit blue.
ASK PIXEL ARRAY or ASK PIXEL VALUE can obtain every color.
But on this mode, the SET COLOR MIX statement is invalid.
the largest color index is 16777215.
The image shall not be cleared after SET COLOR MODE is executed.
Background color, line color, point color, area color, and text color shall be inherited.
Thus CLEAR fills using the color that had been assigned by SET COLOR MIX(0) before the mode transferred.

SET COLOR MODE "REGULAR"
transfers the color mode to regular.
Line color, text color, point color, area color shall be set to 1.

Supplied Function

COLORINDEX(r,g,b)
The color index of color of the strength red r, green g, blue b, where each strength is between 0 and 1.

Example
```
10 SET COLOR MODE "NATIVE"
20 SET TEXT COLOR ColorIndex(0,0,1)
30 PLOT TEXT ,AT 0,0:"Hello"
40 END
```

[Note]
BVAL functions can be used for assigning the color index using a hexadecimal number.
For example, the color blue is BVAL("FF0000",16).
Note that the byte order is the reverse of that in HTML.

■ SET DRAW MODE (original enhancement)

Special Effects (Original enhancement)

SET DRAW MODE HIDDEN
Transfers to the mode of drawing only on the internal bitmap.

SET DRAW MODE EXPLICIT
Reflects the internal image, and transfers to the mode of drawing both on the screen and internal bitmap.

■ Metafiles and printers

Printers and MetaFile

Option Menu - Graphics enables us to create a metafile or send graphics command directly to the printer.

1. Setting Printer
Before the execution of the program, set up the printer.
On the Option Menu - Graphics dialog, click Printer Setting button and select the printer and the paper.

2. About the default drawing pane
When Image Format MetaFile(Printer) or Printer(direct) is selected, the default drawing pane is the largest square contained in the printable area and containing the left-bottom end of the printable area.

3. Alteration of drawing pane
(1) Alternation of aspect ratio
The following code changes the drawing pane into the rectangle of which the ratio of the height to the width is 4 : 3, on the assumption that the Paper Orientation is portrait.

```
140 SET DEVICE WINDOW 0, 3/4, 0, 1
150 SET VIEWPORT      0, 3/4, 0, 1
160 SET WINDOW  -3, 3,  -4, 4
170 DRAW grid
180 DEF f(x)=x*(x-1)*(x+1)
190 FOR x=-3 TO 3 STEP 0.01
200    PLOT LINES: x,f(x);
210 NEXT x
220 END
```

Explanation
In order to use the rectangle of which the ratio of the width to the height is 3:4, execute a SET DEVICE WINDOW statement and a SET VIEWPORT statement with the parameters 0, 3/4, 0, 1.
When the Paper Orientation is landscape, to change the ratio of the height to width into 3 : 4, for example, modify the lines as follows.

```
140 SET DEVICE WINDOW 0, 1, 0, 3/4
150 SET VIEWPORT      0, 1, 0, 3/4
```

(2) Additional Margin
Shrinking device viewport enlarges margins.

For example, the following enlarges the left, right, top, bottom margins by 1cm each.
120 ASK DEVICE VIEWPORT dvleft, dvright, dvbottom, dvtop
130 SET DEVICE VIEWPORT dvleft+0.01, dvright-0.01, dvbottom+0.01, dvtop-0.01
(Note.) The measure of the device viewport is a meter when the Image Format is a metafile or the printer.

(3) Designation of a size
If you need the drawing pane of width w(mm) and height h(mm) , use the following codes if w>h.
ASK DEVICE VIEWPORT dvleft, dvright, dvbottom, dvtop
SET DEVICE VIEWPORT dvleft, dvleft + w/1000, dvbottom, dvbottom + h/1000
SET DEVICE WINDOW 0, 1, 0, h/w
SET VIEWPORT 0, 1, 0, h/w
On the case of w<h, the last 2 lines should be altered as follows.
SET DEVICE WINDOW 0, w/h, 0, 1
SET VIEWPORT 0, w/h, 0, 1

4. Supplement
(1) ASK DEVICE SIZE w,h,s$
assigns the width, the height of the printable area to numeric variables w, h and "meters" to a string variable s$. The measure is a meter.
(2) The following statement returns a meaning value. The measure is a pixel.
ASK PIXEL SIZE
(3) The following statements return -1. Note that they do not cause an exception.
ASK PIXEL VALUE
ASK PIXEL ARRAY
(4) The following statements cause exceptions.
LOCATE POINT
GET POINT
(5) When the Image Format is Printer(direct), execution of CLEAR makes a page feed.

Refer to Option Menu - Graphics

Control structures

■ Logical expressions

Logical Expressions

Comparison operators

A comparison operator is one of
= , <> , < , > , <= , >= .

Comparisons
A comparison is
numeric_expression conparison_operator numeric_expression
or
string_expression comparison_operator string_expression

Logical Expressions
A Logical expression is comparisons combined by AND, OR, or NOT.
NOT is written just before the comparison that is to be negated.
AND or OR is written between the two comparisons.
The priority is in order of NOT , AND , OR.
Note that AND has the higher priority than that of OR.
Parentheses can be used for changing priority.
Examples.
NOT(a=1 OR a=2)
a=1 AND (b=0 OR c=0)

Evaluation of logical operation
On the expression ' p AND q', p is evaluated first and if it is false then q shall not be evaluated.
Similarly, on the expression' p OR q', p is evaluated first and if it is true then q shall not be evaluated.

Example.
When the upper bound of an array x is 10 and the value of a variable i is 11, the evaluation of an expression
i<=10 AND x(i)=0
does not cause an exception.

■ IF ... END IF

IF ~ END IF

(1)
IF logical_expression THEN
  ……
END IF

If logical_expression is true, the part …… shall be executed.

The part …… can consist of multiple statements.

Example.
```
 10 ! Solve a quadratic equation.
 20 INPUT a, b, c
 30 LET D=b^2-4*a*c
 40 IF D>=0 THEN
 50    PRINT (-b + SQR(D))/(2*a), (-b-SQR(D))/(2*a)
 60 END IF
 70 END
```

(2)
```
IF logical_expression THEN
  ……1
ELSE
  ……2
END IF
```

if logical_expression is true, the part ……1 shall be executed,
otherwise the part ……2 shall be executed.
Example.
```
 10 ! Solve a quadratic equation.
 20 INPUT a, b, c
 30 LET D=b^2-4*a*c
 40 IF D>=0 THEN
 50    PRINT (-b + SQR(D))/(2*a), (-b-SQR(D))/(2*a)
 60 ELSE
 70    PRINT "no solution"
 80 END IF
 90 END
```

■ ELSEIF

ELSEIF

(1) An IF ~ END IF block can contain an ELSEIF line.

```
IF logical_exp1 THEN
```

```
  ……1
ELSEIF logical_exp2 THEN
  ……2
ELSE
  ……3
END IF
```

is equivalent to

```
IF logical_exp1 THEN
  ……1
ELSE
  IF logical_exp2 THEN
    ……2
  ELSE
    ……3
  END IF
END IF
```

If logical_exp1 is true, ……1 shall be executed, otherwise logical_exp2 is true, ……2 shall be executed, otherwise ……3 shall be executed.

Example.

```
 ! Solve a quadratic equation
 INPUT a, b, c
 LET D=b^2-4*a*c
 IF D>0 THEN
    PRINT (-b+sqr(D))/(2*a),(-b-sqr(D))/(2*a)
 ELSEIF D=0 THEN
    PRINT -b/(2*a)
 ELSE
    PRINT "no solution"
 END IF
 END
```

(2) An IF ~ END IF block can contain more than one ELSEIF line.

```
IF logical_exp1 THEN
……1
```

ELSEIF logical_exp2 THEN
……2
ELSEIF logical_exp3 THEN
……3
~ ~ ~
~ ~ ~
ELSEIF logical_expn-1 THEN
……n-1
ELSE
……n
END IF

■ IF statement

IF statement(IF line)

IF logical_expression THEN imperative_statement
IF logical_expression THEN imperative_statement1 ELSE imperative_statement2

Example.
IF D>0 THEN PRINT (-b+sqr(D))/(2*a),(-b-sqr(D))/(2*a)
IF A=1 THEN PRINT "Y" ELSE PRINT "N"

Only one **imperative statement** can be written succeeding THEN and only one imperative statement can be written succeeding ELSE (if any).
An imperative statement is a statement that is neither declarative nor part of a block.
Many statements such as LET, PRINT, INPUT, SET, PLOT, GOTO, EXIT, CALL, STOP are imperative, but a IF statement is not imperative.
Thus, no IF statement can be written succeeding THEN.
Therefore, Full BASIC never generate an obscure statement like
IF a=0 THEN IF b=1 THEN … ELSE …

Note.
An IF-line that has no statement succeeding THEN is regarded as part of an IF ~ END IF block, and consequently a corresponding END-IF-line is needed.

■ SELECT ... END SELECT

SELECT CASE

The block succeeding the CASE-line which contains the item that agrees with the value of the expression written at the SELECT-CASE-line shall be executed.
If no CASE-item agrees, the block succeeding the CASE-ELSE-line shall be executed.
Example.

```
INPUT m
SELECT CASE m
CASE 1,3,5,7,8,10,12
  LET d=31
CASE 4,6,9,11
  LET d=30
CASE 2
  LET d=28
CASE ELSE
  PRINT "?"
END SELECT
PRINT d
END
```

More than one item can be written in a CASE-line punctuated by commas.
An item must consist of only constants.
An item is a constant, a constant TO a constant, or IS a comparison operator a constant.
Example.

```
SELECT CASE 2*x-3
CASE 0 TO 2
  PRINT "y"
CASE IS <0, IS >2
  PRINT "n"
END SELECT
```

The items are tested downward from the top. Once an item agrees, the rest CASE items are not tested.
If a CASE-ELSE-line is omitted, an exception of EXTYPE 10004 shall be raised when no item agrees.
Either a numeric expression or a string expression can be written in the SELECT-CASE-line.

■ DO ... LOOP

DO ~ LOOP

On a DO ~ LOOP block, each statement written between the DO-line and the LOOP-line is executed repeatedly.

Example 1.

```
10 ! Repeats infinitely
20 LET a=0
30 DO
40    PRINT a
50    LET a=a+1
60 LOOP
70 END
```

DO WHILE

When a WHILE-clause is written in the DO-line, each statement written between the DO-line and the LOOP-line is executed repeatedly while the condition (logical expression) written in the WHILE-clause is true. If the condition is false from the first, no statement between the DO-Line and the LOOP-line is executed.

Example 2.

```
10 INPUT n
20 LET i=1
30 DO WHILE i<=n
40    PRINT i
50    LET i=i+1
60 LOOP
70 END
```

DO UNTIL

When a UNTIL-clause is written in the DO-line, each statement written between the DO-line and the LOOP-line is executed repeatedly while the condition written in the UNTIL-clause is false. If the condition is true from the first, no statement between the DO-Line and the LOOP-line is executed.

Example 3 (equivalent to Example 2).

```
10 INPUT n
20 LET i=1
30 DO UNTIL i>n
40    PRINT i
50    LET i=i+1
```

```
60 LOOP
70 END
```

LOOP WHILE

When a WHILE-clause is written in the LOOP-line, after executing each statement between the DO-Line and the LOOP-line, the condition written in the WHILE-clause shall be tested. If the result is true, the control goes back to the DO-Line.

Example 4.

```
10 INPUT n
20 LET i=1
30 DO
40    PRINT i
50    LET i=i+1
60 LOOP WHILE i<=n
70 END
```

If a number greater than 1 is entered, Example 1 and Example 4 act as the same, but if 0 is entered, the 40-line and the 50-line of Example 2 never be executed, while the 40-line and the 50-line of Example 4 shall be executed once respectively.

LOOP UNTIL

When a UNTIL-clause is written in the LOOP-line, after executing each statement between the DO-Line and the LOOP-line, the condition written in the WHILE-clause shall be tested. If the result is false, the control goes back to the DO-Line.

Example 5 (equivalent to Example 4).

```
10 INPUT n
20 LET i=1
30 DO
40    PRINT i
50    LET i=i+1
60 LOOP UNTIL i>n
70 END
```

EXIT DO

If a EXIT DO statement is executed, the control branches to the next line of the LOOP-line.

Example 6.

```
10 INPUT n
20 LET i=1
30 DO
40    PRINT i
50    IF i>=n THEN EXIT DO
60    LET i=i+1
70 LOOP
```

```
80 END
```

An EXIT DO statement can be written in a DO ~ LOOP block. If EXIT DO statement has two or more DO ~ LOOP blocks that include it, the execution of it means the escape from the most inner DO-block. For example, if the EXIT DO statement in the 50-line is executed, the control branches to the 80-line on the following program.

```
10 DO
20   ……
30    DO
40       ……
50       EXIT DO
60       ……
70    LOOP
80    ……
90 LOOP
100 ……
```

If you want to switch the control to the 100-line from the location of the 50-line, use a GOTO statement.

■ FOR ... NEXT

FOR ~ NEXT (FOR-block)

A FOR ~ NEXT block is written as the form

```
FOR Control_Var = Initial_Value TO Limit STEP Increment
……
NEXT Control_Var
```

Control_Var is a numeric simple variable (any indexed variable not allowed).
Initial_Value, Limit, and Increment are numeric expressions.
NEXT statement must have the same variable written in the FOR statement.

Limit and Increment are computed only when the FOR statement is executed, those values remain unchanged while the repetition progresses.

If the increment is positive, the repetition acts as follows.
(1) Initial_Value is assigned to Control_Var.
(2) If Control_Var is greater than the limit, the control branches to the next line of the NEXT line.
(3) When the NEXT line is executed, Control_var is added the increment, and the control goes back to (2).
If the increment is negative, the comparison in (2) is inverted.

Example
```
10 FOR N=10 TO 1 STEP -1
20   PRINT N
30 NEXT N
40 END
```

"STEP Increment" can be omitted. On that case, the increment is assumed to be 1.

EXIT FOR
Use EXIT FOR to escape from a FOR ~ NEXT repetition.
Example.
```
10 INPUT n
20 FOR f=2 TO n-1
30   IF MOD(n,f)=0 THEN
40     PRINT f
50     EXIT FOR
60   END IF
70 NEXT f
80 END
```

If a EXIT FOR has nested FOR ~ NEXT blocks that include it, the execution of it means the escape from the most inner FOR block.

■ STOP and GOTO Control Statements

Control Statements

STOP statements
STOP
stops the exception and then lets the program finish.

GOTO statements
GOTO line_number
lets the control go to the line of line_number.

Note.
The control must not be transferred by a GOTO statement as follows.
- From outside to inside of a block.
- From inside to outside of an internal procedure.
- From inside to outside of an exception handling block.

■ PROGRAM and CHAIN

PROGRAM, CHAIN

A PROGRAM statement declares a program name and parameters, which are used by CHAIN statements.
A CHAIN statement transfers the control to another program and terminates the program itself.

PROGRAM statements

PROGRAM Program_Name
PROGRAM Program_Name(variable, variable , variable , ··· , variable)
The naming rule for Program_Name is same as that for numeric variables.
Example.
PROGRAM ABC(a,s$)
When arrays are written as parameters, they are written followed by parentheses and commas of the number of the dimension minus 1.
Example.
PROGRAM SAMPLE(A(),B(,),C(,,))
A is a 1-dimensional array, B 2-dimensional, C 3-dimensional.

A PROGRAM statement must be written in the first line of a program.
When a CHAIN statement has launched this program, the values that the CHAIN statement has handed over are substituted for the parameters.
A PROGRAM statement can be used for a program that is not necessarily written by BASIC to launch the program and hand over some parameters.

Exceptions
extype4301 Parameter mismatch
extype4302 Mismatched dimension of array parameter

CHAIN statements

CHAIN File_Name
CHAIN File_Name WITH (expression , expression , … , expression)
File_Name is a string expression whose value shall be the name of the file that stores a BASIC program.
The values of expressions written in the WITH-clause shall be the parameters for the PROGRAM statement.
Expressions are numerical expressions, string expressions or arrays.

Arrays are written without parentheses.
Example 1.
CHAIN "FRACTAL\DRAGON.BAS"
Example 2.
CHAIN "A:\ABC.BAS" WITH (12,"Quiz")
Example 3.
DIM A(10),B(3,3),C(2,3,4)
CHAIN "A:\SAMPLE.BAS" WITH(A,B,C)

(Note)
Although the program name that is written in the PROGRAM statement can differ from the name of the file which stores the program, the argument of a CHAIN statement is not a program name but a file name.

Refer to command line parameters

■ REM (!), DIM, DECLARE, OPTION, and DATA Declarative Statements

Declarative Statements

Declarative Statements
REM, DIM, DECLARE, OPTION, and DATA statements are called declarative statements.
Declarative statements are valid only if they are just written.
Any declarative statement is processed on compiling, and is not an object of the control.
Example.
The following code let the lower bound of indices of B to be 0 even if the value of A is non-positive.

```
20 IF A>0 THEN
30    OPTION BASE 0
40 END IF
50 DIM B(10)
```

In addition, the scope of a declarative statement is the program unit.

■ WAIT DELAY, PAUSE

WAIT DELAY and PAUSE

WAIT DELAY n
halts the control for n seconds. (The fractional part of n is valid)
(Note.) WAIT DELAY is defined in the real-time module of Full BASIC.

PAUSE
waits for Enter key pressed.

PAUSE string_expression
waits for Enter key pressed with displaying string_expression.

EXECUTE program (original enhancement)

EXECUTE (Original Enhancement)

EXECUTE statements (Original Enhancement)

EXECUTE File_Name
EXECUTE File_Name WITH (expression , expression , ··· , expression)
EXECUTE NOWAIT File_Name
EXECUTE NOWAIT File_Name WITH (expression , expression , ··· , expression)
In the above, File_Name is a string expression whose value is a file name.

(Meanings)

When an executable file is designated, that file shall be executed.
In such a case, if the EXECUTE statement has a WITH-clause, expressions shall be evaluated and given to the program as command-line parameters, with numbers converted to strings, and strings quoted.
Example.
EXECUTE "c:\windows\notepad.exe" WITH ("C:\BASICw32\README.TXT")

When a file which has an extension for BASIC programs is designated , BASIC.EXE shall be launched to run the BASIC program. The values of expressions are passed to the PROGRAM statement as parameters.

When a file excluding the above is designated, it shall be opened using the file association. The values of expressions are passed as additional parameters.

If an EXECUTE statement has NOWAIT, the control goes to the next line immediately, otherwise the control waits for the finish.

The file designated by File_Name shall be searched in the directories in order of the current directory, the directory where BASIC.EXE lies, and the directories designated in the PASS environment variable of Windows.

An EXECUTE statement is able to launch BASIC.EXE to run a BASIC program.
In such a case, the BASIC program file name is written in the WITH-clause.
When the BASIC program has a PROGRAM statement, the parameters are written succeeding the file name in the WITH-clause.
The start-up parameter for BASIC.EXE is written preceding the BASIC program file name in the WITH-clause.
The arguments written in the WITH-clause are all punctuated by commas.
Example.
EXECUTE "A:\ABC.BAS"

EXECUTE "BASIC.EXE" WITH("A:\ABC.BAS")
The launched BASIC.EXE terminates after the execution of "A:\ABC.BAS".
EXECUTE "BASIC.EXE" WITH("/OR","A:\ABC.BAS")
The launched BASIC.EXE stands by after the execution of "A:\ABC.BAS".
EXECUTE "BASIC.EXE" WITH("/NR","A:\ABC.BAS")
The launched BASIC.EXE stands by after loading "A:\ABC.BAS".
EXECUTE "A:\ABC.BAS" WITH(12, "Quiz")
EXECUTE "BASIC.EXE" WITH("A:\ABC.BAS" ,12, "Quiz")
If ABC.BAS has a PROGRAM statement, that program shall be executed with 12 and "Quiz" substituted for the program parameters.
EXECUTE NOWAIT "A:\ABC.BAS"
EXECUTE NOWAIT "BASIC.EXE" WITH("A:\ABC.BAS")
The current program and "A:\ABC.BAS" are executed simultaneously.

< Notice >
The arguments in the WITH-clause are converted into character representations and supplemented to the command-line. Note that the length of a command-line has the upper limit that may not be plentiful due to the specification of Windows.

Utilization of file association

PLAY File_Name
PLAY NOWAIT File_Name
plays the file using the file association.
(Refer to PlaySound)

ASSOC PRINT File_Name
prints the file using the file ssciation.

■ SWAP, BEEP and PLAYSOUND (miscellaneous original enhancements)

Original Enhancement (miscellaneous)

SWAP x,y
interchanges the values of variables x and y.
Same as Microsoft BASIC's SWAP.
Notice that SWAP is not contained in the standard.

BEEP
plays the sound specified as the Windows Default Beep sound.

BEEP exp1, exp2
invokes the BEEP function on Windows API.
exp1 and exp2 mean the frequency in hertz and the duration in milliseconds. Note that exp1 and exp2 have no efficacy on Windows 95/98/Me.

