



PI-1010/PI-1030

# Basic Programming Manual

**Version:1.01**

Copyright © 2016 by ARGOX Information Co., Ltd.

<http://www.argox.com>

# Preface

To satisfy the user's customization needs, PI-1010 / PI-1030 Basic provides effective approaches for users to generate programs right to their actual demands. This allows users to collect data, execute data processing, then store the processed data into proper location for future use.

PI-1010 / PI-1030 Basic interpreter provides a platform for users to develop application programs to be executed on the PI1 series data terminals using BASIC language. Users can develop an application to meet their own individual needs efficiently.

You'll soon learn how to use BASIC language to write application programs. Please proceed and enjoy the perfect combination of PI-1010 / PI-1030 Basic and PI1 series and the productivity they can boost for you in your application.

---

# Table of Contents

<b>Preface.....</b>	<b>1</b>
<b>Table of Contents.....</b>	<b>2</b>
<b>1 How to run BASIC program.....</b>	<b>5</b>
1.1 User Menu .....	5
1.1.1 Run program .....	5
1.1.2 Remote Link.....	5
1.1.3 Information .....	6
<b>2 Program Structure.....</b>	<b>7</b>
2.1 Constants .....	7
2.1.1 String.....	7
2.1.2 Numeric .....	7
2.2 Variables .....	7
2.2.1 Variable Names and Declaration Characters.....	8
2.2.2 Array Variables .....	8
2.3 Expression and Operators .....	8
2.3.1 Assignment Operator.....	9
2.3.2 Arithmetic Operator .....	9
2.3.3 Relational Operator .....	9
2.3.4 Logical Operator .....	9
2.4 Operator Precedence .....	10
2.5 Labels .....	10
2.6 Subroutines .....	11
2.7 Exit program .....	12
2.8 Special notes .....	12
<b>3 Command Sets .....</b>	<b>13</b>
3.1 General commands .....	13
3.2 Commands for decision structures .....	17
3.3 Commands for looping structures .....	20
3.4 Commands for string processing .....	22
3.5 Commands for event trapping .....	28
3.6 System commands .....	38
<b>3.7 Reader commands.....</b>	<b>42</b>
3.8 Beeper commands.....	48
3.9 Calendar and timer commands.....	50

---

3.10	LED Command .....	52
3.11	Keypad commands.....	53
3.12	LCD Commands .....	60
3.13	Font.....	64
3.13.1	User font commands .....	64
3.14	TextBlock.....	66
3.14.1	TextBlock commands .....	67
3.15	File manipulation commands .....	71
3.15.1	Standard Commands .....	71
3.15.2	DBMS Commands .....	79
3.16	Vibrator commands.....	84
3.17	Communication port commands .....	85
3.18	Memory commands .....	96
3.19	USB commands .....	97
3.20	LinkingPort commands .....	98
3.21	Simulator (Only for PC simulator) commands .....	107
<b>4</b>	<b>Appendices .....</b>	<b>108</b>
	Appendix A .....	108
	<b>PT-Basic Commands list .....</b>	<b>108</b>
A1.	General commands.....	108
A2.	Commands for decision structures .....	108
A3.	Commands for looping structures.....	109
A4.	Commands for string processing .....	110
A5.	Commands for event trapping .....	111
A6.	System commands.....	111
A7.	Reader commands .....	112
A8.	Buzzer commands .....	112
A9.	Calendar and timer commands .....	112
A10.	LED command .....	113
A11.	Keypad commands.....	113
A12.	LCD Commands .....	114
A13.	User font commands .....	114
A15.	File manipulation commands .....	116
A16.	Vibrator commands .....	117
A17.	Communication port commands.....	117
A18.	Memory commands.....	117
A19.	USB commands .....	118
A20.	LinkingPort commands .....	118

---

A21. Simulator (Only for PC simulator) commands .....	118
<b>Appendix B .....</b>	<b>119</b>
<b>Scan Module (CCD) Configuration Table.....</b>	<b>119</b>

---

# 1 How to run BASIC program

## 1.1 User Menu

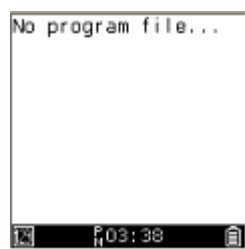


If you have already downloaded FW file, then you can view the User Menu by pressing the power key.

### 1.1.1 Run program

If the BASIC program file (xxx.bas) in the direct path (D:\\Program\\) then you can run the BASIC program now.

If the BASIC program file (xxx.bas) is not in the direct path (D:\\Program\\) then the following message will prompt you.



### 1.1.2 Remote Link

You can use this item to download program file or download/upload other files.

---

### 1.1.3 Information

You can use this item to get version information of all software and firmware parts of the system.



---

## 2 Program Structure

### 2.1 Constants

Constants are the actual values used or generated in the program. There are two types of constants:

#### 2.1.1 String

A string constant is a sequence of up to 255 alphanumeric characters or symbols enclosed in a pair of double quotation marks.

→ "BASIC"

→ "2007.05.13"

→ "ArgoBasic program guide"

→ "168 IbB....."

→ "IbB 168 .....!"

#### 2.1.2 Numeric

Numeric constants include positive and negative numbers.

Numeric constants in BASIC cannot contain commas. There are two types of numeric constants that can be used in the PT-Basic interpreter.

Integer constants:  $-2147483648 \sim +2147483647$

Real number constants: Positive or negative real number, that contain a decimal point, such as 1.23 or  $-3.5897$

### 2.2 Variables

Variable are symbols used to represent data items, such as numerical values or character strings that are used in BASIC program. The value of a variable may be assigned explicitly and can be changed during the execution of the program. Value of a variable is assumed to be undefined until a value is assigned to it.



---

### 2.2.1 Variable Names and Declaration Characters

The following are the rules to declare variable names and characters:

- A variable name must be begun with a letter.
- The remaining characters can be letters, numbers, or underscores.
- The last character can be one of these declaration characters:
  - % (Integer) : 4 bytes (- 2147483648 to 2147483647)
  - ! (Real number) : 8 bytes
  - \$ (String) : 255 bytes
- Variable name cannot be any BASIC reserved words.
- Only 3 types of variable are supported.
- Variable names are case ( upper or lower case ) dependent.

### 2.2.2 Array Variables

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression.

Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression. In PT-Basic, the maximum number of dimensions for an array is 2.

**For example:**

- A\$(8) 'one dimension array
- Str%(2,5) 'two dimension array
- DIM A%(23) 'declares an integer array with 23 elements.
- DIM Str\$(60) 'declares a string array with 60 elements.

## 2.3 Expression and Operators

An expression may be a string or numeric constant, or a variable, or it may be a combination of constants and variables with operators to produce a string value.

Operators perform mathematical or logical operations.

---

### 2.3.1 Assignment Operator

PT-Basic interpreter supports an assignment operator “=”

**For example:**

→Size% =100

→PI! =3.1415

→Str1\$=”back”

### 2.3.2 Arithmetic Operator

The arithmetic operators are:

Operator	Operation	Example
^	Exponentiation	A% = 9^6
-	Negation	A% = -B%
*	Multiplication	A% = B% * C%
/	Division	A% = B% / C%
+	Addition	A% = B% + C%
-	Subtraction	A% = B% - C%
MOD	Modulo arithmetic	A% = B% MOD C%

### 2.3.3 Relational Operator

Relational operators are used to compare two values. Result of the comparison is either “True” or “False”.

Operator	Operation	Example
=	Equality	A% = B%
<>	Inequality	A% <> B%
>	Greater than	A% > B%
<	Less than	A% < B%
>=	Greater than or equal to	A% >= B%
<=	Less than or equal to	A% <= B%

### 2.3.4 Logical Operator

Logical operators perform tests on multiple relations and Boolean operations. Logical operator returns a result which is either “True” (not zero) or “False” (zero). In an expression, logical operations are performed after arithmetic and relational operations.