PLAYSOUND File_Name
plays the sound file using Windows API.
PLAYSOUND File_Name ,ASYNC
plays the sound file asynchronously, that is, does not wait for the finishing of the play.

Exception Handling

■ Exception Handling

Exception Handling

An error that occurs during the execution of a program is called an exception.
Exception handling enables the program to continue by describing the process that shall be executed when an exception has occurred.

Example 1. A program without exception handling.

```
10 OPTION ANGLE DEGREES
20 SET WINDOW 0,180,-10,10
30 FOR x=0 TO 180 STEP 0.1
40    PLOT LINES: x,TAN(x);
50 NEXT x
60 END
```

This program stops at x=90 for an exception.

Example 2. A program with exception handling.

```
10 OPTION ANGLE DEGREES
20 SET WINDOW 0,180,-10,10
30 FOR x=0 TO 180 STEP 0.1
40    WHEN EXCEPTION IN
50       PLOT LINES: x,TAN(x)
60    USE
70       PLOT LINES
80    END WHEN
90 NEXT x
100 END
```

If an exception occurs on the 50-line, the control jumps to the 70-line.
That is, on this program, the 70-line is the process that is executed when an exception has occurred on the 50-line.

When EXCEPTION IN ... USE ... END WHEN

WHEN EXCEPTION IN ~ USE ~ END WHEN

WHEN EXCEPTION IN
    when-body
USE
    exception-handler
END WHEN

When-body
The portion between a WHEN-EXCEPTION-IN line and a USE line is called a when-body.
If the when-body has been completed normally, the cotrol proceeds to the the next line of the END-WHEN. Otherwise, the control branches to the exception-handler.

Exception-handler
The lines written between a USE line and a END-WHEN line is called an exception-handler.
If the control has reached the END-WHEN line, the control escapes the when-block and proceeds to the next line.
No branch out of an exception-handler using GOTO or IF-THEN-line_number is allowed.

The followings are statements that can be used in an exception-handler.

RETRY
resumes the statement that has caused the exception.

CONTINUE
resumes the next, provided that the line where the exception has occurred is the beginning or part of a block, the control proceeds to the next line of the block. For example, if an exception has occurred in a FOR line, the control proceeds to the next line of the NEXT line. If an exception has occurred in a ELSEIF-line, the control proceeds to the next line of the corresponding END-IF.

■ Propagation of an Exception

Propagation of an Exception

If an exception has occurred on the execution of a function definition or a subprogram, the exception is propagated to the statement invoking it unless the line is within a when-block.

Example. The graph of y=1/x

```
100 DEF f(x)=1/x
110 SET WINDOW -2,2,-2,2
120 FOR x=-2 TO 2 STEP 0.01
130   WHEN EXCEPTION IN
140     PLOT LINES:x,f(x);
150   USE
160     PLOT LINES
170   END WHEN
180 NEXT x
190 END
```

Explanation.
If the PLOT-line at 140-line is executed when x=0, the DEF-statement at 100-line is executed and it causes an exception of zero division. And then the exception is passed back to 140-line, the exception is handled with the WHEN-block.

[Supplement]
If an exception is propagated to a statement invoking a routine, 100000 is added to the value of EXTYPE.

■ Nonfatal Exception

Non-fatal exceptions

There are two types of exceptions, fatal and no-fatal.
Ordinal errors such as division-by-zero and overflow are fatal exceptions.
Non-fatal exceptions are raised by specific statements or functions such as INPUT, PRINT USING, USING$-functions, BREAK-statements.
A non-fatal exception can be raised only if it is contained in a when-body.

INPUT
An INPUT statement has the default recovery procedure. Ordinarily an INPUT statement itself manages insufficiency or excess of data or inconsistency of data type to request re-input.
However, when an INPUT statement is written in a when-body, insufficiency or excess of data or inconsistency of data type raises an exception, and then let the control branch to the exception handler.
Example.

```
10 WHEN EXCEPTION IN
20    INPUT A,B$
30 USE
40    PRINT EXTYPE
50 END WHEN
```

The value of EXTYPE is as follows.
8102 syntactically incorrect input reply
8103 Non-numeric datum for a numeric variable.
8002 Too few data
8003 Too many data (or extra comma on the tail)

When an INPUT statement is written in a when-body, it is recommended to use a RERTY statement as follows.

```
100 WHEN EXCEPTION IN
110    INPUT A
120 USE
130    SELECT CASE EXTYPE
140    CASE 8102,8103,8002,8003
150       RETRY
160    CASE ELSE
170       ! Manages other exceptions
180    END SELECT
190 END WHEN
```

PRINT USING , USING$
When a PRINT USING statement or a USING$ function is written in a when-body, the following exceptions can be raised.
EXTYPE
8203 Format-item too short
8204 Exrad overflow

BREAK
If a BREAK is executed on a when-body, it merely raises an exception of extype 10007, and does not display the debug window. Therefore it is difficult to debug a when-body using a BREAK statement.
(However, using Break-Points on the debug window enables us to debug a when-body.)

■ CAUSE EXCEPTION

CAUSE EXCEPTION

CAUSE EXCEPTION n
raises an exception of extype n rounded to an integer, where n is a numeric expression.
Available exception numbers are from 1 to 999 (reserved for users) and pre-defined extypes.

If a CAUSE EXCEPTION statement is executed in an exception handler, an exception shall be raised again. If a when-body includes the when-block that the exception handler belongs to, the control branches to the exception handler related to the when-block. Otherwise if no when-body includes the when-block, when the CAUSE EXCEPTION statement is located in a procedure, the exception shall be propagated to the invoking statement.

EXIT HANDLER

EXIT HANDLER

EXIT HANDLER
cancels the execution of the exception handler and then restores the control to the state of exception.
EXIT HANDLER can be written only in an exception handler.

EXIT HANDLER resembles CAUSE EXCEPTION EXTYPE, but when an EXIT HANDLER statement is executed, the origin becomes the statement that first caused an exception. Therefore, differences shall be made if a RETRY or CONTINUE statement is executed in a superior exception handler.

Example 1. 145-line shall be executed.

```
100 WHEN EXCEPTION IN
110   WHEN EXCEPTION IN
120     PRINT 1/0
125     PRINT "125"
130   USE
140     CAUSE EXCEPTION EXTYPE
145     PRINT "145"
150   END WHEN
160 USE
170   CONTINUE
180 END WHEN
190 END
```

Example 2. 125-line shall be executed.

```
100 WHEN EXCEPTION IN
110   WHEN EXCEPTION IN
120     PRINT 1/0
125     PRINT "125"
130   USE
140     EXIT HANDLER
145     PRINT "145"
150   END WHEN
160 USE
170   CONTINUE
180 END WHEN
190 END
```

■ WHEN EXCEPTION USE and HANDLER

WHEN EXCEPTION USE and HANDLER

Exception handling process can be described as an internal procedure so that more than one when-block has an identical exception-handler in common.

An exception handler described as an internal procedure is called a detached handler.
A detached handler begins with a HANDLER line and ends with an END-HANDLER line.
A HANDLER line holds a name of the handler. The rule for naming a handler is same as that of a numerical variable.
When you use a detached handler, replace WHEN EXCEPTION IN ~ USE ~ END WHEN with WHEN EXCEPTION USE ~ END WHEN. A WHEN-EXCEPTION-USE line holds the name of the detached handler.
Example

```
100 WHEN EXCEPTION USE h
110    PRINT 1/A
120 END WHEN
130 WHEN EXCEPTION USE h
140    PRINT SQR(-1)
150 END WHEN
160 HANDLER h
170    PRINT EXTYPE
180 END HANDLER
190 END
```

EXTYPE Function

EXTYPE function

EXTYPE
returns the number that means the exception type.
EXTYPE can be written only in an exception handler.

Note. EXTYPE is a reserved word.

Principal EXTYPEs
1000 level Overflow
  1002 Overflow in a numeric expression
  1003 Overflow in a numeric supplied function
  1004 Overflow in a VAL function
  1005 Overflow in a numeric array
  1006 Overflow in a READ statement
  1008 Overflow in a file input
  1009 Overflow in a DET or DOT function
2000 level Subscript Errors
  2001 Subscript out of bounds
3000 level Math Errors
  3001 Division by zero
  3002 Negative to the non-integral power
  3003 Zero to the negative power
  3004 Logarithm of zero or negative
  3005 SQR of a negative number
  3006 Zero divisor in MOD or REMAINDER
  3007 Out of range in ACOS or ASIN
  3008 ANGLE(0,0)
  3009 INV of a singular matrix
4000 level Parameter Errors
  4001 VAL of non-numeric-constant
  4002 CHR$ Out of Range
  4003 ORD Out of Range
  . . . .
5000 level Array Storage Errors
6000 level Matrix Errors

7000 level File Usage Errors
8000 level I/O Errors
  8001 READ out of data
  8011 End of File
  8012 Too Few Data in a record
  8013 Too Many Data in a record
  8101 Non-numeric constant on file input
  8105 Syntactically incorrect input reply from file
  8120 Type Mismatch on INTERNAL input
  8201 Incorrect Format String
  8202 No Format Item
  8401 Timeout on INPUT
9000 level Device Errors
10000 level Control Errors
11000 level Graphics Errors
See also Nonfatal Errors.

Implementation defined EXTYPEs
-3009 Invalid operation on transformation
-1005 Overflow on transformation
-103 Out of Memory for Arrays
-102 Out of memory for Simple Variables
-101 System Stack Overflow
-100 Out of memory for the execution
7101 Incorrect File Description
7102 File Not Exist
9003 Failed to Delete File
9004 File Already Exist

Propagation of Exception
If an exception is propagated out of the procedure, EXTYPE becomes the value obtained by adding 100000 to the initial extype, while if the initial extype is negative, EXTYPE becomes the value obtained by subtracting 100000 from it.
Note.
The extype before propagated can be obtained using REMAINDER(EXTYPE,100000).

Procedures

- **DEF**

DEF Statement

DEF statements enables us to write functions defined by complicated expressions simply as f(x), g(x,y) and so on.
Example.

```
10 DEF f(x,y)=x^2+3*y^2
20 FOR x=-4 TO 4
30    FOR y=-4 TO 4
40        PRINT x,y,f(x,y)
50    NEXT y
60 NEXT x
70 END
```

Parameters

Variables written in the parentheses succeeding DEF are called parameters.
Any variable which is written in other than the DEF line is a different variable, that is, even if it have the same name as that of a parameter, they have distinct locations on the memory.
Example.

```
10 DEF f(x)=x^3
20 LET x=4
30 PRINT f(2)
40 PRINT x
50 END
```

Variables x in line 10 and x in line 40 are different.
Thus the PRINT statement in line 40 outputs 4.

Variables in Common

Any variables other than parameters are common among the program unit.
Example.

```
10 DEF f(x)=a*x^2+b*x*c
20 INPUT a,b,c
30 PRINT f(1),f(2)
40 END
```

On the above program, variables a, b, c in line 10 and variables a, b, c in line 20 are the same, respectively.

Define a function with no parameter

A function that have no parameter is written without parentheses.
Example.

```
10 DEF spot=INT(6*RND+1)
20 FOR i=1 TO 100
```

```
30    PRINT spot
40 NEXT i
50 END
```

Note.
The result of a DEF statement has the precision of over 16 digits on the decimal operation mode. (not rounded to 15 digits as on a LET statement)

DEF Statement □□□

DEF statements enables us to write functions defined by complicated expressions simply as f(x), g(x,y) and so on.
Example.

```
10 DEF f(x,y)=x^2+3*y^2
20 FOR x=-4 TO 4
30    FOR y=-4 TO 4
40       PRINT x,y,f(x,y)
50    NEXT y
60 NEXT x
70 END
```

Parameters
Variables written in the parentheses succeeding DEF are called parameters.
Any variable which is written in other than the DEF line is a different variable, that is, even if it have the same name as that of a parameter, they have distinct locations on the memory.
Example.

```
10 DEF f(x)=x^3
20 LET x=4
30 PRINT f(2)
40 PRINT x
50 END
```

Variables x in line 10 and x in line 40 are different.
Thus the PRINT statement in line 40 outputs 4.

Variables in Common
Any variables other than parameters are common among the program unit.
Example.

```
10 DEF f(x)=a*x^2+b*x*c
20 INPUT a,b,c
30 PRINT f(1),f(2)
```

```
40 END
```
On the above program, variables a, b, c in line 10 and variables a, b, c in line 20 are the same, respectively.

Define a function with no parameter
A function that have no parameter is written without parentheses.
Example.
```
10 DEF spot=INT(6*RND+1)
20 FOR i=1 TO 100
30    PRINT spot
40 NEXT i
50 END
```

Note.
The result of a DEF statement has the precision of over 16 digits on the decimal operation mode. （not rounded to 15 digits as on a LET statement）

Note.
Functions can not be re-defined during execution as on some interpreter dialect.

■ FUNCTION ... END FUNCTION (internal functions)

Internal Function Definition

Although a DEF statement defines only a function that is represented by an expression, function definitions enables us to define a function by describing the process.An internal function definition is a function definition written within a program unit.All variables other than the parameters and data written in DATA statements are shared with the program unit.An internal function definition begins with FUNCTION line and ends with END FUNCTION Line.A FUNCTION line describes the name and the parameters.Parameters are written punctuated by commas and enclosed in parentheses if any.The rule for naming a function is the same as that of a variable.To determine the function value, use a LET statement where the function name is written on the left hand side of a equal sign.This type of LET statement is called a function - let - statement.Example.An internal function definition included by the main program.
```
10 FUNCTION P (n, r)
20 LET a = 1
30 FOR k = 1 TO r
40 LET a = a*(n - k + 1)
50 NEXT k
60 LET P = a
70 END FUNCTION
80 PRINT P (8, 3)
90 END
```

Note.A function - let - statement is not a LET - statement, and the function name is not a variable on the syntax of Full BASIC-.Thus the function value is not rounded to the precision of a variable.Note.A internal function definition must be written in before the line in which the function is referred.If the function definition is written in after the reference, write a DECLARE FUNCTION statement which holds the name of the function in the beginning.Example.100 DECLARE FUNCTION P
110 PRINT P (8, 3)
120 FUNCTION P (n, r)
130 LET a = 1
140 FOR k = 1 TO r
150 LET a = a*(n - k + 1)
160 NEXT k
170 LET P = a
180 END FUNCTION
190 END

■ EXTERNAL FUNCTION ... END FUNCTION (external functions)

External Function Definition

An external function definition begins with EXTERNAL FUNCTION line and ends with END FUNCTION line. An external function definition is written outside of the main program and of any other external procedures. This means that external functions are written below the END line and written not nested.
Example.
10 DECLARE EXTERNAL FUNCTION P
20 PRINT P(8,3)
30 END
100 EXTERNAL FUNCTION P(n,r)
110 LET a=1
120 FOR k=1 TO r
130 LET a=a*(n-k+1)
140 NEXT k
150 LET P=a
160 END FUNCTION
The portion of from the beginning to the END line, that is, form lines 10 to line 30, is the main program, and the rest is an external function definition.
The EXTERNAL FUNCTION line holds the name of the function and the parameters. Parameters are punctuated by commas and enclosed in parentheses if any. If a function has no parameters, parentheses are not written.
Use a LET statement as in the 150-line, to determine the value. Note that only the function name is written on the left hand side of an equal sign.

Any program unit that contains a reference to an external function other than itself holds a DECLARE EXTERNAL FUNCTION statement as in the 100-line, which notifies the compiler that the name is a function name.
An external function definition has no variables in common with any other program unit. For example, on the above example, even if the main program has variables named n, r, a or k, they are given no influence of invoking the function P.

Note.
If an external function has the same name as a supplied function, the supplied function is valid in the program unit that do not have a DECALARE EXTERNAL FUNCTION that declares the name is that of an external function.
Example.
If the 100-line is omitted in the following program, SQR in the 100 line becomes to mean the supplied function SQR.

```
100 DECLARE EXTERNAL FUNCTION SQR
110 PRINT SQR(4)
120 END
200 EXTERNAL FUNCTION SQR(x)
210 LET SQR=x^2
220 END FUNCTION
```

■ DECLARE EXTERNAL FUNCTION

DECLARE EXTERNAL FUNCTION

DECLARE EXTERNAL FUNCTION identifier, identifier, ・・・, identifier
designates that the identifiers succeeding DECLARE EXTERNAL FUNCTION are the names of external functions in the program unit where this statement exists.
BASIC gives the priority to the supplied functions in the program unit that lacks the designation by this statement.
Example.

```
10 DECLARE EXTERNAL FUNCTION INT
20 PRINT INT(3)
30 END
100 EXTERNAL FUNCTION INT(n)
110 LET INT=n^3-1
120 END FUNCTION
```

If the 10-line lacks in the above program, INT in the 20-line means the supplied function INT.

■ SUB ... END SUB (internal subprograms)

Internal Subprogram

A subprogram describes a definite procedure as an object.
An internal subprogram is a subprogram written within a program unit, begins with a SUB line and ends with a END-SUB line.
A SUB line describes its name and parameters. The parameters are written separated by commas and enclosed in parentheses if any.
The rule for naming a subprogram is the same as that for a numeric variable.
Use a CALL statement to execute a subprogram. To execute a subprogram that has parameters, write the arguments enclosed in parentheses as in function reference.
Example

```
100 OPTION ANGLE DEGREES
110 SUB circle(a,b,r)
120    FOR t=0 TO 360
130       PLOT LINES: a+r*COS(t),b+r*SIN(t);
140    NEXT t
150    PLOT LINES
160 END SUB
170 SET WINDOW -4,4,-4,4
180 CALL circle(1,-1,2)
190 CALL circle(-1,2,1)
200 END
```

All variables except parameters are common with the program unit.
When the control comes to the SUB-line, it jumps to the next line to the END-SUB line.
If a CALL statement is executed , the control jumps to the SUB-line, and if it reaches the END-SUB-line, it returns to the next line to the CALL-line.

■ DECLARE EXTERNAL SUB (external subprograms)

External Subprogram

An external subprogram is a subprogram that is written as a program unit, begins with EXTERNAL-SUB-line and ends with END-SUB-line.
Use a CALL statement to execute an external subprogram.
Write a DECLARE EXTERNAL SUB statement in the program unit that has a CALL statement that invokes the subprogram as in the 10-line of the following program.
Example.

```
10  DECLARE EXTERNAL SUB circle
20  SET WINDOW -4,4,-4,4
30  CALL circle(1,-1,2)
40  END
100 EXTERNAL SUB circle(a,b,r)
110 OPTION ANGLE DEGREES
120 FOR t=0 TO 360
130     PLOT LINES: a+r*COS(t),b+r*SIN(t);
140 NEXT t
150 PLOT LINES
160 END SUB
```

Note.
While the current version of Decimal BASIC allows DECLARE EXTERNAL SUB statements to be omitted, it is recommended that DECLARE EXTERNAL SUB statements should be written for compatibility with Full BASIC.

■ Internal and External Procedures

Internal Procedures and External Procedures

Internal function definitions including DEF-line, internal subprogram definitions, internal picture definitions and detached handlers (HANDLER ~ END HANDLER) are called **internal procedures**.
Besides, external function definitions, external subprograms and external picture definitions are called **external procedures**.
Furthermore, The main program and external procedures are called **program units**.

Internal procedures are written in a program unit, and share variables and the declaration such as option base or option angle with the program unit.
A Program unit is an individual object and does not share variables except parameters and the declarations with any other program unit.

Example.1 Internal Procedure

```
100  FUNCTION fact(n)
110    LET p=1
120    FOR i=1 TO n
130        LET p=i*p
140    NEXT i
150    LET fact=p
160  END FUNCTION
170  FOR i=1 TO 10
180     PRINT fact(i)
190  NEXT i
200  END
```

The internal function defined in lines 100 to 160 calculates the factorial of n.
But this program does not act as expected because when the FOR-loop in lines 120 to 140 is executed, the value of i, which is used on the FOR-loop in lines 170 to 190, is changed.

Example 2. External procedure

```
160 DECLARE EXTERNAL FUNCTION fact
170 FOR i=1 TO 10
180    PRINT fact(i)
190 NEXT i
200 END
300 EXTERNAL FUNCTION fact(n)
310 LET p=1
320 FOR i=1 TO n
330    LET p=i*p
340 NEXT i
350 LET fact=p
360 END FUNCTION
```

On the above program, as the variable i in lines 170 to 190 and the variable i in line 320 to 340 are different variables, the execution result shall be as expected.

Variable Paremeters (Pass by Reference)

Variable Parameter (Pass by Reference)

When a variable is passed to a subprogram, the parameter is made to point to the same location in the memory as that of the argument.
Therefore, if the argument passed to a subprogram is a variable, when the parameter is changed, the argument also changes.

Example.

```
100  DECLARE EXTERNAL SUB exchange
110  LET a=1
120  LET b=2
130  CALL exchange(a,b)
140  PRINT a,b
150  END
200  EXTERNAL SUB exchange(x,y)
210  LET t=x
220  LET x=y
230  LET y=t
240  END SUB
```

If you want the argument not to be changed, write it in parenthesis as follows.

```
130 CALL exchange((a),(b))
```

Note.
On function definitions, parameters are placed at different location of the arguments. Thus any function definition does not change arguments.

Array Parameters

Array parameter

Function definitions and subprograms can take arrays as parameters.

(1) Notation for Parameters
Array parameters are denoted by its name, parentheses and commas.
A() for 1-dimensional array A
A(,) for 2-dimensional array A
A(,,) for 3-dimensional array A
Example.
EXTERNAL SUB SAMPLE(A())

(2) Notation for Arguments
Array arguments are denoted by only array names.
Example.
CALL SAMPLE(A)

Example. Dot product of vectors
```
10 DEF DOT(a(),b())=a(1)*b(1)+a(2)*b(2)
20 DIM m(2),n(2)
30 MAT READ m
35 DATA 1,2
40 MAT READ n
45 DATA 3,4
50 PRINT DOT(m,n)
60 END
```

Use a UBOUND function to know the upper bound of indices of an array within a subprogram.

Example. A function that calculate the sum of all elements of an array.
```
100 DECLARE EXTERNAL FUNCTION ADD
110 DIM m(100)
120 MAT INPUT m(?)
130 PRINT ADD(m)
140 END
200 EXTERNAL FUNCTION ADD(a())
```

```
210 LET t=0
220 FOR i=1 TO UBOUND(a)
230    LET t=t+a(i)
240 NEXT i
250 LET ADD=t
260 END FUNCTION
```

Supplementary Explanation

When the 120-line is executed, enter an arbitrary but not greater than 100 number of numerics separated by commas. Then the upper bound of the array m becomes the number of numeric inputed.

Note.

When a function definition has an array parameter, the array shall be copied to the parameter.
Thus even if the parameter is changed, the argument does not change.
When a subprogram has an array parameter, the array itself is passed to the subprogram.
If the element of the array is changed, the alteration influences the argument.

■ Recursive Call

Recursive Call

Function definitions and subprograms are allowed to use themselves on their definition.
Example.
The factorial of n (n!) can be defined as follows.
1!=1
n!=n · (n-1)!
The following function FACT calculates factorials using the recursion above.

```
10 FUNCTION FACT(n)
20  IF n=1 THEN
30    LET FACT=1
40  ELSE
50     LET FACT=n*FACT(n-1)
60  END IF
70 END FUNCTION
80 PRINT FACT(10)
90 END
```

Supplementary Explanation.
The function FACT is exceuted with n=10 first.
Then to calculate n*FACT(n-1), FACT is executed with n=9 on the line 50.
And then to calculate n*FACT(n-1), FACT is executed with n=8 on the line 50 again.

This is repeated until n becomes 1.

Memory locations for all parameters of an internal function definition are newly gotten for each time they are invoked.
Memory locations for all variables of an external function definition are newly gotten for each time they are invoked.
Memory locations for all parameters except variable parameters of an internal subprogram are newly gotten for each time they are invoked.
Memory locations for all variables except variable parameters of an external subprogram are newly gotten for each time they are invoked.
Memeory locations newly gotten are removed when they return.

Note.
In an external procedure, its name need not be declared with a DECLARE EXTERNAL statement within itself.

■ EXIT FUNCTION, EXIT SUB, EXIT PICTURE

EXIT Statements

EXIT FUNCTION
terminates the execution of the function definition and returns.

EXIT SUB
terminates the execution of the subprogram and returns.

EXIT PICTURE
terminates the execution of the picture definition and returns.

■ Program Units

Program Unit

The main program and external procedures are called Program units.
Declarations by OPTION or DECLARE statements, the data sequence provided by DATA statements, variable names , channel numbers except zero and so on are independent for each program unit.
That is,
OPTION ANGLE, OPTION BASE must be written within each program unit, but each program unit can be set differently,
DECLARE EXTERNAL FUNCTION must be written within every program unit that uses the function,
and the data written in another program unit can not be read.

■ PICTURE, DRAW, SHIFT, SCALE, ROTATE, SHEAR

PICTURE Definition

A PICTURE definition is a variation of a subprogram.
A PICTURE definition can be transformed on invocation.
Use a DRAW statement to invoke a picture.
To define a picture, use EXTERNAL PICTURE, or, internal PICTURE definition syntax.
IF 'tulip' is defined as a picture,
DRAW tulip
executes the picture 'tulip'.

Transformation of a picture
DRAW Statements can be provided with a transformation.
Example.
DRAW tulip WITH SCALE(2)
executes 'tulip' with the coordinate multiplied by 2.
That is, 'tulip' is enlarged 2 times of center the origin.

Transform functions

SHIFT(a,b)	Moves horizontally a units, vertically b units
SCALE(a,b)	Enlarges horizontally a times , vertically b times
.SCALE(a)	Enlarges a times of center the origin
ROTATE(a)	Rotates by a of center the origin.
SHEAR(a)	Inclines horizontally by a. That is, (x, y) →(x + y tan a, y)

Composition of Transformations
Transform functions can be composed with *. Composition is made from left to right.
Example.
DRAW tulip WITH SHIFT(1,0) * SCALE(2)
Scaling shall be executed after translation.

[Note]
The beam is set off when entering and exiting of a picture definition.
Refer to PLOT LINES

■ EXTERNAL PICTURE Definition

External Picture Definition

An external picture definition is one of external procedures, and is functionally the same as an external subprogram.
An external picture definition begins with EXTERNAL-PICTURE-line and ends with END-PICTURE-line.
A DECLARE EXTERNAL PICTURE statement that specifies the name of an external picture is written in any program unit that uses it.

Example 1. An external picture that draws a regular n-gon.

```
10 DECLARE EXTERNAL PICTURE polygon
20 OPTION ANGLE DEGREES
30 SET WINDOW -4,4,-4,4
40 DRAW polygon(6)
50 DRAW polygon(7) WITH ROTATE(-90/7)*SCALE(1.5)*SHIFT(2,2)
60 END
100 EXTERNAL PICTURE polygon(n)
110 OPTION ANGLE DEGREES
120 FOR i=0 TO n
130    PLOT LINES:COS(i*360/n),SIN(i*360/n);
140 NEXT i
150 END PICTURE
```

Explanation
The part consisting of lines 10 to 60 is the main program,
and the part from lines 100 to 150 is an external picture definition.
As an external picture is a program unit, OPTION ANGLE statements are needed by both the main program and the external picture definition.

A feature of external picture definitions
All variables written within an external picture definition are independent of the variables within the main program .
For example, on the following program, variables i on the main program and on the external picture polygon are different.
Example 2.

```
10 DECLARE EXTERNAL PICTURE polygon
20 OPTION ANGLE DEGREES
30 SET WINDOW -4,4,-4,4
40 FOR i=-4 to 4
50    DRAW polygon(7) WITH SHIFT(i,i)
60 NEXT i
70 END
100 EXTERNAL PICTURE polygon(n)
110 OPTION ANGLE DEGREES
120 FOR i=0 TO n
130    PLOT LINES:COS(i*360/n),SIN(i*360/n);
140 NEXT i
150 END PICTURE
```

■ Internal PICTURE Definition

Internal Picture Definition

An internal picture definition is one of internal procedures, and is functionally the same as an internal subprogram.
An internal picture definition is written within a program unit.
An internal picture definition begins with a PICTURE line and ends with a END-PICTURE line.

Example 3. An internal picture

```
100 OPTION ANGLE DEGREES
110 SET WINDOW -4,4,-4,4
120 DRAW polygon(6)
130 DRAW polygon(7) WITH ROTATE(-90/7)*SCALE(1.5)*SHIFT(2,2)
140 PICTURE polygon(n)
150    FOR i=0 TO n
160       PLOT LINES:COS(i*360/n),SIN(i*360/n);
170    NEXT i
180 END PICTURE
190 END
```

Explanation
The main program consists of the whole lines.
Among these lines, lines 140 to 180 make up an internal picture.

A feature of internal picture definitions
All variables except parameters are shared with the program unit.
For example, on the following program, variable i in lines 120 to 140 and variable i in lines 170 to 190 are identical. Therefore, Because the variable i changes when the DRAW statement in 130-line is executed, the FOR ~ NEXT loop in lines 120 to 140 does not work correctly.
Example 4

```
100 OPTION ANGLE DEGREES
110 SET WINDOW -4,4,-4,4
120 FOR i=-4 to 4
130    DRAW polygon(7) WITH SHIFT(i,i)
140 NEXT i
150 PICTURE polygon(n)
170     FOR i=0 TO n
180        PLOT LINES:COS(i*360/n),SIN(i*360/n);
190     NEXT i
200 END PICTURE
210 END
```

■ Transformation with Arrays

Transformation with arrays

Transformation can be assigned with a 4×4 matrix instead of transform functions.

LET A be a 4×4 matrix defined by DIM A(4,4).
If (x y 0 1)A = (x' y' z' c),
a point (x, y) is transformed to (x' / c, y' / c).

Especially, if the 4-th column is 0, 0, 0, 1,
DRAW a_pict WITH A
makes x, y transformed
A(1,1)*x + A(2,1)*y + A(4,1)
A(1,2)*x + A(2,2)*y + A(4,2)
respectively.

Example

```
100 DECLARE EXTERNAL PICTURE a_pict
110 DIM A(4,4)
120 MAT READ A
130 DATA   1,   2,  0,  0
140 DATA 0.8, 0.5,  0,  0
150 DATA   0,   0,  1,  0
160 DATA   1,  -2,  0,  1
170 SET WINDOW -4,4,-4,4
180 DRAW a_pict WITH A
190 END
200 EXTERNAL PICTURE a_pict
210 PLOT AREA : 0,0; 1,0; 1,1; 0,0
220 END PICTURE
```

Transform functions and 4×4 matrices can be mixed.

Example

```
180 DRAW a_pict WITH SHIFT(-1,1)*a
```

Composition of transformations is identical with multiplication of matrices.
Thus composition of transformation can be obtained by multiplication of matrices.

Example. A transformation of A composed n times.

```
100 DECLARE EXTERNAL PICTURE circle
110 DIM A(4,4),B(4,4)
120 MAT READ A
130 DATA   1,   2,  0,  0
140 DATA 0.8, 0.5,  0,  0
150 DATA   0,   0,  1,  0
160 DATA   1,  -2,  0,  1
170 MAT B=IDN     !identity
180 INPUT n
190 FOR i=1 TO n
200    MAT B=B*A
210 NEXT i
220 SET WINDOW -10,10,-10,10
230 DRAW circle WITH B
240 END
900 EXTERNAL PICTURE circle
910 FOR t=0 TO 2*pi STEP pi/180
920    PLOT LINES:cos(t),sin(t);
930 NEXT t
940 END PICTURE
```

■ TRANSFORM Assignment

Transform assignment

An arbitrary multiplication of 4×4 matrices or transform functions can be written on the right side of a MAT statement.
Example.

```
DIM a(4,4)
MAT a=SHIFT(1,0)*SHEAR(pi/6)*SCALE(0.8)
```

Inside of a picture definition, the reserved keyword TRANSFORM means the current transform matrix.
EXAMPLE

```
10 DIM a(4,4)
20 DRAW a_pict WITH ROTATE(PI/6)
30 PICTURE a_pict
40    MAT A=TRANSFORM
50    MAT PRINT A
60 END PICTURE
70 END
```

■ Modules

Modules

The mechanism of modules enables sharing variables among some program units.

1. A Module contains public variables, and share variables.
Public variables are shared with the whole module and the program units where the external declaration for that variable is written.
(Note that a program unit is an external procedure or the main program.)
Share variables are shared with the whole module.

2. Example. (Share variables)
On the following program, the numeric variables 'index' and 'stack' defined in the 2030-line are shared with the module 'stack'.

```
1000 DECLARE EXTERNAL SUB stack.push
1010 DECLARE EXTERNAL FUNCTION stack.pop
1020 CALL push(12)
1030 PRINT pop
1040 END
1050 !
2000 MODULE stack
2010 PUBLIC SUB push
2020 PUBLIC FUNCTION pop
2030 SHARE NUMERIC index, stack(100)
2040 LET index=0
2100 EXTERNAL SUB push(a)
2110    LET index=index+1
2120    LET stack(index)=a
2130 END SUB
2200 EXTERNAL FUNCTION pop
2210    LET pop=stack(index)
2220    LET index=index-1
2230 END FUNCTION
2240 END MODULE
```

3. A module consists of the module-body and external procedures.
The module-body can contain public declarations and share declarations.
The module-body shall be executed before the main program.

PUBLIC declarations declares the variables or procedures shared with the program.
PUBLIC NUMERIC numeric_variable, numeric_variable , ...
PUBLIC STRING string_variable, string_variable , ...
PUBLIC FUNCTION function_name, function_name, ...
PUBLIC SUB subprogram_name, subprogram_name, ...
PUBLIC PICTURE picture_name, picture_name, ...

SHARE declarations declare the variables shared within the module.
SHARE NUMERIC numeric_variable, numeric_variable , ...
SHARE STRING string_variable, string_variable , ...
SHARE FUNCTION function_name, function_name, ...
SHARE CHANNEL channel-number, channel_number, ...
(channel_numbers are as #1, #2 and so on.)

For an array declaration, the bounds-declaration is included.
Example.

```
MODULE m1
PUBLIC NUMERIC a(2,2), b
PUBLIC STRING s$(4)
SHARE NUMERIC c(10)
LET b=pi/4
   ……
END MODULE
```

4. Module-option statement can be written in the module-body.
EXAMPLE.

```
MODULE OPTION ANGLE DEGREES
MODULE OPTION ARITHMETIC NATIVE
```

Any module option statement is valid through the whole module.
[note]
(1) OPTION declaration written in a module-body is valid only through the module-body.
(2) Neglecting the standard, MODULE OPTION statements and OPTION statements must be written before any PUBLIC statement or any SHARE statement.
(3) The module-body of a module cannot contain any internal procedure.

5. A Program unit that has references to public-declared procedures or variables must contain the DECLAE EXTERNAL declarations for those references.
DECLARE EXTERNAL NUMERIC module_name.numeric_variable, module_name.numeric_variable, ...
DECLARE EXTERNAL STRING module_name.string_variable, module_name.string_variable, ...

DECLARE EXTERNAL FUNCTION module_name.function_name, module_name.function_name ,…
DECLARE EXTERNAL SUB module_name.subprogram_name, module_name.subprogram_name, ...
DECLARE EXTERNAL PICTURE module_name.picture_name, module_name.picture_name, ...

If an arrays is specified, parenthesis and commas of number of dimension minus 1 follows the array name.
On the program unit that declares external procedures or variables, the procedures or variables can be written with or without module name part.

Example

```
DECLARE EXTERNAL NUMERIC m1.a(,), m1.b
DECLARE EXTERNAL STRING m1.s$()
FOR i=1 TO UBOUND(s$)
  PRINT s$(i)
NEXT i
PRINT m1.a(1,1), m1.b
……
END
```

[disclaimer]
On This version, external subprograms and external pictures are declared public implicitly.
Any SHARE declaration for external subprograms or external pictures is ignored.

For more information about modules, refer to ANSI X3.113a-1989

■ PUBLIC statement (original enhancement)

PUBLIC sentence extended

As an extension of its own, the main program PUBLIC NUMERIC, PUBLIC STRING allow you to write.
The main program, shall be deemed to constitute a single module with an external procedure definition does not belong to any module.
Example

```
10 PUBLIC NUMERIC a (20)
20 CALL s1
30 PRINT a (4)
40 END
50 EXTERNAL SUB s1
60 LET a (4) = 12
70 END SUB
```

PROGRAM and writing a sentence, to refer to global variables declared in the main program
Point variable name program name
Forms may be used.
Example

```
100 PROGRAM pro1
110 PUBLIC NUMERIC a
120 DECLARE EXTERNAL SUB m1.s1
130 CALL s1
140 PRINT a
150 END
200 MODULE m1
210 PUBLIC NUMERIC a
220 EXTERNAL SUB s1
230 DECLARE EXTERNAL NUMERIC pro1.a
240 LET pro1.a = 12
250 LET a = 18
260 END SUB
270 END MODULE
```

<Note>
250-line program on global variables declared in the statement let the line 210 a means to assign.

- LOCAL (original enhancement)

LOCAL (original enhancement)

LOCAL variable, variable, · · ·

A LOCAL statement is written in an internal procedure, and it declares its local variables, numeric or string, to enable recursive call of the procedure.

Note.
Because LOCAL is a non-standard enhancement, it is recommended that global variables should be declared as a public variables in a module and local variables should be those of external procedures within the module.
However, it is unavoidable to use a LOCAL statement if an internal procedure in an external procedure that performs recurrence needs local variables.
In such a case, check Option-Syntax "Compel All Variables Declared" to confirm no variable in the internal procedure not to be declared.

■ MERGE (original enhancement)

MERGE (original enhancement)

MERGE file_name
file_name is a string constant.

A MERGE statement can be written outside the program units and the modules.
The text included in the file are inserted on compiling, and removed after the execution.
The name of the file that contains the text that is to be supplemented is described in the MERGE statement enclosed in quotation marks.
The file is searched first in the current directory, second UserLib subdirectory of the directory BASIC was launched, and last Library subdirectory.
When you want to appoint multiple files, write a MERGE statement for each file.
The text that is to be merged can contain MERGE statements.

Significant notice.
If the number of lines below the MERGE-line changes by automatic correction , deletion of the text after the execution shall not be done correctly. Thus, it is recommended that MERGE statements should be written in the tail of a program.

OLE, ActiveX (original enhancement)

Using OLE Automation (proprietary extension)

BASIC does this, OLE objects will be used for Full BASIC module.
Must meet the following criteria, ActiveX and OCX also available.
Hidden (does not need the parent window)
Which corresponds to the dispatch interface

OLE objects assigned to the module

The module may be any name.
OLE object modules There are two ways to assign.
Typically, the module itself ProgID (or ClassID string representation of) something that takes a quoted sentence to write OLE CretaeObject. This statement, OLE object creation means that, in fact, be executed during translation.
ProgID cases specified by
OLE CreateObject "Word.BASIC"
Cases specified by ClassID
OLE CreateObject "{000209FE-0000-0000-C000-000000000046}"

Another method, OLE objects to use when assigning a module that exists as an object property (see below).

OLE method assigned to an external procedure

Method is assigned to an external procedure module.
Method that returns a value assigned to the external function, external function end function between the lines and lines to write OLE METHOD statement. If the type is a string as a function of the result is a string or numeric value when the result type of BOOL is defined as a function of the number (when the function string $ function name at the end of the addition.)
A method that returns a value assigned to the external subprogram, external sub and end sub lines between the lines to write OLE METHOD statement.
OLE METHOD argument statement, write the method name enclosed in quotation marks.
Number and type of procedure arguments (by the numbers and strings) are assigned to match that of the method.
The method name to check the translation. Argument type and number of errors are detected at run time.

Example
If MS-Word is installed, the following executable program.
280 lines of line modules 160 Word from the OLE objects that incorporate Word.BASIC. Word.BASIC ProgID that are registered in Windows registry.

As shown in line 170 from outside the module to use external procedures that are defined in the module written PUBLIC statements. (For external functions, PUBLIC SUB PUBLIC FUNCTION statement instead of writing a sentence)
Program unit using the external subprogram defined in the module, as shown in line 100 DECLARE EXTERNAL written statement. DECLARE EXTERNAL external procedures defined in the statement, adding it to write the module name may be omitted to write the module name. (Written and added to the module name, line 110 CALL Word.FileNew ("") as a) If you use an external function in a manner similar to writing a line 100 DECLARE EXTERNAL FUNCTION statement.

```
100 DECLARE EXTERNAL SUB Word.Filenew, Word.Insert, Word.FileSaveAs
110 CALL FileNew ("")
120 CALL Insert ("The string is inserted into the document.")
130 CALL FileSaveAs ("A: \ TEST01.DOC")
140 END
150
160 MODULE Word
170 PUBLIC SUB FileNew, Insert, FileSaveAS
180 OLE CREATEOBJECT "Word.Basic"
190 EXTERNAL SUB FileNew (s $)
200 OLE METHOD "FileNew"
210 END SUB
220 EXTERNAL SUB Insert (s $)
230 OLE METHOD "Insert"
240 END SUB
250 EXTERNAL SUB FileSaveAs (s $)
260 OLE METHOD "FileSaveAs"
270 END SUB
280 END MODULE
```

OLE property assignment to an external procedure

Property is set to assign external subprogram.
external sub end sub line and the line between writing a sentence OLE PROPERTYPUT. The argument, write the property name enclosed in quotation marks. Arguments should be set to a value of subprograms.
If a property of type array, index as an argument, writing a value to be set to last argument.

Read the property is assigned to an external function.
external function end function between the rows and rows OLE PROPERTYGET written statement. The argument, write the property name enclosed in quotation marks.
If a property of type array, the index is a function argument.

Available data types

Available data types are limited to a number or string. Complex numbers and arrays are not supported.
Boolean numbers are part of Microsoft, if you assign a number to use. OLE provides a true 1, false is defined as 0.