---

Operator	Operation	Example
NOT	Logical negation	NOT (A% = B%)
AND	Logical and	(A% = B%) AND (C% = D%)
OR	Inclusive or	(A% = B%) OR (C% = D%)
XOR	Exclusive or	(A% = B%) XOR (C% = D%)

## 2.4 Operator Precedence

The precedence of BASIC operators affects the evaluation of operands in expressions. Expressions with higher precedence operators are evaluated first. Precedence of BASIC operators is listed below in the order of precedence from highest to lowest.

Order of Precedence	Type of Operation	symbol
Highest	Arithmetic	^
↓	Arithmetic	*, /, MOD
↓	Arithmetic	+, -
↓	Relational	=, <>, >, <, >=, <=
↓	Logical	NOT, AND, OR, XOR
Lowest	Assignment	=

## 2.5 Labels

Line labels are used to represent some special lines in the BASIC program. They can be either integer numbers or character strings.

- A valid integer number for the line label is in the range from 1 to 65279.
- A character string label can have up to 255 characters (if the string label has more than 255 characters, error can be it cannot be anticipated).

A character string label that precedes a program line must have a colon between the label and the program line, but it is not necessary for an integer label.

---

**For example:**

```
GOTO 100
...
100
...
  GOTO LABEL2
...
LABEL2:
...
```

## 2.6 Subroutines

A subroutine is a set of instructions with a particular name or a line label. User can simplify their programming by breaking programs into subroutines. A subroutine will be executed when being called by a GOSUB command.

**For example:**

```
ON COM (1) GOSUB ReadCOM
...
ReadCOM:
...
  RETURN
```

The command RETURN marks the end of the subroutine and tells the processor to return to the caller. A subroutine has to be appended at the end of the main BASIC program. A subroutine can be defined with or without a pair of brackets.

For example:

```
GOSUB FUN
GOSUB Place
GOSUB Test
END
...
SUB FUN( )
  PRINT "Run function!!"
END SUB
```

---

Place:

```
PRINT "Run Place!!"
```

```
RETURN
```

```
SUB Test
```

```
PRINT "TEST..."
```

```
END SUB
```

## 2.7 Exit program

- **In any place of the program, you can use “END” to exit the program. The system will go to BASIC Menu.**

```
PRINT "Press key to exit!"
```

```
WHILE INKEY$ = ""
```

```
WEND
```

```
END
```

## 2.8 Special notes

- **Commands have to be appeared in uppercase letters**

```
PRINT "OK..." → right
```

```
print "NG..." → error
```

- **Variable names are case sensitive.**

```
ABC% 、 ABc% 、 AbC% → Three kind of different variables
```

```
ARGO% 、 ARGO! 、 ARGO$ → Three kind of different variables
```

---

## 3 Command Sets

### 3.1 General commands

#### ABS

Purpose : To return the absolute value of a numeric expression.

Syntax :  $A\% = ABS(N\%)$  *or*  $A\% = ABS(N!)$

Example : Num1% = 2.89

Num2% = 9.55

Difference% = ABS (Num1% - Num2%)

Description :  $A\%$  is numeric variable to be assigned to the absolute value of a numeric expression.

$N\%$  *or*  $N!$  is a numeric expression, it can be an integer or a real number.

#### DIM

Purpose : To specify the maximum value of variable subscripts and to allocate storage accordingly.

Syntax : ***DIM Array (range {,range}) {, Array(range {,range})}***

Example : DIM A%(8), B%(5,5), C\$(6)

Description : ***Array*** is an array variable.

***Range*** can be an integer or an integer expression.

---

## GOSUB

Purpose : To call a specified subroutine.

Syntax : ***GOSUB SubName/SubLabel/SubNumber***

Example :  
GOSUB FUN  
GOSUB Place  
GOSUB 100  
END  
SUB FUN()  
    PRINT "Run SUBNAME"  
END SUB  
Place:  
    PRINT "Run SUBLABEL"  
    RETURN  
100  
    PRINT "RunSUBNUMBER"  
    RETURN

Description : ***SubName*** is the name of a subroutine.