Assigned to the module exists as an object property

OLE CREATEOBJECT OLE ASSIGN statement instead of statement you run in the module, an object can be assigned a number which refers to the module object. This number is obtained by reading the properties of the parent object.
When reading the property, specifying the object-valued properties, the result is a number which refers to that object. This value must be a single OLE ASSIGN statement must be assigned. (Or just stop by to receive the value, assign values to variables obtained by reading a single property, with multiple OLE ASSIGN statements that do not pass)
OLE ASSIGN statement, OLE CREATEOBJECT unlike statements are executable statements. OLE ASSIGN statement in the module object is assigned, the method name, the name of the property inspections will be suspended until the procedure is called initially assigned to them.

Example
For this example, Word2002 can perform when installed (on other versions of Word have not been tested.)

```
100 DECLARE EXTERNAL FUNCTION Word.Documents
110 DECLARE EXTERNAL FUNCTION Word.Caption $
120 DECLARE EXTERNAL SUB Word.SetVisiBle
130 DECLARE EXTERNAL SUB Documents.SetDocuments, Documents.Open
140 DECLARE EXTERNAL SUB Selection.SetSelection, Selection.EndKey
150 CALL SetDocuments (Word.Documents)
160 CALL Open ("C: \ BASICw32 \ README.TXT")
170 CALL SetSelection (Word.Selection)
180 CALL SetVisible (-1)
190 WAIT DELAY 10
200 PRINT Caption $
210 CALL EndKey
220 END
230
240 MODULE Word
250 OLE CREATEOBJECT "Word.Application"
260 PUBLIC FUNCTION Selection, Caption $, Documents
270 PUBLIC SUB SetVisible
280 EXTERNAL FUNCTION Selection
290 OLE PROPERTYGET "Selection"
300 END FUNCTION
```

```
310 EXTERNAL FUNCTION Caption $
320 OLE PropertyGet "Caption"
330 END FUNCTION
340 EXTERNAL SUB SetVisible (n)
350 OLE PropertyPut "Visible"
360 END SUB
370 EXTERNAL FUNCTION Documents
380 OLE PropertyGet "Documents"
390 END FUNCTION
400 END MODULE
410
420 MODULE Documents
430 PUBLIC SUB Open, setDocuments
440 EXTERNAL SUB setDocuments (n)
450 OLE ASSIGN n
460 END SUB
470 EXTERNAL SUB OPEN (s $)
480 OLE METHOD "Open"
490 END SUB
500 END MODULE
510
520 MODULE Selection
530 PUBLIC SUB SetSelection, EndKey
540 EXTERNAL SUB SetSelection (n)
550 OLE ASSIGN n
560 END SUB
570 EXTERNAL SUB EndKey
580 OLE METHOD "EndKey"
590 END SUB
600 END MODULE
```

(Note)
When the method returns a value assigned to a function, written as a rewrite of the values of variables as arguments to that method, the result does not reflect the BASIC variables.
Because this BASIC immediately remove the object has finished executing the program works if you use asynchronous OCX, OCX to terminate the program execution until after the operation.

Event processing

When the event processing, OLE CREATEOBJECT argument, ProgID (or ClassID string representation) followed by those enclosed in quotation marks, separated by a comma, the event interface GUID string representation in quotes Write something enclosed.
Written in the form of one of the following cases.
OLE CREATEOBJECT "xxxxx.xxxx", "{1234567A-1234-1234-89AB-0123456789AB}"
OLE CREATEOBJECT "{12345678-1234-1234-89AB-0123456789AB }","{ 1234567A-1234-1234-89AB-0123456789AB}"

OCX module written in the external subprogram is called to handle events from the external sub at the end of the line, separated by commas "DispID" DispID its event number that indicates the constant writing.
Cases EXTERNAL SUB OnEvnet (s $), DispID 2

<Note> This is BASIC, so may not be called from different threads in an asynchronous process. If you call OCX is an asynchronous event processing, BASIC original thread WAIT or PAUSE statement to hibernate we must effectively run the statement.

<Note> OCX called from a procedure performed in STOP statement does not mean the end of the BASIC program. When the STOP statement, BASIC is to suspend execution of the procedure should be generated saying that the OCX exception. OCX is a subsequent operation as soon as it is. Even if you choose to stop the debug window.

Signal

Signals can be used to inform the end of the process is called asynchronous event handling procedure.
Signals are identified by names beginning with letters (unquoted in quotes.)

SIGNAL Signal Name
Signaling that a signal is identified by name.
WAIT SIGNAL Signal Name
Wait until the signal is emitted signal is identified by name.
Numeric expression TIMEOUT WAIT SIGNAL Signal names
Time specified by the numeric expression (s) or expires, or wait until it receives a signal.
0.01 minimum numbers may be specified up to 429,496,729.5
TIMEOUT is specified, Ctrl-B is also waiting for the press not be interrupted.

■ Windows, API and external DLLs (ASSIGN DLL)

DLL function call (proprietary extension)

DLL functions and sub-programs can be defined using. Internal or external may be (DEF sentence is not applicable.)
Used to define a DLL function (or subprogram) provides, FUNCTION line (SUB or rows), END FUNCTION line (END SUB or row) of the form during the writing ASSIGN statement.

ASSIGN DLL file name, function
DLL file name, function, write a string constant.

Example
FUNCTION GetVersion
ASSIGN "kernel32.dll", "GetVersion"
END FUNCTION
DLL file name, including trying to use a function that specifies the name of the DLL. The function name, DLL to specify the name has been registered in. Function names that identify the differences between small and large aware of the letter.

Numeric arguments to pass an assessment significant digits truncated 32-bit integer. Even if the variable arguments to a subprogram is not, as is always the argument.
String argument is a pointer to the first character (32 bits) is passed as. However, when a pointer to an empty string is not null, pass null. BASIC strings can be used as a null-terminated string.
Arguments are pushed onto the stack starting at the end of the argument. DLL functions, arguments must be removed before returning control to the stack.
When you define a numeric functions, DLL functions when the control is returned from the function value and the value of the EAX register. This value is interpreted as always 32-bit signed integer.
If it's a string function, subject to DLL functions are null-terminated ASCII (or Shift-JIS) should be a function returns a pointer to a string.

1 cases
```
DECLARE EXTERNAL FUNCTION MesBox
LET n = MesBox (0, "Hello", "BASIC", 3)
PRINT n
END
EXTERNAL FUNCTION MesBox (owner, text $, caption $, flag)
ASSIGN "user32.dll", "MessageBoxA"
END FUNCTION
```

Requesting the address of a variable as an argument when calling a DLL function, the argument can be a string variable. REPEAT $ functions, etc. to use, the number of characters in string variables as arguments to the DLL you need more than a function of size. The value of this variable which was created by assigning the other variables as well, and this variable should not even those who have assigned to other variables. The value received is used to eject a given substring. Programs using this variable in units OPTION CHARACTER BYTE Once you have declared, in bytes, the buffer can be used as a string variable.
2 cases
```
DECLARE EXTERNAL FUNCTION CurrDir $
PRINT CurrDir $
```

```
END
EXTERNAL FUNCTION CurrDir $
OPTION CHARACTER BYTE
FUNCTION GetCurrentDirectory (n, s $)
ASSIGN "kernel32.dll", "GetCurrentDirectoryA"
END FUNCTION
LET s $ = Repeat $ ("", 200)
LET n = GetCurrentDirectory (200, s $)
LET CurrDir $ = s $ (1: n)
END FUNCTION
```

Take real values when using DLL function, write a statement of the form: ASSIGN.
ASSIGN DLL file name, function, FPU
This form of statement and ASSIGN, FPU to Sutakkutoppurejisuta of the POP, the value of the function and its value.

<Supplement> DLL receives an exception from Windows, ASSIGN statement to generate exceptions Extype =- 9900. However, DLL exceptions that may occur in the DLL should be treated with internally.
<Supplement> ASSIGN statement does not save the FPU flags, DLL when you change the flag in the FPU must be restored before returning.
<Reference> Code of those arguments are passed (the caller is stacked on the stack in reverse order argument on the release of the arguments made by the called the stack) is the same as those of the Windows API.
<Note> DLL is loaded at compile time. Other specified DLL DLL DLL Tei Sono link to get an error because even if you can not load the specified DLL programs to display the name of grammatical errors.
<Note> Wrong number of arguments are not checked at run-time translation as well. Incorrect number of arguments leads to disastrous results.

<Reference> 32 n-bit signed integer function to convert hexadecimal string to an unsigned integer
BSTR $ (MOD (n, 4294967296), 16).

<Reference> ORD function CHR $ function, given substring

■ CallBback function (original enhancement)

Function CallBack (original extended)

CallBack function, BASIC is a procedure called from the Win32 API from a DLL that can be invoked by a procedure call.
CallBack function can be used up to 10, they are identified by numbers 0 through 9.

CallBack function specification

As a function of the procedures for using CallBack FUNCTION or SUB format following lines at the end of the line that specifies the identification number of the CallBack function. Identification number must be constant to 9-0.
FUNCTION function name (argument list), CALLBACK identification number
SUB subprogram name (argument list), CALLBACK identification number
CallBack procedure to assign an external function may be internally.
All the argument is an argument. Pointer to a null-terminated string can receive a string variable.
Otherwise, the argument must be assigned a numeric variable names for all 32-bit signed integer.
(A pointer to a string variable receives a pointer to a non-null-terminated string must be.)
Argument with a magnitude greater than 32-bit number to receive a variable divided into 32-bit.
All arguments of the callback function (including padding), and the number of bytes must match four times the number of arguments written in the argument list.
Boolean variable that receives a number. May be interpreted as a true non-zero.
Result of a Boolean function (return) when required as the 1 for a true (or -1), should I usually returns 0 if false.

Get the address of the function CallBack

CallBackAdr CallBack function to get the address of the function. Argument, CallBack function identification number.
Do not specify internal procedures in different program units.

Example

```
100 OPTION CHARACTER BYTE
110 LET Name $ = REPEAT $ (CHR $ (0), 255
120 CALL EnumWindows (CallBackAdr (9), 0)
130 SUB EnumWindows (IpEnumFunc, IPalam)
140 ASSIGN "user32.dll", "EnumWindows"
150 END SUB
160 FUNCTION GetWindowText (hWnd, IpString $, cch)
170 ASSIGN "user32.dll", "GetWindowTextA"
180 END FUNCTION
190 FUNCTION Enum (Handle), CALLBACK 9
200 IF GetWindowText (Handle, Name $, LEN (Name $))<> 0 THEN
210 PRINT NAME $ (1: POS (NAME $, CHR $ (0)) -1)
220 END IF
230 LET Enum = -1
240 END FUNCTION
250 END
```

Notes
CallBack to use asynchronous function calls should not be.
Incorrect number of arguments are not checked at run-time translation as well. Incorrect number of arguments, there is serious potential damage to computer systems.

■ PackDBL$m UnPackDBL, WinHANDLE functions

PackDBL $ function, UnpackDBL function, WinHandle function (original extended)

Built-in Functions
PACKDBL $ (numeric expression)
Memory image of a numeric expression whose value doubles (8 bytes) is returned.

UNPACKDBL (string expression)
Memory image of doubles (8 bytes) is converted to a numeric value of BASIC.

WINHANDLE (string expression)
Win32API to pass, BASIC own window system (or control) of the handle.
Argument, "MAIN", "TEXT", "GRAPHICS", "INPUT", "CHARACTER INPUT", "TRACE", "LOCATE", "TEXTWINDOW1", "TEXTWINDOW2", "RICHEDIT", "RICHEDIT1", "RICHEDIT2 "either.
Ignore case differences.
"RICHEDIT", "RICHEDIT1", "RICHEDIT2" the text window, TextWindow1, TextWindow2 the RichEdit control.

Example 1
```
100 DECLARE EXTERNAL SUB MoveTextWindow, ResizeTextWindow
110 FOR i = 1 TO 5
120 PRINT "Hello"
130 CALL MoveTextWindow (i * 100, i * 60)
140 CALL ResizeTextWindow (320,160 + i * 20)
150 WAIT DELAY 1
160 NEXT i
170 END
180 EXTERNAL SUB MoveTextWindow (x, y)
190 SUB SetWindowPos (hwnd, HwndInsAfter, x, y, cx, cy, nFlags)
200 ASSIGN "user32.dll", "SetWindowPos"
210 END SUB
220 CALL SetWindowPos (WinHandle ("TEXT"), 0, x, y, 0,0,1)
230 END SUB
240 EXTERNAL SUB ResizeTextWindow (x, y)
```

```
250 SUB SetWindowPos (hwnd, HwndInsAfter, x, y, cx, cy, nFlags)
260 ASSIGN "user32.dll", "SetWindowPos"
270 END SUB
280 CALL SetWindowPos (WinHandle ("TEXT"), 0,0,0, x, y, 2)
290 END SUB
```

Example 2

```
FUNCTION SendMessage (Handle, MSG, WParam, IParam)
  ASSIGN "user32.dll", "SendMessageA"
END FUNCTION
FOR i = 1 TO 100
  PRINT i
NEXT i
LET EM_SCROLL = BVAL ("B5", 16)
LET SB_LINEUP = 0
LET SB_LINEDOWN = 1
LET SB_PAGEUP = 2
LET SB_PAGEDOWN = 3
FOR i = 1 TO 100
  WAIT DELAY 0.02
  LET a = SendMessage (WINHANDLE ("RICHEDIT"), EM_SCROLL, SB_LINEUP, 0)
NEXT i
END
```

Files

Overview

Text Files - An Overview

A BASIC program manages file input or output using virtual channels, which are connected with actual files.
The process necessary to use a file is
1) to assign the file to a channel (to say, open the file),
2) to do input or output,
3) to release the assignment of the file to the channel (to say, close the file).
Channels are identified with positive integers.
The channel designated by an integer n is denoted by #n.

Example 1. File Output.
If the file already exists, all the contents of it is erased before the output.
This program outputs also commas so that a program example 2 can read the file.

```
LET F$ = "A:ABC.TXT"       ! Assign the name of a file to a string variable F$.
OPEN #1: NAME F$           ! Assign the file "A:ABC.TXT" to a channel #1.
ERASE #1                   ! Erase the file contents if any.
FOR x=0 TO 10
  PRINT #1 : x; ","; SQR(x)   ! write to the channel #1.
NEXT x
CLOSE #1                   ! Release the assignment to the channel
END
```

Example 2.
A program that reads data from the file written by the above program.

```
LET F$="A:ABC.TXT"
OPEN #2: NAME F$
FOR i=1 TO 10
  INPUT #2: a, b
  PRINT a, b
NEXT i
CLOSE #2
END
```

Example 3.
A program that reads every line of a file.

```
LET F$="A:ABC.TXT"
OPEN #2: NAME F$
DO
  LINE INPUT #2, IF MISSING THEN EXIT DO: a$
  PRINT a$
LOOP
CLOSE #2
END
```

A LINE-INPUT statement read a line from the channel and assign it to the string variable.
IF-MISSING-THEN clause indicates the action that is executed when the file data is exhausted.

- **OPEN #**

OPEN #

An OPEN statement assigns a file to a channel.

Channel_number is an numeric expression, and File_Name is a string expression below.

When a file name without an extension is assigned, if the file designated does not exist, the file name is assumed to have the extension ".TXT".

OPEN #Channel_Number : NAME File_Name ,ACCESS OUTIN
OPEN #Channel_Number : NAME File_Name
Example. OPEN #1: NAME "A:ABC.TXT"
When the designated file does not exist, the file shall be newly created.
However if the drive name or the directory name is wrong, an exception of extype 7101 shall be raised.
Just after the execution of an OPEN statement, the file pointer points the beginning of the file.
In case of output using PRINT statements, it is necessary for beginning output either to erase the contents using a ERASE statement or to move the file pointer to the end using a SET POINTER END statement.

OPEN #Channel_Number : NAME File_Name ,ACCESS INPUT
opens the file only for input.
If the designated file does not exist, an exception of extype 7102 shall be raised.
When a ERASE statement or a PRINT statement is executed, an exception shall be raised (extype7301,7302) .

OPEN #Channel_number : NAME File_Name ,ACCESS OUTPUT
opens the file only for output.
The file pointer points the end.
If the file designated does not exist, the file shall be created.
If a ERASE statement or a INPUT statement is executed, an exception shall be raised(extype7301,7303).
If SET POINTER BEGIN has been executed, an exception may raised when a PRINT statement is executed.
Thus, an OPEN-ACCESS-OUTPUT statement is used for appending output.

Note.
Channel numbers greater than 0 are independent for each program unit.
A Channel can be made a parameter of an external subprogram.
Example.

```
100 DECLARE EXTERNAL SUB s1
110 OPEN #1:NAME "A:TEST1.TXT"
120 CALL s1(#1,"ABC")
130 CLOSE #1
```

```
140 END
200 EXTERNAL SUB s1(#2,s$)
210 PRINT #2:s$
220 END SUB
```

Note.
The console is assigned to #0 beforehand. That is, input or output for #0 are the same as those written omitted a channel expression. No OPEN statement is needed for #0.

Refer to COM-port (serial port)

■ INPUT

INPUT #

INPUT #Channel_Number : variable, variable, … , variable
reads data from the file and assigns them to the variables.
Data must be separated by commas when the statement has more than one variable.
Regularly, execution of an INPUT statement reads a line from the file, but if a line ends with a comma, it is assumed to be continued to the next line.

LINE INPUT #Channel_Number : string_variable
reads a line from the file and assigns it to the variable.

INPUT #Channel_Number : variable, variable, … , variable, SKIP REST
If the line has too much data, excessive data is ignored.

INPUT #Channel_Number , IF MISSING THEN EXIT DO : variable, variable, … , variable
LINE INPUT #Channel_Number , IF MISSING THEN EXIT DO : string_variable
If the file pointer exceeds the end of the file, EXIT DO is executed.
EXIT FOR or a line number can be written instead of EXIT DO.

LINE INPUT #Channel_Number , TIMEOUT numeric_expression : string_variable
If no input reply is made within the specified seconds, an exception of extype 8401 is raised.

MAT INPUT #Channel_Number : array , array , … , array
All data must be written separated by commas, even if more than one array is specified.
If a line ends with a comma, this line is assumed to be continued to the next line.
2 or 3-dimensional arrays read data in the order of latter index changing fast.

let A be a 1-dimensional array.
MAT INPUT #Channel_Number : A(?)
reads data separated by commas. The upper bound may change according to the number of data.

Refer to
INPUT
INPUT TIMEOUT
INPUT ELAPSED
LINE INPUT
MAT INPUT

■ CHARACTER INPUT

CHARACTER INPUT #

CHARACTER INPUT #Channel_Number : string_variable
reads a character from the file.
The character set is the Microsoft Windows default character set.
if a character consists of 2 bytes, 2 bytes may be read once.
Refer to Option Compatibility

If you want to read a byte always, write
OPTION CHARACTER BYTE
in the beginning of the program unit

The following forms of CHARACTER INPUT statements may be useful for COM-ports.
If more than one clause is included, they are written puctuated by commas.

CHARACTER INPUT #Channel_Number ,CLEAR : string_variable
erases the buffer before input, if the channel has an input buffer.

CHARACTER INPUT #Channel_Number ,NOWAIT : string_variable
does not wait for input. string_variable remains the former if no character is input.

CHARACTER INPUT #Channel_Number ,TIMEOUT numeric_expression : string_variable
raises an exception of extype 8401 if no input relay is given within the specified seconds.

CHARACTER INPUT #Channel_Number ,ELAPSED numeric_variable : string_variable

The time (in soconds) that elapsed until the input reply is given is assigned to numeric_variable.

ASK #Channel_Number : CHARACTER PENDING numeric_variable
The number of characters (in bytes) staying on the buffer is assigned to the variable.

ASK #Channel_Number : TYPEAHEAD string_variable
tests if the channel have an input buffer. The result is one of "YES" , "NO" or "UNKNOWN".

Refer to CHARACTER INPUT

Note.
Decimal BASIC does not defines the end-of-line character for CHARACTER INPUT. Therefore, if the program unit contains OPTION CHARACTER BYTE, byte access for a binary file is possible.

■ PRINT

PRINT #

PRINT #Channel_Number : items
PRINT #Channel_Number
PRINT #Channel_Number ,USING format_string : items
writes items into the file.
If output is performed at the location of the file except the end, an exception shall be raised.
Regularly, items are written separated by commas or semicolons as in an ordinary PRINT statement.
A comma or a semicolon also can be written on the tail.
When no item is written, no colon is written just after the channel_number.

MAT PRINT #Channel_Number : array
MAT PRINT statements are available. Refer to MAT PRINT

Refer to ERASE, SET POINTER END

Note.
INPUT statements do not accept the output of PRINT statements.
Refer to Internal Files

■ ERASE

ERASE #

ERASE #Channel_Number
erases the content of the file. (does not erase the file itself.)
Even if the file is empty or created newly, no exception shall be raised.
If this statement is executed for the file opened with ACCESS INPUT or ACCESS OUTPUT specified, an exception shall be raised.
This statement is useful when the file may already exist.

Example.

```
10 OPEN #1:NAME "A:ABC.TXT"
20 ERASE #1:
30 PRINT #1: DATE$,TIME$
40 CLOSE #1
50 END
```

■ SET # POINTER

SET # POINTER

1. SET #Channel_Number : POINTER BEGIN
moves the file pointer to the beginning.
Note that it is not allowed to overwrite the existing records.