***SubLabel*** is the line label of a subroutine.

***SubNumber*** is the line number of a subroutine.

## GOTO

Purpose : To branch unconditionally to a specified line number or line label from the normal program sequence.

Syntax : ***GOTO LineNumber/LineLabel***

Example :  
GOTO FUN  
100  
    PRINT "NUMBER"  
    WHILE INKEY\$=""  
    WEND  
    END  
FUN:  
    PRINT "LABEL NAME"  
    GOTO 100

Description : ***LineNumber*** is the integer number in front of a program line.

***LineLabel*** is the string label of a program line.

---

## INT

Purpose : To return the largest integer that is less than or equal to the given numeric expression.

Syntax : ***A% = INT(N%) or A% = INT(N!)***

Example : A% = INT(9.86)

PRINT A%

B% = INT(-5.68)

PRINT B%

Description : ***A%*** is an integer variable to be assigned to the result.  
***N% or N!*** is a numeric expression, it can be an integer or a real number.

## REM

Purpose : To insert explanatory remarks in a program.

Syntax : ***REM remark or 'remark'***

Example : REM This is function

' This is BASIC program

Description : ***remark*** may be any sequence of characters. BASIC interpreter will ignore whatever follows the REM or ' until end of the line'.

## SET PRECISION

Purpose : To set the precision of the decimal points for printing real number expression.

Syntax : ***SET\_PRECISION(N%)***

Example : A! = 3.141592654

SET\_PRECISION(6)

PRINT "A = ", A!                      ' A = 3.141593

Description : ***N%*** is a numeric expression in the range of 0 to 6.  
The precision default setting is two digits.



---

## **SGN**

Purpose : To return an indication of the mathematical sign (+ or -) of a given numeric expression.

Syntax : ***A% = SGN(N%) or A% = SGN(N!)***

Example : **A% = SGN(9.86)**

**PRINT A%**

**B% = SGN(-5.68)**

**PRINT B%**

**B% = SGN(0)**

**PRINT B%**

Description : ***N% or N!*** is a numeric expression, it can be an integer or a real number.

***A%*** is an integer variable to be assigned to the result.

<b>A%</b>	<b>Meaning</b>
1	N% >0
0	N% =0
-1	N% <0

---

## 3.2 Commands for decision structures

### IF ... THEN ... {ELSE IF...} [ELSE...] END IF

Purpose : To provide a decision structure for multiple-line conditional execution.

Syntax : ***IF condition1 THEN [statements1] {ELSE IF condition2 THEN statements2} [ELSE elstatements] END IF***

Example : PRINT "Input a number:"  
Result%=INPUT("",K%)  
IF K% < 10 THEN  
    PRINT "One digit"  
ELSE IF K% < 100 THEN  
    PRINT "Two digits"  
ELSE  
    PRINT "Over one Hundry!"  
END IF

Description : ***condition*** is a logical expression.  
***statements*** can be multiple lines of BASIC statements.

---

## ON ... GOSUB ...

Purpose : To call one of the specified subroutines depending on the value of the expression.

Syntax : ***ON N% GOSUB SubLabel/ SubName {,SubLabel/ SubName}***

Example :  
D% = DAY\_OF\_WEEK  
ON D% GOSUB MON, THE, WED, THR, FRI, SAT, SUN  
WHILE INKEY\$=""  
WEND  
END  
MON:  
PRINT "MONDAY"  
RETURN  
THE:  
PRINT "TUESDAY"  
RETURN  
WED:  
PRINT "WEDNESDAY"  
RETURN  
THR:  
PRINT "THURSDAY"  
RETURN  
FRI:  
PRINT "FRIDAY"  
RETURN  
SAT:  
PRINT "SATURDAY"  
RETURN  
SUN:  
PRINT "SUNDAY"  
RETURN

Description : *N%* is a numeric expression that is rounded to an integer. The value of *N%* determines which subroutine is to be called. If the value of *N%* is 0 or greater than the number of routines listed, the interpreter will continue with the next executable statement.  
***SubLabel*** is the name of a subroutine.

---

*SubName* is the line label of a subroutine.

## ON ... GOTO ...

Purpose : To branch to one of several specified Line Labels depending on the value of an expression.

Syntax : ***ON N% GOTO LineLabel / LineNumber {LineLabel / LineNumber}***

Example : D% = DAY\_OF\_WEEK  
ON D% GOTO 1, 2, 3, 4, 5, 6, 7

```
1
  PRINT "MONDAY"
  END
2
  PRINT "TUESDAY"
  END
3
  PRINT "WEDNESDAY"
  END
4
  PRINT "THURSDAY"
  END
5
  PRINT "FRIDAY"
  END
6
  PRINT "SATURDAY"
  END
7
  PRINT "SUNDAY"
  END
```

Description : *N%* is a numeric expression which is rounded to an integer. The value of *N%* determines which line label in the list will be used for branching. If the value *N%* is 0 or greater than the number of line labels listed, the interpreter will continue with the next executable statement.

***LineLabel*** is the string label of a program line.

***LineNumber*** is the integer number in front of a program line.

---

### 3.3 Commands for looping structures

#### EXIT

Purpose : To provide an alternative exit for looping structures, such as FOR...NEXT and WHILE...WEND statements.

Syntax : **EXIT**

Example : WHILE 1  
          IF INKEY\$=CHR\$(27) THEN     'if press ESC key  
          then quit  
          EXIT  
          END IF  
          WEND  
          PRINT "EXIT..."

Description : **EXIT** can appear anywhere within the loop statement.

#### FOR ... NEXT

Purpose : To repeat the execution of a block of statements for a specified number of times.

Syntax : **FOR N% = startvalue TO endvalue [STEP step]**  
          **[Statement Block]**  
          **NEXT**

Example : FOR N% = 1 TO 6 STEP 1  
          PRINT "FOR NEXT",N%  
          NEXT

Description : **N%** is an integer variable to be used as loop counter.  
**Startvalue** is a numeric expression which is the initial value for the loop counter.  
**Endvalue** is a numeric expression which is the final value for the loop counter.  
**Step** is a numeric expression to be used as an increment/decrement of the loop counter. The step is 1 by default.  
If the loop counter ever reaches or beyond the endvalue, the program execution continues to the statement following the NEXT statement. The Statement block will be executed again otherwise.

---

## **WHILE ... WEND**

Purpose : To repeat the execution of a block of statements while a certain condition is TRUE.

Syntax : **WHILE condition**  
**[Statement Block]**  
**WEND**

Example : N%=1  
WHILE 1  
    PRINT "Cnt=",N%  
    N%=N%+1  
    IF N%>5 THEN  
        EXIT  
    END IF  
WEND

Description : If the **condition** is true, loop statements are executed until the WEND statement is encountered. Then the program execution returns to WHILE statement and checks the condition again. If it is still true, the process will be repeated. Otherwise the execution continues with the statement following the WEND statement.

---

## 3.4 Commands for string processing

### LEN

Purpose : To return the length of a string.

Syntax : *A % = LEN(S\$)*

Example : Str\$="ABCDEFGHIJK"  
L% = LEN(Str\$)  
PRINT "Len. = ",L%

Description : *A %* is an integer variable to be assigned to the result.  
*S\$* may be a string variable, string expression, or string constant.

### INSTR

Purpose : To search if one string exists inside another one.

Syntax : *A % = INSTR([N%,] S1\$, S2\$)*

Example : Str\$="ABCGEFGHIJK"  
G\$="GH"  
PRINT INSTR(5,Str\$, G\$)  
PRINT INSTR(3, Str\$, "CGE")

Description : *A %* is an integer variable to be assigned to the result.  
*N%* is a numeric expression. Optional offset *N%* sets the position for starting the search.  
*S1\$, S2\$* may be a string variable, string expression, or string constant.  
If *S2\$* is found in *S1\$*, it returns the position of the first occurrence of *S2\$* in *S1\$*, from the starting point.  
If *N%* is larger than the length of *S1\$* or if *S1\$* is null, or if *S2\$* cannot be found, it returns 0.  
If *S2\$* is null, it returns *N%* (or 1 if *N%* is not specified).