2. SET #Channel_number : POINTER END
moves the file pointer to the end.
Use this to append records to the existing records.
Even if the file is empty, no exception shall be raised.
Example.

```
10 OPEN #1:NAME "A:ABC.TXT"
20 SET #1: POINTER END
30 PRINT #1: DATE$,TIME$
40 CLOSE #1
50 END
```

3. SET #Channel_Number : POINTER SAME
moves the file pointer back to the beginning of the line read just before.

4. SET #Channe_Number : IF MISSING THEN EXIT DO
executes EXIT DO if the file pointer exceeds the end.

[Note]
Just after a file is opened with ACCESS OUTPUT, the file pointer points the end.

■ Other SET and ASK statements

Other SET statements and ASK statements

To set MARGIN or ZONEWIDTH for a file, use the following form of a SET statement, where n is a channel number (numeric expression).
SET #n : MARGIN m
SET #n : ZONEWIDTH m

ASK statements
The following ASK statements are available.
On the following, n is a channel number (numerical expression), a is a numerical variable，s$ is a string variable.
ASK #n: MARGIN a
ASK #n: ZONEWIDTH a
ASK #n: POINTER s$
ASK #n: NAME s$
ASK #n: ACCESS s$
ASK #n: RECTYPE s$
ASK #n: ORGANIZATION s$
ASK #n: ERASABLE s$
ASK #n: SETTER s$
ASK #n: FILETYPE s$
ASK #n: DATUM s$

■ CLOSE

CLOSE #

CLOSE #Channel_Number
releases the assignment of the channel.

Note.
When a program terminates even if it is caused by an exception, all the files that are open shall be closed.

Warning.
When a file is opened in a removable disk, do not remove it until the file is closed.

■ Internal Files

Record Type INTERNAL

1. About Record Type INTERNAL
Files of record type INTERNAL are those in which the records output by WRITE statements can be input by READ statements.

2 . OPEN statements
OPEN statements are written in the following form.
OPEN #Channel_Number NAME File_Name ,RECTYPE INTERNAL
OPEN #Channel_Number NAME File_Name ,ACCESS OUTIN ,RECTYPE INTERNAL
OPEN #Channel_Number NAME File_Name ,ACCESS INPUT ,RECTYPE INTERNAL
OPEN #Channel_Number NAME File_Name ,ACCESS OUTPUT ,RECTYPE INTERNAL
(Omitting ACCESS-clause means OUTIN)

3. WRITE Statements
To output data to a file of record type internal, use WRITE statements instead of PRINT statements.
Output items are written in a WRITE statement separated by commas.
No semicolon can be written, and no comma can be written on the tail.
Example 1.

```
10 OPEN #1: NAME "A:TEST.CSV",RECTYPE INTERNAL
20 ERASE #1
30 WRITE #1: 1/7, "ABC"
40 CLOSE #1
50 END
```

4. READ statements
To input data from a file of record type internal, use READ statements instead of INPUT statements.
Example 2.

```
10 OPEN #1: NAME "A:TEST.CSV",ACCESS INPUT, RECTYPE INTERNAL
20 READ #1: A,S$
30 PRINT A,S$
40 CLOSE #1
50 END
```

5. MAT WRITE, MAT READ
Arrays that was output to a file of record type internal using a MAT WRITE statement can be read using a MAT READ state-

ment.
Example 3. Output

```
10 DIM A(4),B$(2,2)
20 MAT READ A,B$
30 DATA 1,2,3,4,ABC,DEF,HIJ,KLM
40 OPEN #1:NAME "A:MAT.TXT",RECTYPE INTERNAL
50 ERASE #1
60 MAT WRITE #1: A,B$
70 CLOSE #1
80 END
```

Example 4. Input

```
10 DIM A(4),B$(2,2)
20 OPEN #1:NAME "A:MAT.TXT",RECTYPE INTERNAL
30 MAT READ #1: A,B$
40 CLOSE #1
50 MAT PRINT A,B$
60 END
```

An example dealing with a variable length array
Example 5. Output

```
10 DIM A(100)
20 MAT INPUT A(?)      ! The size of A varies
30 OPEN #1:NAME "A:MAT.TXT",RECTYPE INTERNAL
40 ERASE #1
50 WRITE #1:UBOUND(A)  ! Record the number of elements
60 MAT WRITE #1:A
70 CLOSE #1
80 END
```

Example 6. Input

```
10 DIM A(100)
20 OPEN #1:NAME "A:MAT.TXT",RECTYPE INTERNAL
30 READ #1:N           ! The number of elements
40 MAT READ #1:A(N)    ! Read dada after chaging the size of the array
50 CLOSE #1
60 MAT PRINT A
70 END
```

Notes.
On this BASIC, internal record type is comma-separated values(CSV).

Numeric values are expressed by figures. E-notation may be used.
A String is always put in double quotes. When a string contains a double quote, it shall be duplicated.
The end of record is the characters CR(CHR$(13)) and LF(CHR$(10)).

For example, After the execution of Example 1, the contents of "A:TEST.CSV" becomes
.142857142857143 ,"ABC" (CR and LF follow)

A MAT WRITE statement outputs all elements of arrays separated by commas.
For example, After execution of Example 3, the contents of "A:MAT.TXT" becomes
1 , 2 , 3 , 4 ,"ABC","DEF","HIJ","KLM"

Note.
The complex operation mode and the rational operation mode is incompatible with internal record type.

■ Stream Files

STREAM File

An internal record type file can be made of a STREAM file.
A stream file is a sequence of values, rather than records.
To manipulate stream files, USE the following form of OPEN statements.
OPEN #Channel_Number NAME File_Name ,RECTYPE INTERNAL ,ORGANIZATION STREAM
OPEN #Channel_Number NAME File_Name ,ACCESS OUTIN ,RECTYPE INTERNAL ,ORGANIZATION STREAM
OPEN #Channel_Number NAME File_Name ,ACCESS INPUT ,RECTYPE INTERNAL ,ORGANIZATION STREAM
OPEN #Channel_Number NAME File_Name ,ACCESS OUTPUT ,RECTYPE INTERNAL ,ORGANIZATION STREAM

Note.
On this BASIC, the separator of values is the characters CR(CHR$(13)) and LF(CHR$(10)).
That is, one value is recorded in one line. Numeric values are recorded in figures, Strings are quoted.
Example.

```
100 DIM a(4),b$(3)
110 DATA 1,2,3,4,a,b,c
120 MAT READ a
130 MAT READ b$
140 OPEN #1: NAME "a:stream.txt", RECTYPE INTERNAL, ORGANIZATION STREAM
150 ERASE #1
160 MAT WRITE #1:a
170 MAT WRITE #1:b$
180 CLOSE #1
190 END
```

The above program makes "a:stream.txt"

```
1
2
3
4
"a"
"b"
"c"
```

CSV Files

CSV File

[CSV = comma separated values]
On this BASIC, internal record type files are CSV files, which are commonly used on spreadsheets.
One line of a CSV file is one record of an internal record type file.
That is, one execution of a WRITE or READ statement carries out output or input of one line of a CSV file.

Example.
Outputs data on arrays declared with DIM s$(10), h(10), w(10) to a CSV file.
The file output can be read as a table of 3 columns, 10 rows in spreadsheets.

```
OPEN #1:NAME "A:DATA.CSV",RECTYPE INTERNAL
ERASE #1
FOR i=1 TO 10
  WRITE #1:s$(i),h(i),w(i)
NEXT i
CLOSE #1
```

Conversely, the following program reads CSV table data of 3 columns, 10 rows into BASIC.

```
DIM s$(10), h(10), w(10)
OPEN #1:NAME "A:DATA.CSV",RECTYPE INTERNAL
FOR i=1 TO 10
  READ #1:s$(i),h(i),w(i)
NEXT i
CLOSE #1
```

A matrix data should be handled as a sequence of rows.
Example.
Reads CSV data of 4 columns and 10 rows into an array declared with DIM A(10,4).

```
OPEN #1:NAME "A:DATA.CSV",RECTYPE INTERNAL
FOR i=1 TO 10
  READ #1:a(i,1),a(i,2),a(i,3),a(i,4)
NEXT i
CLOSE #1
```

This data can be re-written into the file as follows.

```
OPEN #1:NAME "A:DATA.CSV",RECTYPE INTERNAL
ERASE #1
FOR i=1 TO 10
  WRITE #1:a(i,1),a(i,2),a(i,3),a(i,4)
NEXT i
CLOSE #1
```

■ COM-Ports (Serial ports)

COM-port (Serial Port)

1. Setting the communication condition
Use Device Manager on Windows Control Panel to set the communication condition of a COM port.
Make baud rate, data bits, parity, stop bits, flow control agree with the opposite side.
Note that flow control RTS/CTS agrees with flow control hardware on the PC side.

2. OPEN statement
To use a COM port, use such an OPEN statement as
OPEN #1: NAME "COM1"
Just after the execution of the OPEN statement, receiving starts and data are stored on the buffer even if no INPUT statement is executed.
Since data received before the execution of the OPEN statement are canceled, data transmission from the device must start after the execution of the OPEN statement.
[Supplement]
The communication conditions can be written in the form similar to the MODE command of MS-DOS just after the PORT name with a colon.
Example 1. OPEN #1:NAME "COM1: baud=9600 parity=N data=8 stop=1"
Example 2. OPEN #1:NAME "COM1: 9600,n,8,1,p"
In case of example 2,
first parameter baud rate
second parameter parity, either of n(none)，e(even)，or o(odd)
third parameter data bits (character size), ordinary either 7 or 8
forth parameter stop bits, either 1, 1.5, or 2
fifth parameter, ordinary either x（handshake with xon/xoff）or p（hardware handshake)

3. In case of line-based communication
Use SET ENDOFLINE, to make the end of line code agree with the opposite side.
Example. SET #1: ENDOFLINE CHR$(13)
If the EOL of the opposite side is CR+LF, this process is not required.

4. In case of character-based communication
Use a PRINT statement that has a semicolon at the tail to transmit a chaarcter, and use a CHARACTER INPUT statement to receive a character.

5 In case of binary data

Append a OPTION CHARACTER BYTE line in the program unit in order to manipulate a byte as a character.
Use ORD and CHR$ functions to convert a character to a byte and vice versa.

6. Receive Buffer
Data are received and accumulated in the buffer, unless any INPUT statement is executed.
The number of characters (in bytes) accumulated in the buffer can be known by exectuing an ASK statement with CHARACTER PENDING.
Example. ASK #1: CHARACTER PENDING n

7. CLOSE statement
Execute a CLOSE statement to terminate the COM port.
Example. CLOSE #1

There are some sample programs of using a COM port in COMM subfolder of the folder in which BASIC is installed.

■ File statements (original enhancements)

Original Enhancement (files)

SET #Channel_number : ENDOFLINE string_expression
sets the end-of-line (or end-of-record) character that is used on the statements PRINT, INPUT, LINE INPUT, WRITE, READ.
Available values for string_expression are CHR$(13)&CHR$(10), CHR$(13) or CHR$(10).
The default EOL is CHR$(13)&CHR$(10).
CHARACTER INPUT statements ignore the setting made by this statement.

ASK #Channel_Number : FILESIZE numeric_variable
The file size in bytes is assigned to numeric_variable.

SET DIRECTORY a$
changes the current directory to the directory that a string expression a$ indicates.

ASK DIRECTORY s$
assigns the string that indicates the current directory to a string variable s$.

FILE GETNAME s$
FILE GETNAME s$, string_expression
displays an open-file-dialog and assigns the file name to a string variable s$,
where string_expression indicates the default extension.

FILE SPLITNAME (a$) path$, name$, ext$
splits the path name given by a string expression a$ into three parts, the path, the name body, and the extension, and assigns them to string variables path$, name$, ext$, respectively.

FILE RENAME a$,b$
renames the file a$ to b$, where a$, b$ are string expressions.
If the file a$ does not exist, an exception of extype 9003 shall be raised,
if the file b$ already exists, an exception of extype 9004 shall be raised,
and if renaming failed by another reason, an excetion of extype 9000 shall be raised.

FILE DELETE a$
deletes the file a$, where a$ is a string expression.
If the file does not exist, an exception of extype 9003 shall be raised, and if deleting failed by another reason, an exception of extype 9000 shall be raised.
This corresponds to UNSAVE of True BASIC and KILL of Microsoft BASIC.

FILE LIST a$,s$
assigns the whole list of the files that match a$ to s$,
where a$ is a string expression and s$ is a 1-dimensional string array.
a$ may have wildcard characters.
The upper bound of s$ shall be changed according to the number of the files matching.
However, if no file matches, an exception of extype 6005 shall be raised,
and if the number of the files exceeds the capacity of the array, an exception of 5001 shall be raised.

Supplied Functions
FILES(a$)
The number of the files that match a$, where a$ is a string expression. 0 if no file matches.

Example.
A program that changes all letters contained in the directed file in upper case.

```
100 FILE GETNAME s$
110 FILE SPLITNAME(s$) path$,name$,ext$
120 LET t$=path$ & name$ & ".TMP"
130 OPEN #1:NAME s$
140 OPEN #2:NAME t$
150 ERASE #2
160 DO
170    LINE INPUT #1,IF MISSING THEN EXIT DO:line$
180    ET line$=UCASE$(line$)
```

```
190    PRINT #2:line$
200 LOOP
210 CLOSE #2
220 CLOSE #1
230 FILE DELETE s$
240 FILE RENAME t$,s$
250 END
```

An example of using FILES and FILE LIST

```
10 LET s$="c:\BASICw32\sample\*.bas"
20 LET n=FILES(s$)
30 IF n>0 THEN
40    DIM names$(n)
50    FILE LIST s$, names$
60    MAT PRINT names$
70 END IF
80 END
```

■ Printer (original ehhancements)

Original Enhancement (Printer)

OPEN #Channel_Number: PRINTER
assigns the channel to the printer.
Outputs are kept not to be transmitted to the printer until the CLOSE statement is executed.
If an ERASE statement is executed, the former output is erased.
The printer to be used in this statement is settled according to the file menu - Printer.
If PRINTER is doubly opened, an exception of extype 9004 shall be raised.
Example.

```
10 OPEN #1: PRINTER
20 SET #1: MARGIN 80
30 FOR i=1 TO 100
40    PRINT #1: SQR(i),
50 NEXT i
60 CLOSE #1
70 END
```

Note.
If you want to print on real-time, output data directly to the port to which the printer is connected.
Example.

```
10 OPEN #1: NAME "LPT1:"
20 FOR i=1 TO 10
30    INPUT n
40    PRINT #1: n,SQR(n)
50 NEXT i
60 CLOSE #1
70 END
```

TextWindow (original enhancements)

Proprietary extensions (TextWindow1, TextWindow2)

TextWindow2 TextWindow1 devices and text-compatible text.
Select to display the View menu, you can manually override the content.
Do not hold to erase a program runs.
Use the following syntax. TextWindow2 TextWindow1 may be part of the writing.
OPEN # numeric expression: TextWindow1
OPEN # numeric expression: TextWindow1, ACCESS INPUT
OPEN # numeric expression: TextWindow1, ACCESS OUTPUT
OPEN # numeric expression: TextWindow1, ACCESS OUTIN

Examples TextWindow1 entering data and running the numbers separated by spaces, TextWindow2 output is converted to a comma delimited data.

```
100 OPEN # 1: TextWindow1
110 OPEN # 2: TextWindow2
120 ERASE # 2
130 DO
140 LINE INPUT # 1, IF MISSING THEN EXIT DO: s $
150 LET i = 1
160 DO
170 DO WHILE s $ (i: i) = ""
180 LET i = i +1
190 LOOP
200 LET i0 = i
210 DO UNTIL i> LEN (s $) OR s $ (i: i) = ""
220 LET i = i +1
230 LOOP
240 PRINT # 2: s $ (i0: i-1);
250 IF i> LEN (s $) THEN EXIT DO
260 PRINT # 2 :",";
270 LOOP
280 PRINT # 2
290 LOOP
300 CLOSE # 2
310 CLOSE # 1
320 END
```

Debugging

Debug

DEBUG

Full BASIC has debugging facilities BREAK and TRACE.
When you use these facilities, you must execute DEBUG ON beforehand.

DEBUG ON
activates the debug facilities.
DEBUG OFF
inactivates the debug facilities.
The default is OFF.
The scope of a DEBUG statement is the program unit, and the setting remains hereafter statically.
(In this sense, DEBUG has abnormality among the statements of Full BASIC.)

BREAK
If BREAK is executed when debug facilities are activated, BREAK interrupts the execution of the program and displays the debug window, provided that when DEBUG is written on a when-body, DEBUG raises an exception of extype 10007.
If debug facilities are not activated, BREAK does nothing.

TRACE ON
If TRACE ON is executed when debug facilities are activated, trace facility is activated on the program unit where is executed.
Trace results are output to the same window as PRINT statements use.
Trace reports are the statements executed, the variables varied and the new value of them.

TRACE ON TO #Channel_Number
Trace reports are output to the channel.

TRACE OFF
Trace facility shall be inactivated on the program unit.

Note.
The scope of a TRACE statement is the program unit. Any program unit is set trace off on the invocation.

Note.
Usage of the debug window is convenient rather than the debug using program codes.
To display the debug window, select RUN-STEP.
Refer to debug Window

Complex numbers and rational numbers

Complex numbers

Complex numbers

Complex numbers

Complex Numbers

If complex mode()is selected or
OPTION ARITHMETIC COMPLEX
is written in every program unit, complex numbers can be handled.

Complex numbers are expressed as a pair of double precision binary floating point numbers.
An imaginary number is outputted in parenthesized form, however this form cannot be used for constants or inputs.
On a power operation, if the base is positive, the exponent can be a imaginary number. If the exponent is an integer of -2147483647 to 2147483647, the base can be an imaginary number.
The domains of functions ABS(z), SQR(z), EXP(z), and LOG(z) are expanded to complex numbers.
COMPLEX(x,y), RE(z), IM(z), CONJ(z), and ARG(z) are the functions only for complex mode.
Graphic transform functions SCALE and SHIFT are extended to complex numbers.
Internal files does not support complex numbers. That is, if complex numbers are written to an internal file, they can not be read.

Complex functions

Complex Functions

Let *i* be an imaginary unit.

Let *x* , *y* be real numbers, *z* be a complex

COMPLEX(x , y)	Complex number x + y i
RE(z)	The real part of z, i.e., RE(x + y i)= x
IM(z)	The imaginary part of z, i.e., IM(x + y i)= y
CONJ(z)	The conjugate of z, i.e., CONJ(x + y i)= x - y i
ARG(z)	The argument of z, i.e., arg z . The result depends on the angle option. In case of angle radians, the result is greater than -π and not greater than π. In case of angle degrees, the result is greater than -180° and not greater than 180°,
ABS(z)	The absolute value of z, |z|
SQR(z)	The square root of z, the argument of which is greater than that -π and not greater than π. For example, SQR(-1)=i.
EXP(z)	The exponential function.
LOG(z)	The natural logarithm of z, i.e., the complex number with real part log|z|, imaginary part arg z, where the unit is radians and is greater than π and not greater than π

<Note> EXP(z) and LOG(z) are not affected by the angle option.

- **Enhancement on Transform Functions**

Complex Transformations ™

Let *a* be a complex number.

SCALE(*a*)	If *a* is imaginary, SCALE(*a*) is equivalent to SCALE(ABS(*a*), ABS(*a*))*ROTATE(ARG(*a*)) That is, the map $z \to az$
SHIFT(*a*)	Equivalent to SHIFT(RE(*a*),IM(*a*)) That is, the map $z \to z+a$

<Note>
The map $z \to a\,z + b$ is SCALE(*a*)*SHIFT(*b*).
The map $z \to a(\,z - b) + c$ is SHIFT(-*b*)*SCALE(*a*)*SHIFT(*c*).
The map $z \to$ CONJ(z) is SCALE(1,-1).

If the transformations that corresponds to the maps *f* and *g* are F and G, respectively, the transformation corresponding to the composite map $g \circ f$ is F*G.

■ Rational numbers

- **Rational Numbers**

Rational Numbers

If the rational number mode is selected (p/q), or
OPTION ARITHMETIC RATIONAL
is written in a program unit, calculation becomes rational.
Digits have no upper bounds.
Rational numbers are outputted as a form of a improper fraction, but this form cannot be used for inputs or datum.
Rational numbers can be outputted as a decimal fraction by use of PRINT USING, whereas PRINT USING can deal at most 1000 digits.
The base for a power operation is limited to an integer of -2147483647 to 2147483647.
Any irrational functions such as SQR and trigonometric, exponential , or logarithmic functions are not available.
Internal files cannot contain rational numbers. That is, if rational numbers are outputted to an internal file, they cannot be read by a READ statement.
<Remarks>
Line breaks may be inserted in a numerals written to the text window at every about 3000 digit. This is due to a specification of MS-Windows. To avoid this, write to a file.
Ref. Overview of Files

■ Functions on Rational mode

Functions for the rational mode

INTSQR(x)	the value of the positive square root of x truncated to an integer.
NUMER(x)	The numerator of x.
DENOM(x)	The denominator of x
.GCD(x,y)	The Greatest Common Divisor of x and y. Results have positive signs. If both x and y are equal to zero, the exception with extype 3006 shall be raised.