---

## LEFT\$

Purpose : To retrieve a given number of characters from the left side of the target string.

Syntax : **A\$ = LEFT\$(Str\$, N%)**

Example : Str\$ = "ABCDEFGHJK"  
PRINT LEFT\$(Str\$,3)  
PRINT LEFT\$("168IbB",3)

Description : **A\$** is a string variable to be assigned to the result.  
**Str\$** may be a string variable, string expression, or string constant.  
**N%** is a numeric expression.  
If **N%** is larger than the length of **Str\$**, the **Str\$** is returned.  
If **N%** is zero, the null string is returned.

## MID\$

Purpose : To retrieve a given number of characters from anywhere of the target string.

Syntax : **A\$ = MID\$( Str\$, N%[, M%])**

Example : Str\$ = "ABCDEFGHJK"  
PRINT MID\$(Str\$,5,3)  
PRINT MID\$("123& #168IbB",6,5)

Description : **A\$** is a string variable to be assigned to the result.  
**Str\$** may be a string variable, string expression, or string constant.  
**N%** and **M%** are numeric expression.  
This command returns a string of length **M%** characters from **Str\$** beginning with the **N%**th character.  
If **M%** is equal to zero, or if **N%** is greater than the length of **Str\$**, then it returns a null string.



---

## RIGHT\$

Purpose : To retrieve a given number of characters from the right side of the target string.

Syntax :  $A\$ = \text{RIGHT}\$(Str\$, N\%)$

Example :  $Str\$ = \text{"ABCDEFGHIJK"}$   
 $\text{PRINT RIGHT}\$(Str\$,3)$   
 $\text{PRINT RIGHT}\$("168IbB",3)$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $Str\$$  may be a string variable, string expression, or string constant.  
 $N\%$  is a numeric expression.  
If  $N\%$  is larger than the length of  $Str\$$ , the entire string is returned.  
If  $N\%$  is zero, the null string is returned.

## TRIM LEFT\$

Purpose : To return a copy of a string with leading blank spaces stripped away.

Syntax :  $A\$ = \text{TRIM\_LEFT}\$(Str\$)$

Example :  $\text{PRINT TRIM\_LEFT}\$(" \text{Happy TEST END}")$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $Str\$$  is a string variable that may contain some space character at the beginning.

## TRIM RIGHT\$

Purpose : To return a copy of a string with trailing blank spaces stripped away.

Syntax :  $A\$ = \text{TRIM\_RIGHT}\$(Str\$)$

Example :  $\text{PRINT TRIM\_RIGHT}\$("Happy TEST END ")$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $Str\$$  is a string variable that may contain some space characters at the end.

## ASC

Purpose : To return the decimal value for the ASCII code for the first character of a given string.

Syntax :  $A\% = \text{ASC}(Str\$)$

Example :  $A\% = \text{ASC}(\text{"Test..."})$       $A\% = 84$

Description :  $A\%$  is an integer variable to be assigned to the result.  
 $Str\$$  is a string variable, consisting of characters.

---

## CHR\$

Purpose : To return the character for a given ASCII value.

Syntax :  $A\$ = \text{CHR}\$(N\%)$

Example :  $A\$ = \text{CHR}\$(66)$        $'A\$ = 'B'$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $N\%$  is a numeric expression in the range of 0 to 255.

## HEX\$

Purpose : To return a string that represents the hexadecimal value (base 16) of the decimal argument.

Syntax :  $A\$ = \text{HEX}\$(N\%)$

Example :  $A\$ = \text{HEX}\$(136)$        $'A\$ = "88"$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $N\%$  is a numeric expression.

## OCT\$

Purpose : To return a string that represents the octal value (base 8) of the decimal argument.