Language Specifications

■ Differences from the standard

Specification Details

We describe what differs from ANSI Full BASIC.
These descriptions presume that the compatibility options are all set to ISO.

References
(1)ANSI X3.113-1987 American National Standard for Information Systems -Programming Languages- Full BASIC
(2)ANSI X3.113a-1989 American National Standard for Information Systems -Programming Languages- modules and individual character input for Full BASIC

These documents can be purchased from TECHSTREET
http://www.techstreet.com
Digital documents also can be purchased from ANSI ONLINE
http://www.ansi.org/

ISO Full BASIC is a super-set of ECMA BASIC.
Decimal BASIC is compatible with ECMA BASIC-1, ECMA GRAPHICS module, and the EXCEPTION HANDLING part of BASIC-2.
The reserved words of Decimal BASIC are those of BASIC-2.

ECMA-116 BASIC (pdf)
http://www.ecma-international.org/publications/files/ECMA-ST-WITHDRAWN/ECMA-116, 1st edition, June 01986.pdf

[View ECMA-116 (PDF)] (67.62MB)

2. Conformance with the standard

Specification Details

Conformance

[Disagreement with ANS]
2.2
This implementation does not conform to any set of modules.
2.4
Some overflow error occurs on compiling.
In this case, the program cannot be executed. (See 5.1.5 [Numbers])

4. Program Elements

Specification Details

4. Program Elements

[Disagreement with ANS]

4.2
Line numbers can be omitted.
4.2.1
Any parallel section cannot be contained.

[Implementation-defined]
4.2.2 16.
end of line□□CR LF
4.2.4
The program name is independent of the program-designator.
4.2.4
Parameters in a program-name-line for a program executed in isolation are taken from the additional parameters for the command prompt that indicates basic and the program file name.

5. Numbers

Specification Details

5. Numbers

[Disagreement with ANS]
5.1.5
If the absolute value of a numeric constant is extremely large, an error occurs on compiling.
5.4.4
On the explanation of EPS(x), x, x', and x" are chosen on the accuracy of variables.

[Implementation-defined]
5.1.4
Numerical constants have a precision between 19 digits and 27 digits.
5.2.4
The initial value of a variable is zero.
5.3.6
An underflow is replaced by zero.
5.3.6
Operations of the same priority are performed from left to right.
5.4.4
If arithmetic option is DECIMAL,
EPS(0)=1.E-99
MAXNUM=1.E99.
The values of RND functions are pseudo-random numbers.
5.4.6
Randomize-statement uses the real-time clock.
5.6.4
m=15, i.e. a numeric variable has precision of 15 digits.
5.6.4
Numeric constants and results of addition, subtraction, multiplication and division have 19 to 27 digits of precision. The result of a transcendental function, including a power operation, has 17 digits of precision, where the 17th digit is even. The other supplied functions have same digits of precision as above.
5.6.4
If arithmetic option is NATIVE, numbers are double precision numbers of Intel 80x87 FPU, the positive minimum number is about 4.9E-324 , and the maximum number is about 1.8E308.
[Note]
Accuracy of numerical operations is not guaranteed.

6. Strings

Specification Details

6. Strings

[Disagreement with ANS]
6.4.4
USING$
If an exception provided in 10.4.5 occurs on USING$, the default handler does not report the unformatted representation of the value on the next line.
MAXLEN(s$)
returns 2147483647 if the length is not limited.

[Implementation-defined]
6.1.6
The maximum physical line length depends on the RichEdit control of Windows.
6.2.4
The maximum length of a string variable depends on the system resource and is at most 2147483647.
6.2.4
The initial value of a string variable is a null string.

7. Arrays

Specification Details

7. Arrays

[Disagreement with ANS]
7.1
An index of an array must be in the range of -2147483647 to 2147483647.

[Implementation-defined]
7.2.6
An underflow does not cause an exception.
7.3.4
If an array is assigned to an array with a substring qualifier and the size of those dose not coincide, an exception with extype 6101 shall be raised. Note that if the size of left-hand array is smaller than that of right-hand one then extype shall be 5001.

8. Control Structures

Specification Details

8. Control Structure

[Disagreement with ANS]
None.

9. Program Segmentation

Specification Details

9. Program Segmentation

[Disagreement with ANS]
9.3.5
An exception with extype 4301 may not be raised. This exception is raised on the program-line and not propagated to the chain-statement. An exception with extype 4303 shall not occur.

[Implementation-defined]
9.3.4
The initial values of variables in a chained-to program are zero or the null string.
[Memo]
The program shown on 9.2.6 has an extra keyword "INTERNAL". Line 100 should be "100 DECLARE SUB S".

10. Input and Output

Specification Details

10. Input and Output

[Disagreement with ANS]
10.1.2 8.
An empty datum may be allowed.
10.1.2 9.
The definition of plain-string-character is extended.
10.4
In a format string, the characters except those used in format-items and commas not succeeding a digit-place are transacted as literal-items.
10.5.4.5
@ After all arrays are outputted, an extra end-of-line shall be generated.

[Implementation-defined]
10.2.2 9.
@AS in 10.2.6.
10.3.4.3
Tabular control may be confused by some control characters such as CR, LF.

11. Files

Specification Details

11. Files

This system implements only the core module. The other file facilities are not available.

[Disagreement with ANS]
11.1.4.9
The value of the variable for the maximum length of records may be 2147483647, if it should be maxnum.
11.4.5
Extype 8101 may be applied to internal files.

■ 12. Exception Handling and Debugging

Specification Details

12. Exception Handling and Debugging

[Disagreement with ANS]
12.1.4
If the line number is omitted, the value of EXLINE is 0.
12.1.4
If an exit-do or exit-for statement is executed in the exception-handler of a when-use-block that is contained in a do-block or for-block, the control escapes from the exception handler. (ANS does not mention this.)
12.1.4
When an exception is caused by a statement lexically within an exception-handler, this exception is handled as an ordinary exception.
12.1.6 Remarks
When an exception occurs within a def-statement, this system obeys the rule in 12.1.4. (The description in ANS is probably incorrect.)
12.2.4
Results of a trace-statement may slightly differ from the standard, in particular, even if an assign-statement is executed, no report shall be outputted if the value of the variable is not changed.
12.2.6
Only array elements that are changed in values are reported.

Note.
On the description at 12.1.4
"A separate GOSUB stack is associated with each exception-handler (cf. 8.2) ... ",
"exception-handler" seems to be a mistake for "detached-handler".

13. Graphics

Detailed Specification

13. Graphics

[Disagreement with ANS]
13.3.4.3
Even if an exception occurs in formatting, the unformatted representation that is prescribed in 10.4.5 shall not be outputted.
13.3.5
Even if the geometric object of a GRAPH LINES is fewer than two points, no exception shall be raised.
13.4.4
Locate-statements and mat-locate-statements with a point-clause are available only on the BITMAP graphics mode. On execution of ASK PIXEL SIZE, the perimeter of the rectangle is assumed to be 'within' it.
13.5.4.2
The beam shall be turned off before executing a MAT PLOT LINES statement.

[Implementation-defined]
13.1.4
The Unit for device viewport and device size is meters.
13.1.6
Inverted windows are valid.
13.3.6
On execution of an array-cells statement, a zero-width or zero-height rectangular shall not be drawn.
13.3.6
An exception with extype 6401 shall occur for a two-dimensional numeric-array in an array-point-list when the size of its second dimension is greater than 2.
13.5.4
If a transformation is singular, an exception of extype -3009 shall be caused.
[Note]
Full BASIC may have ambiguous sentences.
Example.

```
10 DEF ROTATE(x)=2*x
20 DIM a(4,4)
30 MAT a=IDN
40 MAT a=ROTATE(PI/6)*a
```

The 20-line can be interpreted as both a numeric mat-statement and a transformation mat-statement. In these cases, THIS BASIC gives priority on the interpretation as a numeric mat-statement.

■ Individual Character Input

Specification Details

17. Individual Character Input (Addendum)

[Disagreement with ANS]
17.1

A char-input-reply can be an other-character. End-of-line in a char-input-reply is not defined. CR and LF are treated as two other-characters.
[Implementation-defined]
17.4

Pending characters shall be erased when a chain-statement is executed. Character echoing is available only for the keystrokes.
17.6

There are no typeahead buffers for INPUT and LINE INPUT. SET ECHO is valid for INPUT and LINE INPUT. An other-character is defined as a byte. If an end-of-file occur during the execution of a char-input statement, an exception with extype 7305 shall be raised.

■ Modules (Addendum)

Specification Details

18. Modules (Addendum)

[Disagreement with ANS]
18.2
A function name or routine name and parameters written in DEF-line or routine header cannot be of the form in 27., 28., 29., and, 30.. So is a function name written in a function-let-statement. In the rule 26., defined-function, routine-identifier, simple-numeric-variable, simple-string-variable, and formal-array cannot be of the form in 27., 28., 29., and ,30.. A numeric function declared in a share-statement or a public-statement must be the same arithmetic option as the module body.
18.2
An OPTION ARITHMETIC□Cor MODULE OPTION ARITHMETIC must precede PUBLIC NUMERIC statements and SHARE NUMERIC statements. An OPTION BASE or MODULE OPTION BASE must precede PUBLIC-statements, and SHARE-statements.
18.4
External subprograms and external pictures contained in a module are assumed to be declared PUBLIC. Even if they are declared SHARE, they are assumed PUBLIC.
18.4
Module bodies are executed before execution of the main-program. If a program has two or more modules, the first module shall be executed first.
[Memo]
The Syntax defined on 18.2 is thought to be immature.

Reference:
ANSI body and ANSI Addendum provide the following syntax.

qualified-id ⊃ module-name period simple-numeric-variable (18.2 (26))
simple-numeric-variable = numeric-identifier (5.2.2 (2))
numeric-identifier ⊃ module-name period letter identifier-character* (18.2 (27))

From these, the following syntax can be deduced.

qualified-id ⊃ module-name period module-name period letter identifier-character*

■ Supplementary Explanations

Specification Details

Supplement

The features defined on the chapter 14. Real-time, 15. Fixed Decimal Numbers, 16. Editting are not implemented.

A Note on DECIMAL operation.
Because the evaluation precision of numeric expressions is higher than that of variables, the result of the following program never be zero on the decimal operation mode.

```
10 LET A=(1/3)*3
20 PRINT A-(1/3)*3
30 END
```

(This is true as far as the implementation complies with the standard.)

■ Restrictions

Specification Details

Restrictions

[Syntax]

Program Length
The length of a program is at most 1M bytes.

[Run time]

System Stack Size
The system stack size is 48M bytes.
The system stack is exhausted by execution of a recurrence.
The system stack size shall be set to be 1M bytes for the first, and can be enlarged to at most 48M bytes for the need.

Virtual Stack Memory
Virtual Stack Memory is used for variable management.
The size is self-adjusted on the start-up time according to the situation of demands.
However, users is allowed to set the size of Virtual Stack Memory directly manipulating the 'VirtualMemry' key on the [Frame] section of BASIC.INI.
An error message 'EXTYPE -103 Virtual memory not allocated' suggests that id should be increased,
and an error message 'out of memory' suggests that it should be decreased.

Size of an array
The size of an array is at most 34217728,
provided that a memory allocation allows the size.

File Size
This system is made to cope with a file of size over 4GB, but no one may tested it.

What differs from previous versions

What differs from version 3

Differences with the previous 3.38
write sentence
Keyed to the write operation of the JIS statement.
When the input in the form of export import statement, you must open the file in internal format.
The open sentence ", RECTYPE INTERNAL" Please add.

Differences with the previous 3.59
Debug state
Debugging options for the state abolished a program runs. If you use a Break statement at the beginning of each program unit please write a DEBUG ON.

Differences with the previous 3.68
RANDOMIZE
Randomize statement that the seed for Microsoft to write only mode.

Differences with the previous 3.70
ASK PIXEL SIZE
To count for points on the edges and vertices of the rectangle.
However, the extension orders
ASK PIXEL SIZE X, Y
There are no changes.
Therefore,
SET WINDOW L, R, B, T
ASK PIXEL SIZE (L, T; R, B) A, B
Obtained by running the A, B and
ASK PIXEL SIZE X, Y
Obtained by running the X, Y between
X = A-1, Y = B-1
Differences arise.

Differences with the previous 3.74
REDIM statement
REDIM to initialize the array elements do not even execute the statement.

When you create a program does not work correctly in earlier versions, REDIM statement followed
MAT A = ZER
Please add (to the target array and A).

Differences with the previous 3.85
CHARACTER INPUT type-ahead buffer is attached to the statement.
CHARACTER INPUT NOWAIT: s $
CHARACTER INPUT # n, NOWAIT: s $
JIS to conform to the provisions of the absence of characters into a buffer, s $ will remain so for the previous value.
If there is no character in the buffer s $ = "" If you want to be with, CHARACTER INPUT statement in front of
LET s $ = ""
Please add.

■ What differs from version 4

Differences with the previous 4.11

OPEN filename when no extension statement, if a file exists by that name, ". TXT" decided not to attach.

Differences with the previous 4.12
Changed from upper left to lower left edge pixel coordinate origin.
Y pixel coordinates of the positive sense to raise.
More

When you create a program does not work correctly in earlier versions,
PIXELY, WORLDY, PROBLEMY the start of the program units that use
DECLARE EXTERNAL FUNCTION PIXELY, WORLDY, PROBLEMY
Adding further, please add the following external function definition at the end of the program.

```
EXTERNAL FUNCTION PIXELY (y)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
LET PIXELY = ROUND ((py-1) * (top-y) / (top-bottom), 0)
END FUNCTION

EXTERNAL FUNCTION WORLDY (y)
ASK WINDOW left, right, bottom, top
ASK PIXEL SIZE (left, bottom; right, top) px, py
LET WORLDY = top-(top-bottom) * y / (py-1)
```

END FUNCTION

EXTERNAL FUNCTION PROBLEMY (y)
DECLARE EXTERNAL FUNCTION WORLDY
LET PROBLEMY = WORLDY (y)
END FUNCTION

Differences with the previous 4.25
DIM changed the design so as not to redefine the bounds of more elements than declared in the statement.
Cases DIM A (10) for an array declared in A, N> 10 when
MAT A = ZER (N)
When you run, an exception condition (extype 5001)'s.

Differences with the previous 4.54
1E99 MAXNUM the mode in decimal, EPS (0) was changed to 1E-99.

Differences with the previous 4.63
Lift restrictions on the number of digits in a rational mode.
MAXNUM rational mode and undefined.

4.9.2 Differences with the previous
You can not use the name begins with full-width alphanumeric identifiers.

4.9.4 Differences with the previous
Built-in function name string SUBSTR $, MID $, LEFT $, RIGHT $ added.
The program uses these names as an external function name, in each program unit to use them
DECLARE EXTERNAL FUNCTION SUBSTR $, MID $, LEFT $, RIGHT $
Such a statement (if that is omitted if) you need to add.

■ What differs from version 5

5.0.8 Differences with the previous
When the destination graphics printer line width, axis color of the initial value, respectively, one changed to 15 (the same as if you printed to the screen.)

Differences with the previous 5.2.x
DISK DRAW decided to change the status of the 描点.

GRAPHICS DEVICE PRINTER statement was changed to non-executable statements.

5.3.5 Differences with the previous
The syntax of the menu options "to force variable declaration" when wearing a check, we can not allow the abbreviated name of the function declaration.

5.3.7 Differences with the previous
CHAIN statement to change the behavior.
CHAIN programs started to boot from BASIC statements are executed in the window.
PRINT statement is the result of output followed by the text window.
CHAIN run in graphics over the image before executing statement.
Remain after the execution of the program text editor.
If you keep running in compatibility with previous versions, CHAIN EXECUTE NOWAIT rewrite the line, please insert the next STOP statement.

CHAIN statement, EXECUTE BASIC statements when you specify a file with an extension of the program, not the association, directly BASIC.EXE changed to start and run.
If you want to start to open the file, please rewrite the following form.
Aya Tokino CHAIN
EXECUTE NOWAIT "BASIC.EXE" WITH ("/ NR", "ABC.BAS")
STOP
Aya Tokino EXECUTE
EXECUTE "BASIC.EXE" WITH ("/ NR", "ABC.BAS")

5.6.4 Differences with the previous
COLOR MODE
COLOR MODE changed when you change the color treatment.
CLEAR statement is used before changing the background color mode.
Time to keep backward compatibility, color mode before switching to the native
SET COLOR MIX (0) 0,0,0
Please run.

5.8.9 Differences with the previous
PLOT TEXT
When you run the shape transformation, to convert fixed orientation of the character and magnitude.
To maintain compatibility with previous versions, PLOT TEXT Please rewrite the PLOT LABEL.

■ What differs from version 7

7.0.1 Differences with the previous
ASK # n: CHRACTER PENDING
When you open a normal file in a format, EOF 0, then the return to a changed otherwise.

7.2.1 Differences with the previous
OPTION ARITHMETIC DECIMAL_HIGH
Or
OPTION ARITHMETIC RATIONAL
Writing a built-in function EXP (), LOG (), SIN (), COS () and made to be enabled.
If you are using an external function definition with the same name built-in functions of these
DECLARE EXTERNAL FUNCTION statement is mandatory by the declaration.

7.3.5 Differences with the previous
And when the printer directly to a bitmap to be drawn, due to changes at run-time geometry shape of the characters to be drawn PLOT TEXT statement shall be defined by the coordinates for the problem. Therefore, the aspect ratio of 1:1 if the problem coordinate system is not distorted characters.
PLOT TEXT statement earlier statement that inherits the behavior of the newly established PLOT LETTERS.
If you draw a character in the eyes of the purpose of reading a person, PLOT TEXT PLOT LETEERS please rewritten to.
However, as usual metafile PLOT TEXT statement is incompatible with JIS.
The compatibility options "TEXT drawing" to "physical coordinates" By changing can also be performed based on physical coordinates of all the characters drawn.

7.4.0 Differences with the previous
LET function definition was changed to a local variable to hold the function call is reserved by the results of the statement.
JIS expected to operate as a program provisions may not operate correctly.
LET function name definitions JIS statement if you want to work as the provision of compatibility options "-defined function name" to "broad-based (JIS)" please change.

What differs from other BASICs

- **Differences between Minimal BASIC and Full BASIC**

- **Differences from the obsolete Minimal BASIC**

Differences from Minimal BASIC

The following modifications are needed for programs written in Minimal BASIC.

These are due to the incompatibilities between the Minimal BASIC standard and the Full BASIC standard.

1. When OPTION BASE is not written, option base is assumed to be 1 on Full BASIC while it is assumed to be 0 on Minimal BASIC.
   If arrays need indices 0, add
   OPTION BASE 0
   at the beginning.

2. Full BASIC demands all arrays to be declared.
   For all arrays not declared, add declarations such as
   DIM A(10),B(10),C(10)

Refer to Automated Correction

■ Differences from Microsoft BASIC

■ Differences from Microsoft BASIC

Differences between Full BASIC and Microsoft BASIC

Before proceeding to this section, confirm Differences from Minimal BASIC.

■ Variables and Operations

Difference between Full BASIC and Microsoft BASIC

Operations and variables

Numeric values have no distinction of types.
No numeric variables or numeric constants end with '#','!' or '%'.
There are no statements such as 'DEFDBL' or 'DEFINT'.

An array and a simple variable can not have the same name.

Numeric expression syntax is different.
Full BASIC does not allow expressions such as 'A*-3' or 'A^-2'.
These must be written as 'A*(-3)' or 'A^(-2)'.

Note that ' 2^3^4 ' means ' (2^3)^4 ' while an expression ' 2^-3^4 ' on Microsoft BASIC means ' 2^(-3^4) ' .

Full BASIC does not have MOD operator. To find a remainder, use a supplied function MOD(a,b),
provided that MOD(a,b) and 'a MOD b' coincides only if a and b are non-negative integers.
cf. MOD Detail

Use a LET statement for assignment. 'LET' can not be omitted.

The concatenation operator is not '+' but '&'.
For example, A$ + "s" should be changed to A$ & "s" .

Input and Output

Difference between Full BASIC and Microsoft BASIC

Input and Output

LPRINT statement does not exist.
On this BASIC, text output can be sent to a printer using the File Menu of the text output window.

INPUT a_string_constant ; a_variable
is not acceptable on Full BASIC. Correct syntax is as follows:
INPUT PROMPT a_string_expression : a_variable
Example
INPUT "A=" ; A
should be changed to
INPUT PROMPT "A=": A

The delimiter character followed by a format in a PRINT USING statement is not a semicolon but a colon.
Example.
Not
PRINT USING "#.#####"; A
but
PRINT USING "#.#####": A

Format strings are not identical. The common format characters are only three, '#','.' and '^'.
Refer to PRINT USING

- **Control statements**

Difference between Full BASIC and Microsoft BASIC

Control statements

A tail comment begins with not ' but ! .

There are no multi-statements.
Multi-statements on a IF-statement can be re-written using IF ~ END IF blocks.

Any nested single line IF statements are not allowed.
These can be re-written using IF ~ END IF blocks.

Omission of the variable on a NEXT statement is not allowed.

The role of the END statement is different.
The END statement of Full BASIC indicates the end of the main program.
cf. (END-statement)

STOP does not mean extraordinary finishing.
STOP corresponds a END command of Microsoft BASIC.

The result of a comparison is not a number.
AND, OR, NOT are not numerical operations.
' p AND q ' does not evaluate q if p does not hold.
' p OR q ' does not evaluate q if p holds.

There is no WHILE ~ WEND structure.
DO WHILE ~ LOOP is a substitute.
The conditions are not numbers.
For Example,

```
 WHILE 1
  ……
 WEND
```

should be changed to

```
 DO
 ……
```

LOOP

If a SELECT block has no CASE ELSE line, an exception shall be caused if there are no corresponding items. Thus, the CASE ELSE line may not be omitted.

A CHAIN statement does not hand over the variables.

Notable Syntax of Microsoft BASIC
---

1. Conditions are numbers on Microsoft BASIC.
For example,
IF A THEN …
yields … executed if the value of A is not zero.

2. The result of a comparison is a number, -1 for true, 0 for false.
Thus,
IF 1<A<3 THEN B=0
yields B=0 whatever the value of A is.

3. Multi-statements
Multi-statements do not mean sequential execution of two or more statements.
For example,
10 IF A=1 THEN GOTO 100 : IF A=2 THEN GOTO 200
and
10 IF A=1 THEN GOTO 100
20 IF A=2 THEN GOTO 200
have different meaning.

- **END statement**

Difference between Full BASIC and Microsoft BASIC

END statement

On Microsoft BASIC, END means finishing the program, but on ISO Full BASIC, END means that line is the final line of the main program.
Only external SUBs, external function definitions, external picture definitions, and modules are written followed by the END line.

Any subroutine invoked by a GOSUB statement is not put below the END-line.
DATA statements that supplies for READ statements in the mainprogram must be written in the mainprogram.

How to write DATA statements
DATA statements can be written anywhere above the END-line.
Wrong
```
10 READ A,B,C
. . . . . .
100 END
110 DATA ○, ○, ○
```

Right
```
10 DATA ○, ○, ○
20 READ A,B,C
. . . . . .
100 END
```

How to write a subroutine
Wrong
```
20 GOSUB 100
. . . . . .
90 END
100 REM
. . . . . .
200 RETURN
```
Right
```
20 GOSUB 100
. . . . . .
90 STOP
100 REM
. . . . . .
200 RETURN
210 END
```

- **MOD operator**

Difference between Full BASIC and Microsoft BASIC

MOD-operator and the MOD-function

MOD(a,b) is used to find the remainder of a divided by b on Full BASIC.
If a and b are both positive integers, a MOD b in Microsoft BASIC and MOD(a,b) coincides.
However, if a is negative or if a or b is not an integer, those do not coincide.
For Example,
(-1)MOD 3 = -1
MOD(-1,3)= 2

If you want to the result that coincide with the MOD operation of Microsoft BASIC, use REMAINDER(a,b).
For Example, REMAINDER(-1,3) = -1

While the MOD operation of Microsoft BASIC rounds the operands into integers, MOD functions and REMAINDER functions have the domain of the real numbers.
For example,
4.71 MOD 3.14 = 5 MOD 3 = 2
MOD(4.71, 3.14) = 1.57

Refer to Supplied Numerical Functions (general)

- **Supplied functions**

Difference between Full BASIC and Microsoft BASIC

Supplied Functions

Corresponding functions for the built-in functions of Microsoft BASIC

ASC	→ ORD
ASC(a$)	= ORD(a$(1:1))
ASC("x")	= ORD("x")
CINT	→ ROUND
FIX	→ IP
HEX$	→ BSTR$
HEX$(n)	= BSTR$(n,16)
INSTR	→ POS
INSTR(n,a$,b$)	= pos(a$,b$,n)
SPACE$	→ REPEAT$
SPACE$(n)	= REPEAT$(" ",n)
SPC	→ REPEAT$

SPC(n) = REPEAT$(" ",n)
STRING$ → REPEAT$
STRING$(n,a) = REPEAT$(CHR$(a),n)
STRING$(n,a$)= REPEAT$(a$(1:1),n)
STRING$(n,"x")= REPEAT$("x",n)

The above functions except SPC, STRING$(n,a) are defined as external functions in LIBRARY\MS.LIB.

INKEY$ → character input

```
EXTERNAL FUNCTION INKEY$
  SET ECHO "OFF"
  LET S$=""
  CHARACTER INPUT NOWAIT: S$
  LET INKEY$=S$
END FUNCTION
```

A sample for this lies in LIBRARY\INKEY$.BAS.
Since the INKEY$-function has no argument,
DECLARE EXTERNAL FUNCTION INKEY$
must be written in the program unit that invokes INKEY$.

Hexadecimal Constants

&H7F → BVAL("7F",16)

Substitution for MID$

MID$(a$,m,n)=s$ → LET a$(m:m+n-1)=s$

- **Files**

Difference between Full BASIC and Microsoft BASIC

Text Files

Corresponding commands shown to the right.

```
OPEN "ABC.TXT" FOR INPUT AS #1              OPEN #1: NAME "ABC.TXT" ,ACCESS INPUT
OPEN "ABC.TXT" FOR OUTPUT AS #1             OPEN #1: NAME "ABC.TXT"
                                            ERASE #1
OPEN "ABC.TXT" FOR APPEND AS #1             OPEN #1: NAME "ABC.TXT" ,ACCESS OUTPUT
PRINT #1, A;B,C$                            PRINT #1: A;B,C$
PRINT #1,USING "###";A                      PRINT #1,USING "###":A
INPUT #1,A,B,C$                             INPUT #1:A,B,C$
```

EOF-function

```
WHILE NOT EOF(3)                    DO
  LINE INPUT #3,A$                    LINE INPUT #3,IF MISSING THEN EXIT DO:A$
  PRINT A$                            PRINT A$
WEND                          LOOP
```

Open a file as internal for WRITE statements.
Example.
OPEN #1: NAME "ABC.TXT" ,RECTYPE INTERNAL

- **Error handling**

Difference between Full BASIC and Microsoft BASIC

Error Handling (Exception Handling)

Corresponding error handling is shown to the right.

```
100 ON ERROR GOTO 400           100 WHEN EXCEPTION IN
110 target                      110 target
                                120 USE
                                130 error handling
140 ON ERROR GOTO 0             140 END WHEN
150 next transaction            150 next transaction
……                        ……
390 END                         390 END
400 error handling
410 RESUME 140
```

- **Graphics**

Difference between Decimal BASIC and Microsoft BASIC

Graphics

1. Coordinates
A coordinate system is introduced by a SET WINDOW statement.
SET WINDOW has four parameters left, right, bottom, top.
That is, first two parameters are the range of x-coordinate, the rest are the range of y-coordinate.
SET WINDOW x1,x2,y1,y2
correspond to WINDOW (x1,y2)-(x2,y1) on Quick BASIC.
The positive direction of y-axis is upward, though a user can set the coordinate system y-axis downward.
Thus, WINDOW (x1,y1)-(x2,y2)
can be replaced by
SET WINDOW x1,x2,y2,y1
If you want to set the pane size 640×480, execute an original statement
SET BITMAP SIZE 640,480

2. Points and lines
In most cases, re-written can be done by
replacing PSET(x,y) by PLOT x,y
LINE (x1,y1)-(x2,y2) by PLOT x1,y1 ; x2,y2
LINE -(x,y) by PLOT x,y;
PLOT x,y; means LINE (x,y)- in the notation style of Microsoft BASIC.
The PLOT x,y; executed first does not draw a line except the point.
That is, the PLOT x,y; executed first determines the initial point of a line graph.
To terminate a line graph, add a PLOT statement that has no arguments.

Refer to Introduction to BASIC Curves and Graphs Foundation of Graphics

- **Special commands in N88BASIC**

Difference between Decimal BASIC and Microsoft BASIC

Major differences from other BASIC interpreters
Variables are initialized for every execution.
The dimension of an array can not be changed during execution.
The definition of a DEF statement can not be changed during execution.

Hints on rewriting of special commands of N88-BASIC
[memo]
N88-BASIC is a Microsoft compatible dialect that has the screen of size 640×400.

CLS
CLEAR is a substitute when CLS is used for erasing the graphics screen.

CLEAR, CONSOLE
Commonly, these statement are negligible.
Note that CLEAR has different meaning on Full BASIC.

SCREEN
If you want the drawing pane to be the size of 640×400, replace SCREEN 3 with
SET BITMAP SIZE 640,400
SET WINDOW 0,639,399,0

COLOR
Replace COLOR n with SET TEXT COLOR n.
Replace COLOR ,,,n with SET LINE COLOR n .
Color numbers are 0 for white , 1 for black , 2 for blue , 3 for green , 4 for red , ...

LOCATE
Replace
LOCATE a,b
PRINT ~ ~
with
SET WINDOW 0,80,39,-1
PLOT TEXT, AT a,b: ~ ~

However, if ~ ~ is a numeric expression, replace PRINT ~ ~ with
PLOT TEXT, AT a,b: STR$(~ ~)
and replace PRINT USING "###"; ~ ~ with
PLOT TEXT, AT a,b, USING "###": ~ ~

GET@ , PUT@
Replace
GET@(x1,y1)-(x2,y2), B
. . . .
PUT@(x1,y1), B, PSET
with
OPTION BASE 0
DIM B(639,399)
MAT B=ZER(x2-x1, y2-y1)
ASK PIXEL ARRAY (x1,y1) B
. . . .
MAT PLOT CELLS, IN x1,y1; x2,y2: B

Note.
LOCATE and GET are keywords of Full BASIC, which have different syntax and different meaning.

■ Differences from True BASIC

■ Differences in Behavior

Difference between Decimal BASIC and True BASIC

1. Behavior

(1) The behavior of the following statements are different from True BASIC due to conformity with the standard.

SET WINDOW
Initial aspect of the pane is a square.
If you want to change the aspect of the pane, use SET BITMAP SIZE.

PLOT POINTS
The initial shape of a point is *.
If you want to have compatibility with True BASIC, do

SET POINT STYLE 1
beforehand.

PLOT TEXT
PLOT TEXT may be affected by a transformation.
Use PLOT LABEL instead of PLOT TEXT for compatibility with True BASIC.

GET POINT
The beam is set to be OFF.
If you want to have compatibility with True BASIC, do
SET BEAM MODE "IMMORTAL"
beforehand.

(2) Permissible difference with the standard

Color Indices
The initial color of each color index may be different.

(3) Commands that are not contained in the standard.

PLOT x, y
interpreted as PLOT LINES: x, y

SET COLOR n
changes all of line color, point color, area color, text color.

(4) Accuracy

Some programs that run thanks to computational errors on True BASIC may not work.
For example, if angle option is degrees, 1/SIN(180) or TAN(90) causes an exception on Decimal BASIC.

- **Differencece in Syntax**

Differences between Decimal BASIC and True BASIC

2. Syntax

True BASIC allows the formats on the left-hand side, while Decimal BASIC allows only the formats on the right-hand side.

ELSE IF	→ ELSEIF
GET POINT x, y	→ GET POINT: x, y
A$[1:3]	→ A$(1:3)
WHEN ERROR IN	→ WHEN EXCEPTION IN
CAUSE ERROR	→ CAUSE EXCEPTION
CALL sub(A())	→ CALL sub(A)
DEF ~ END DEF	→ FUNCTION ~ END FUNCTION
DECLARE DEF	→ DECLARE FUNCTION
RESET #n:	→ SET #n: POINTER
ORG STREAM	→ ORGANIZATION STREAM, RECTYPE INTERNAL

Substring modifiers can be applied only to string variables, string expressions except string variables cannot have substring modifiers.
Use a originally enhanced built-in function SUBSTR$(a$,m,n) to get a substring of a string expression.

- **Other differences**

Differences between Decimal BASIC and True BASIC

3. True BASIC original commands that are not available on Decimal BASIC

```
· Control
    library
        MERGE is a substitute
    OPTION TYPO
        Menu Option - Syntax - Compel All Variables Declared
    OPTION NOLET
        Menu Option - Syntax - Microsoft BASIC Compatible
    GET KEY
        Apply ORD-function after CHARACTER INPUT
    KEY INPUT
        use ASK CHARACTER PENDING
    END DATA, MORE DATA
        use READ IF MISSING
    DECLARE PUBLIC
        DECLARE EXTERNAL NUMERIC
        DECLARE EXTERNAL STRING
    PUBLIC
        PUBLIC NUMERIC
```

```
        PUBLIC STRING
    PRIVATE
        SHARE NUMERIC
        SHARE STRING

· Graphics
    SET BACKGROUND COLOR  (SET BACK)
        SET COLOR MIX(0) and CLEAR
    ASK BACKGROUND COLOR  (ASK BACK)
        use ASK COLOR MIX(0)
    BOX LINES
        use GRAPH LINES
        EXTERNAL SUB BOX_LINES(l,r,b,t)
            GRAPH LINES: l,b; r,b; r,t; l,t; l,b
        END SUB
    BOX AREA
        use GRAPH AREA
        EXTERNAL SUB BOX_AREA(l,r,b,t)
           GRAPH AREA: l,b; r,b; r,t; l,t
        END SUB
    BOX CLEAR
        SET AREA COLOR 0 and GRAPH AREA
    BOX CIRCLE , BOX ELLIPSE
        use DRAW CIECLE or DRAW DISK
    BOX KEEP
        use ASK PIXEL ARRAY
    BOX SHOW
        use MAT PLOT CELLS
    GET MOUSE
        use MOUSE POLL
    SET MODE, ASK MODE
        None. ( but graphics can be used always )
        To determine the bitmap size, use SET BITMAP SIZE
    SET CURSOR, ASK CURSOR
        None.
    OPEN SCREEN
        None.
    WINDOW #
        None.

· File I/O
```

```
    ORG RANDOM
        None.
    ORG RECORD
        None.
    Byte file
        OPTION CHARACTER BYTE
        and use CHARACTER INPUT# and PRINT#
    CREATE
        None.(unnecessary)
    END # , MORE #
        use INPUT# IF MISSING THEN
    UNSAVE
        FILE DELETE
    ASK FREE MEMORY
        None.

· Music
    PLAY, SOUND
        None.

· Built-in functions
    CPOS, CPOSR, NCPOS, NCPOSR, POSR
          make external functions such as
          EXTERNAL FUNCTION CPOS(a$,b$,n)
          FOR i=n TO LEN(a$)
             IF POS(b$,a$(i:i))>0 THEN
                LET CPOS=i
                EXIT FUNCTION
             END IF
          NEXT i
          LET CPOS=0
          END FUNCTION
    PEEK, UNPACKB, NUM, NUM$, RUNTIME
        None.

· Built-in subprogram
    DIVIDE
        EXTERNAL SUB DIVIDE(a,b,q,r)
            LET q=INT(a,b)
            LET r=MOD(a,b)
        END SUB
```

Information about this system

- **Command line parameters**

Command-line Parameters

BASIC.EXE is able to have command-line parameters.

When /NI is specified , BASIC.EXE does not re-write BASIC.INI.
When /NR succeeded by a file name is specified, that file is opened without run.
When /OR succeeded by a file name is specified, that program is run.
When a file name is specified without /NR or /OR, BASIC.EXE terminates after the execution of that program.

When two option switches are needed, they must be written in order of /NI /NR (or /NI /OR).
When a file which contains a program with a PROGRAM statement is specified, the program parameters must be written separated by space characters, succeeding the file name.
(Note) Every parameter must be separated by space characters.
(Note) Omission of the extension of a program file name is not allowed.

■ BASIC.INI

BASIC.INI

BASIC.EXE makes a file named BASIC.INI , which records some settings of this system, on finishing.
BASIC.INI is made on the folder BASIC.EXE has been launched.

BASIC.INI can be edited using NOTPAD.EXE

[Setup]
The following key can be added to [Setup] section.
Registry

[Frame]
The following keys can be added to [Frame] section.
IniFileReadOnly
NoRun
NoBackUp
BasExt
LibExt
InitialDir
CharacterByte
BreakKey

VirtualMemory

[Graphics]
The following key can be added to [Graphics] section.
AxisColor

(Example)
[Setup]
Registry=1
[Frame]
IniFileReadOnly=1
NoRun=1
NoBackUp=1
BasExt=.BAS
LibExt=.LIB
InitialDir=c:\BASICw32
CharacterByte=1
BreakKey=C
VirtualMemory=256
[Graphics]
AxisColor=15

(Meaning)
○ Registry=1
BASIC.EXE use the registry. In that case, the other keys are invalid.
○ IniFileReadOnly=1
BASIC.EXE shall not re-write BASIC.INI.
○ NoRun=1
When BASIC.EXE is launched with a file name, BASIC.EXE do not run the program.
○ NoBackUp=1
When a program starts, no back-up file is made.
○ BasExt
The file extension for the programs. This key starts with a period. At most 7 characters including the period.
○ LibExt
The file extension for the library files. This key starts with a period. At most 7 characters including the period.
○ InitialDir
The folder which the File-Open dialog displays first.
○ CharacterByte=0
supports multi-byte character sets.

○ BreakKey=C
makes the short-cut key for the break command Ctrl-C.
○ VirtualMemory=256
reserves 256MB for Virtual Stack Memory.
○ AxisColor
The color index that is used by built-in pictures AXES and GRID.

< Memo >
When SETUP.BAT is performed, a key 'Registry=1' is added to [Setup] section.
If you edit BASIC.INI so that Registry=0, use of registry can be canceled.

BASIC.BAK

BASIC.BAK

A program is saved in a file BASIC.BAK at the start of execution.
BASIC.BAK is removed at the end of execution.
BASIC.BAK is made in the folder where BASIC.EXE lies.
BASIC.BAK is a simple text file.

When BASIC.EXE terminated abnormally, please inform us of the incident with BASIC.BAK attached.

Mutliple execution of BASIC.EXE

Multiple Execution of BASIC.EXE

Multiple Execution of BASIC.EXE is not prohibited.
Note that BASIC.BAK and BASIC.INI are unique.

Customize the words for case changing

Customizing the word for case change

Changing the character types in the Edit menu, BASIC.KW1, BASIC.KW2 is based on the registered string. If you want to change the language you want to change the scope of the casing as a function word, please change the contents of these files.

BASIC.KW1 except the function name, BASIC.KW2 is a list of function names. It is a text file, Windows can open and edit such as Notepad. Please register always in capital letters.

■ Customize the keywords on the Keyword catalog

Customizing the Insert function words

Function word is inserted in the Edit menu, BASIC.KWS, BASIC.KWF is based on the data registered. Rewrite the contents of these files, you can change your window into the function words.
The statement BASIC.KWS, BASIC.KWF holds data about the function.
These data are stored in comma delimited format.
The first column is the function words. "!" Refers to the position of the cursor is moved. "-" Is newline.
The second column is the description that appears below the bottom of the window insert function words.
The third column is the Help context number. This is the page number to identify the help screen. When in doubt, please specify 0.

■ Other information

More Information

1. This program is created using Borland Delphi 7.
Some of the features in "Delphi3 Q & A select 150 (Author Oono Motohisa, Village Center)," to use a component of the Journal.
GIF image loading, saving, Anders Melander's TGIFIMAGE to use.
HTMLHelp to work with, The Helpware Group of Delphi HTML Help Kit to use.
By Dr. T. Nishimura and Makoto Matsumoto has Mersenne Twister random number generation algorithm method.
Show mt19937ar.c the original copyright

2. Exceptions to the use of floating-point exception handling in Windows.
PC9800 Windows95 version of the OS may need to be modified.

3. CPU bugs are not supported.
(1) Pentium floating point division bug affected.
(2) Pentium Pro, Pentium bug that affected □ flag is probably not the exception.

<Memo>
(Tentative) Decimal BASIC is introduced in the internal structure of the paper:.
"Windows on the implementation of JIS Full BASIC environment," Shiraishi Kazuo, Journal of Information Processing: Programming Vol.41 No.SIG 9 (PRO 8), pp. 52-61 (November 2000)

Trouble shooting

Freezing on step execution or break

"Step" or "break" you choose to freeze

Models equipped with certain video cards, the "step" or "abort" command is selected the moment, OS may freeze each. The "insert function words" you choose, BASIC.EXE may be terminated unexpectedly. These are not working properly because the video card drivers. Windows Control Panel
System - Performance - Graphics
Select, please tick one of the hardware down.
Trouble is, your video card STB Velocity 3D models are generated to adopt. 3dfx's site (http://www.stb.com/drivers/) can now be used in a state of maximum performance when you update to get the latest version of the driver. The syngeneic Akuser-aretachippu (S3 Virge) may experience problems similar models that use.

Strange toolbar icons

Icons do not appear correctly on the toolbar

If you see an icon on the toolbar collapsed, please try the following two measures.

(1) Windows, and once started in safe mode, restart again.

(2) Windows Control Panel
System - Performance - Graphics
Choose a lower scale than the two accelerators.

(1) and fix it (2) is unnecessary.
(2) To work correctly, please try updating to the latest video card drivers.

Blue screen on NEC PC9899 version of Windows95

NEC PC98 Blue screen version in Windows95

NEC PC9800 Windows95 version, and binary modes, such as zero divide, and if you get a blue screen failure occurs when a math coprocessor exception is the OS must be modified. Please refer to PC9800.TXT measures.
However, OS is when it is difficult to fix, ver. 4.4.23 Please use.

RichEdit line insertion error

RichEdit line insertion error in

Is RichEdit line insertion error to fail to launch in BASIC.EXE is, Windows could be a bug in the RichEdit control. Please refer to the following site to repair Windows.
http://support.microsoft.com/default.aspx?scid=kb; ja; JP412067

Trouble on 3-mode FDD

3 FDD mode troubles
When you step to freeze programs that manipulate files

3mode FDD models incorporating non-Microsoft driver, 1.2Mbyte FD When you step on a program to manipulate files, OS may freeze each.
This behavior of the Japanese models made in Japan GATEWAY2000 GATEWAY2000 3mode FD distribution occurred when the company built the driver. 3mode Windows95 CD-ROM drivers included with those of Microsoft (EPSON for the machine) will work correctly when you change.

■ Reduction in precision

And precision of

Of BASIC, with built-in functions such as counting in decimal mode, even using the FPU operation. Therefore, problems may occur as follows.

1. CPU manufacturer differences
Intel MMX for Pentium and AMD K6 in case if there is a difference for the calculated result. The same manufacturers that belong to different generations of CPU if there is a difference in the know.

2. Interference device driver
Some, such as device drivers and mouse drivers Video card drivers, FPU to change the flag, there seems to be something it back. Subject to interference and their results or error occurs, the trouble is expected or when an exception condition Naranakatsu be an exception. So far, this is not true failure is reported, people who have encountered a phenomenon that might be applicable, the report is a thank you for your time.

■ Patch patterns on PLOT AREA

The patchy pattern PLOT AREA

Ver. 4.7.7 to run from the PLOT AREA changed to use Windows API. Therefore, some models might be part of the neutral spotted pattern. If so, please try the following measures.
(1) changing the color depth (High Color True Color from to).
(2) second from left down to the scale of the hardware control panel.
(3) Update to the latest video card drivers.

Illegal change on program texts

Improperly rewritten the program text

BASIC program to edit this text uses a Windows RichEdit control. EichEdit control function associated with the OS, Microsoft Office and, Visual BASIC version will change even if you install an application written in.
RichEdit control in certain versions of pulling a bug, and will run the program from 32KB to 64KB program rewritten in a certain position between the (Windows 2000, Windows Me + Office 2000 a lot.)
Also, Windows XP, the problem has been reported disturbed break is broken fix cursor position and repeat using the clipboard (the same happens in Notepad.)
If you are thank you encounter these symptoms reported with the following content.
Names and Windows service packs (for example, Windows2000 SP2)
Version of Internet Explorer (eg IE 5.00.2614.3500)
When MS Word or MS Office is installed, the number of its name and service or service pack release (for example Ofiice2000 SR-1)
BASIC help books - and display the version information RICHED32.DLL RICHED20.DLL version number (cases 4.0.450.28 / 5.30.23.1200)

Poor redraw of the graphics window

Abnormality of the graphic window redraw

After some other overlapping windows graphics display window, when you move or close that window when no image is redrawn, Windows Control Panel "Display Properties" to Select "Settings" "Advanced" in "Troubleshooting" section, select "Enable Write Combining" Please remove the check (menu structure of Windows a bit depending on the version, there is a difference).
"Display Properties", then right click on the background portion of the screen "Properties" will appear in the select.

You turn off Windows Aero Windows Vista's bad draw

In that Winodows Vista Aero is disabled, a pane on the screen does not reflect PLOT POINTS drawing.
This problem occurs when the "Options" menu "compatibility" with "paint", select "Draw on the screen," the "Vista Non-Aero fault tolerance," Please check.

Other troubles

Other problems

BASIC.EXE or is killed by a Windows, Windows freezes when you fall to freeze itself may also display driver and the failure of the device. BASIC.EXE killed when the name of the module the error occurred more click (for example, STBV3D.VXD, GDI.EXE etc.) Please log.
However, Windows95/98/2000/XP in the Control Panel "Display Properties" in "Troubleshooting" will tell you if there is reason to look down and video card drivers on the scale of the hardware. If it works correctly, so it could very well have a problem with the video card drivers.
The above problems is as valuable information to share. If you experience any problems, but was sorry to trouble you, thank you must report.

Known faults

Known Bugs

1. Microsoft BASIC that offers the ability to run programs written by rewriting the grammar,
IF A = 1 THEN ELSE A = 2: B = 1
THEN, as if the empty statements between ELSE and can not be modified correctly.

2. If the display 256 colors, the colors can not be handled correctly.

3. Windows2000 abnormally slow or cause the screen to a large number of Chinese characters and Chinese characters written in a program in mass.
This problem can be avoided by using the input and output data files.

4. In Windows2000, shift F5 ~ F8 can not specify a trailing space to be inserted in the text.
This problem, SETUP.BAT resolves to save the settings you can run the registry.

Other bugs you find a bug report, please make sure this time slot.

Decimal BASIC Reserved Words

- **A**

ABS
ACCESS
ACOS
AND
ANGLE
AREA COLOR
ARG
ARITHMETRIC DECIMAL
ARITHMETRIC
ASC
ASIN
ASK AREA COLOR
ASK AXIS COLOR
ASK BITMAP SIZE
ASK CHARACTER PENDING
ASK COLOR MIX
ASK DIRECTORY
ASK LINE COLOR
ASK STYLE
ASK MAX COLOR
ASK MAX LINE STYLE
ASK MAX POINT STYLE
ASK PIXEL ARRAY
ASK PIXEL SIZE
ASK PIXEL VALUE
ASK POINT COLOR
ASK POINT STYLE
ASK TEXT COLOR
ASK TEXT HEIGHT
ASK TEXT JUSTIFY
ASK TEXT WIDTH
ASK WINDOW
ASSIGN
ASSOC PRINT
ATN
AXES
AXES0

B

BACKGROUND
BASE
BEEP
BEZIER
BITMAP
BLEN
BOX
BREAK
BSTR$
BVAL
BYTE

C

CALL
CALLBACK
CALLBACKADR
CASE
CASE ELESE
CAUSE
CEIL
CELLS
CHAIN
CHARACTER INPUT
CHARACTER INPUT #
CHARACTER PENDING
CHOICE
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
COLOR
COLOR MIX
COLOR MODE
COLORINDEX
COM1
COM2
COMB
COMPLEX
CON
CONJ
CONSOLE
CONTINUE
COS
COSH
COT
CREATEOBJECT
CSC
CSV

D

DATA
DATE
DATE$
DEBUG
DECIMAL
DECLARE EXTERNAL FUNCTION
DECLARE EXTERNAL PICTURE
DECLARE EXTERNAL SUB
DECLARE NUMERIC
DECLARE STRING
DEF
DEFDBL
DEFINT
DEG
DEGREES
DELAY
DELETE
DENOM
DET
DEVICE
DIM
DIRECTORY
DISK
DO
DOT
DRAW AXES
DRAW GRID
DRAW MODE

E

ECHO
ELAPSED
ELSE
ELSEIF
END
END FUNCTION
END IF
END PICTURE
END SELECT
END SUB
END WHEN
ENDOFLINE
EOF
EPS
EPS
ERASE
EXEPTION
EXECUTE
EXIT
EXIT DO
EXIT FOR
EXIT FUNCTION
EXIT HANDLER
EXIT SUB
EXLINE
EXP
EXTERNAL FUNCTION
EXTERNAL PICTURE
EXTERNAL SUB
EXTYPE

F

FACT
FILE
FILE DELETE
FILE GETNAME
FILE GETNAME
FILE SPLITNAME
FIX
FLOOD
FOR
FOR APPEND
FOR APPEND
FOR INPUT
FOR OUTPUT
FP
FUNCTION

G

GCD
GET
GET POINT
GET@
GETKEYSTATE
GETNAME
GLOAD
GOSUB
GOTO
GRAPH
GRID
GRID0
GSAVE

H

HALT
HANDLER
HEIGHT
HEX$
HIDDEN

I

IDN
IF
IM
IMAGE
INKEY$
INPUT
INPUT #
INPUT ELAPSED
INPUT PROMPT
INPUT TIMEOUT
INSTR
INT
INTERNAL
INTSQR
INV
IP

J

JUSTIFY

K

KANJI
KEY
KILL

L

LABEL
LBOUND
LCASE$
LEFT$
LEN
LET
LIBRARY
LIMIT
LINE
LINE COLOR
LINE INPUT #
LINE STYLE
LOCAL
LOCATE
LOG
LOG10
LOG2
LOOP
LPRINT
LTRIM$

M

MARGIN
MAT
MAT GRAPH CELLS
MAT INPUT
MAT PLOT
MAT PLOT AREA
MAT PLOT CELLS
MAT PLOT LINES
MAT PLOT POINTS
MAT PRINT
MAT PRINT USING
MAT READ
MAT REDIM
MAX
MAXNUM
MAXSIZE
MERGE
MID$
MIN
MIX
MOD
MODULE
MORE
MOUSE POL

N

NAME
NATIVE
NEXT
NOBEAMOFF
NOT
NOTXOR
NOWAIT
NUL$
NUMER

O

OLE
ON ERROR GOTO
OPEN
OPTION ANGLE DEGREES
OPTION ANGLE RADIANS
OPTION ARITHMETIC COMPEX
OPTION ARITHMETIC DECIMAL
OPTION ARITHMETIC NATIVE
OPTION ARITHMETIC RATIONAL
OPTION BASE
OPTION CHARACTER
OR
ORD
ORGANIZATION
OUTIN
OUTPUT
OVERWRITE

P

PACKDBL$
PAINT
PAUSE
PENDING
PERM
PI
PICTURE
PIXEL
PIXEL ARRAY
PIXEL SIZE
PIXEL VALUE
PIXELX
PIXELY
PLAY
PLOT
PLOT AREA
PLOT LINES
PLOT POINTS
PLOT TEXT
POINT
POINT COLOR
POINT STYLE
POINTER BEGIN
POINTER END
POLL
POS
PRINT
PRINT #
PRINT USING
PRINTER
PRIVATE
PROBLEMX
PROBLEMY
PROGRAM
PROMPT
PROPERTYGET
PROPERTYPUT
PSET
PUBLIC
PUT@

R

RAD
RADIANS
RANDOMIZE
RATIONAL
RE
READ
READ IF MISSING
RECTYPE
REDIM
REM [or !]
REMAINDER
RENAME
REPEAT$
RESTORE
RESUME
RETRY
RETURN
RIGHT$
RND
ROTATE
ROUND
RTRIM$

S

SCALE
SCREEN
SEC
SELECT CASE
SET AREA COLOR
SET BITMAP SIZE
SET COLOR MIX
SET DEVICE WINDOW
SET DEVICE VIEWPORT
SET DIRECTORY
SET ECHO
SET LINE COLOR
SET LINE STYLE
SET MARGIN
SET POINT COLOR
SET POINT STYLE
SET TEXT COLOR
SET TEXT HEIGHT
SET TEXT JUSTIFY
SET VIEWPORT
SET WINDOW
SET ZONEWIDTH
SGN
SHARE
SHEAR
SHIFT
SIGNAL
SIN
SINH
SIZE
SPACE$
SPC
SPLITNAME
SQR
STEP
STOP
STR$
STREAM
STRING
STRING$
SUB
SUBSTR$
SWAP

T

TAB
TAN
TANH
TEXT
TEXT COLOR
TEXT HEIGHT
TEXT JUSTIFY
TEXTWINDOW1
TEXTWINDOW2
THEN
TIME
TIME$
TIMEOUT
TO
TRACE
TRANSFORM
TRN
TRUNCATE

U

UBOUND
UCASE$
UNPACKDBL
UNSAVE
UNTIL
USE
USING
USING$

V

VAL
VALUE
VIEWPORT

W

WAIT
WAIT DELAY
WAIT SIGNAL
WEND
WHEN
WHEN EXEPTION IN
WHEN EXEPTION USE
WHILE
WINDOW
WINHANDLE
WITH
WORLDX
WORLDY
WRITE

Z

ZER
ZONEWIDTH

Full BASIC to *Mathematica* Conversion

■ Numerical

ABS(x)	Abs
ACOS(x)	ArcCos
ANGLE(x,y)	
ASIN(x)	ArcSin
ATN(x)	ArcTan
CEIL(x)	Ceiling
COS(x)	Cos
COSH(x)	Cosh
COT(x)	Coth
CSC(x)	Csch
DATE	DateList
DEG(x)	Degree
EPS(x)	$MachineEpsilon; $MinNumber
EXP(x)	Exp
FP(x)	FractionalPart
INT(x)	Integer
IP(x)	IntegerPart
LOG(x)	Log
LOG10(x)	Log10
LOG2(x)	Log2
MAX(x,y)	Max
MAXNUM	$MaxNumber
MIN(x,y)	Min
MOD(x,y)	Mod
PI	Pi
RAD(x)	Radian
REMAINDER(x,y)	QuotientRemainder
RND	Random
ROUND(x,n)	Round
SEC(x)	Sec
SGN(x)	Sign
SIN(x)	Sin
SINH(x)	Sinh
SQR(x)	Sqrt
TAN(x)	Tan
TANH(x)	Tanh
TIME	DateString
TRUNCATE(x,n)	N
OPTION ANGLE DEGREES	Degree
OPTION ANGLE RADIANS	(Radian)

Strings

CHR$(x)	ToCharacterCode
DATE$	DateList
LCASE$(x$)	ToLowerCase
LEN(x$)	StringLength
ORD(x$)	FromCharacterCode
POS(x$,y$)	StringPosition
REPEAT$(x$,y)	Do
RTRIM$(x$,y)	StringDrop[x,-y]
STR$(x)	ToString
TIME$	DateList
UCASE$(x$)	ToUpperCase
USING$(x$,y)	StringForm
VAL(x$)	ToExpression
x$(a:b)	x[[Range[a,b]]]

Arrarys and matrices

MAT A=B	Array or Table
MAT A=B+C	Array or Table
MAT A=B-C	Array or Table
MAT A=B*C	Array, KroneckerProduct
MAT A=x*B	Array or Table
MAT A=ZER	Array[a]=0
MAT A=IDN	IdentityMatrix
MAT A=CON	ConstantArray
MAT A=INV(B)	Inverse
MAT A=TRN(B)	Transpose
DET(A)	Det
DOT(A,B)	Dot [or "."]
LBOUND(M,i)	
UBOUND(M,i)	
SIZEM(M,i)	ArrayDepth
SIZE(M)	Dimensions
MAXSIZE(M)	

■ Control Structures

FOR x=a TO b STEP c	For[x=z,x<=b,x=+c, *body*]
NEXT x	
EXIT FOR	Break
DO	Do
DO UNTIL	Do
DO WHILE	While
LOOP UNTIL	Break
LOOP WHILE	Break
EXIT DO	Break
RETURN	Return, Continue, Abort
GOTO	Goto
ON GOTO	Goto, Label
ON GOSUB	Goto, Label

■ Program Segmentation

DEF	f[x_]:=
END	
FUNCTION	f[x_]:=
EXTERNAL FUNCTION	
END FUNCTION	
SUB	Label
EXTERNAL SUB	
END SUB	
PICTURE	Label
EXTERNAL PICTURE	
END PICTURE	
CALL	
DRAW	Plot; Graphics
WITH	
ROTATE	
SHIFT	
SHEAR	
SCALE	
DECLARE EXTERNAL	
MODULE	Module; DynamicModule
END MODULE	
PUBLIC	
SHARE	
MODULE OPTION	

Input and Output

INPUT	Input
LINE INPUT	
READ	Read
READ IF MISSING	
PRINT	Print
PRINT USING	StringForm; ItemSize
MAT READ	
MAT INPUT	
MAT PRINT	MatrixForm
CHARACTER INPUT	InputField
ASK CHARACTER PENDING	

Files

OPEN	OpenRead
CLOSE	Close
ERASE	DeleteFile
WRITE	OpenWrite
READ	Get
SET MARGIN	
SET ZONEWIDTH	
SET POINTER	
ASK MARGIN	
ASK ZONEWIDTH	
ASK POINTER	
ASK NAME	
ASK ACCESS	
ASK RECTYPE	
ASK ORGANIZATION	

Exception Handling

WHEN EXCEPTION IN	
USE	
END WHEN	
HANDLER	EventHandler
END HANDLER	
CAUSE EXCEPTION	Abort
EXIT HANDLER	
CONTINUE	Continue
RETRY	
EXTYPE	
EXLINE	

Debugging

DEBUG ON	Trace
DEBUG OFF	
TRACE ON	Trace
TRACE OFF	
BREAK	Break

■ Graphics

SET WINDOW	CreateDocument; CreateDocument; WindowSize
SET VIEWPORT	ImageSize; PlotRange
SET DEVICE WINDOW	Graphics; GraphicsArray; Inset
SET DEVICE VIEWPORT	ItemSize; PlotRegion; SphericalRegion
SET LINE COLOR	CMYKColor, RGBColor
SET POINT COLOR	CMYKColor, RGBColor
SET AREA COLOR	CMYKColor, RGBColor
SET TEXT COLOR	Syle, Cell; FontColor
SET COLOR MIX	CMYKColor, RGBColor
SET LINE STYLE	Line; Dashing
SET POINT STYLE	Point; CMYKColor, GrayLevel, Hue, Opacity, RGBColor
SET TEXT JUSTIFY	
SET TEXT HEIGHT	Style;FontSize
SET TEXT ANGLE	
ASK ...	
PLOT LINES	Graphics; Line; Dashing, Thickness
PLOT POINTS	Point
PLOT AREA	
PLOT TEXT	Graphics[Text[a,{x,y}]
GET POINT	Locator
MAT PLOT LINES	ArrayPlot
MAT PLOT POINTS	ArrayPlot
MAT PLOT AREA	
MAT PLOT CELLS	
ASK PIXEL VALUE	
ASK PIXEL ARRAY	
ASK PIXEL SIZE	
CLEAR	
LOCATE CHOICE	
LOCATE VALUE	
MAT GET POINT	Slider2D
SET AREA STYLE	
SET AREA STYLE INDEX	
ASK AREA STYLE	
ASK AREA STYLE INDEX	

Added by *Decimal BASIC*

- **for *Complex Number Mode***

COMPLEX(x,y)	Complex
RE(x)	Re
IM(x)	Im
ARG(x)	Arg
CONJ(x)	Conjugate

- **for *Rational Mode***

INTSQR(x)	
NUMER(x)	
DENOM(x)	
GCD(x,y)	GCD

- ***Arithmetic Functions***

FACT(x)	Factorial
PERM(n,r)	Permutations
COMB(n,r)	Tuples
ROUND(x)	Round

- ***String Functions***

SUBSTR$(a$,m,n)	StringTake[a,{m,n}]
MID$(a$,m,n)	StringTake[a,{m,n}]
LEFT$(a$,n)	StringTake[a,n]
RIGHT$(a$,n)	StringTake[a,-n]

- ***Statements***

MAT REDIM
LOCAL

- ***Matrix***

MAT C=CROSS(A,B)	CrossProduct

Graphics

predefined PICTURES

DRAW GRID	Graphics; GridLines->True
DRAW GRID(p,q)	Graphics; GridLines->{p,q}
DRAW AXES	Graphics; Axes->True
DRAW AXES(p,q)	Graphics; Axes->{p,q}
DRAW GRID0	Gaphics; GridLines->None
DRAW AXES0	Graphics; AxesOrigin->Automatic
DRAW CIRCLE	Graphics; Circle
DRAW DISK	Graphics; Disk

new graphic commands

PLOT LABEL ,AT x,y: a$	Text[a,{x,y}]
MOUSE POLL x,y,left,right	MousePosition
SET BEAM MODE "IMMORTAL"	
SET BEAM MODE "RIGOROUS"	
SET TEXT FONT name$,points	Style
SET BITMAP SIZE x,y	ImageResolution; Rasterize, ImageSize
FLOOD x,y	Disk; Polygon; Cuboid; Rectangle
PAINT x,y	Color
SET COLOR MODE "NATIVE"	
SET COLOR MODE "REGULAR"	
SET DRAW MODE HIDDEN	
SET DRAW MODE EXPLICIT	
SET DRAW MODE NOTXOR	
SET DRAW MODE OVERWRITE	
SET AREA STYLE "SOLID"	
SET AREA STYLE "HOLLOW"	
SET AREA STYLE "HATCH"	
SET AREA STYLE INDEX x [1-6]	
PIXELX(x)	
PIXELY(y)	
PROBLEMX(x)	
PROBLEMY(y)	
COLORINDEX(r,g,b)	RGBColor

Real-time

WAIT DELAY x	Pause
BVAL(x$,16)	
BSTR$(x,16)	