Syntax :  $A\$ = \text{OCT}\$(N\%)$

Example :  $A\$ = \text{OCT}\$(136)$        $'A\$ = "210"$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $N\%$  is a numeric expression.

## LCASE\$

Purpose : To return a copy of a string in which all uppercase letters will be converted to lowercase letters.

Syntax :  $A\$ = \text{LCASE}\$(Str\$)$

Example :  $Str\$ = "ABCDEFGH"$   
 $\text{PRINT LCASE}\$(Str\$)$   
 $\text{PRINT LCASE}\$("168BBqRrGgIbB")$

Description :  $A\$$  is a string variable to be assigned to the result.  
 $Str\$$  may be a string variable, string expression, or string constant.

---

## UCASE\$

Purpose : To return a copy of a string in which all lowercase letters will be converted to uppercase letters.

Syntax : ***A\$ = UCASE\$(Str\$)***

Example : ***Str\$="abcdeFG"***  
***PRINT UCASE\$(Str\$)***  
***PRINT UCASE\$("168BBqRrGgIbB")***

Description : ***A\$*** is a string variable to be assigned to the result.  
***Str\$*** may be a string variable, string expression, or string constant.

## STR\$

Purpose : To convert a numeric expression to a string.

Syntax : ***A\$ = STR\$(N%) or***  
***A\$ = STR\$(N!)***

Example : ***Str\$=STR\$(168)***  
***PRINT Str\$***

Description : ***A\$*** is a string variable to be assigned to the result.  
***N%*** is a numeric expression.

## VAL

Purpose : To return the numeric value of a string expression in interger form.

Syntax : ***A% = VAL(Str\$)***

Example : ***PRINT VAL("16898")***

Description : ***A%*** is an integer variable to be assigned to the result.  
***Str\$*** is a string that includes numeric characters.If the first character is not numeric, this command return 0.

## VALR

Purpose : To convert a string expression to a real number.

Syntax : ***A! = VALR(Str\$)***

Example : ***PRINT VALR("168.598")***

Description : ***A!*** is real number variable to be assigned to the result.  
***Str\$*** is a string that includes numeric characters. The precision of converted result is governed by the command SET\_PRECISION.

---

## **STRING\$**

Purpose : To return a string containing the specified number of the requested character.

Syntax : *A\$ = STRING\$(N%, J%)*  
*A\$ = STRING\$(N%, X\$)*

Example : PRINT STRING\$(10, 45)        ' -----  
          PRINT STRING\$(3, "89")     ' 888

Description : *A\$* is a string variable to be assigned to the result.  
*N%* is numeric expression.  
*J%* is numeric expression in the range of 0 to 255, indicating the ASCII code of a character.  
*X\$* may be a string variable or string constant.

---

## 3.5 Commands for event trapping

### OFF ALL

Purpose : To terminate all the event triggers.

Syntax : **OFF ALL**

Example : ON ESC GOSUB ESC\_PRESS

...

ESC\_PRESS:

OFF ALL

PRINT "ESC KEY PRESS..."

ON ESC GOSUB ESC\_PRESS

RETURN

Description : To resume the event trigger, call **ON event GOSUB...**

### OFF ESC

Purpose : To terminate ESC event trigger.

Syntax : **OFF ESC**

Example : ON ESC GOSUB ESC\_PRESS

...

ESC\_PRESS:

OFF ESC

...

ON ESC GOSUB ESC\_PRESS

RETURN

Description : To resume the event trigger, call **ON ESC GOSUB...**

### OFF COM

Purpose : To terminate COM event trigger.

Syntax : **OFF COM(N%)**

Example : ON COM(1) GOSUB READ1

...

READ1:

OFF COM(1)

...

ON COM(1) GOSUB READ1

RETURN

Description : **N%** is an integer variable, indicating the COM port. Now we only can choose 1(RS232).

To resume the event trigger, call **ON COM... GOSUB...**

---

## **OFF HOUR**

Purpose : To terminate HOUR event trigger.

Syntax : ***OFF HOUR***

Example : ON HOUR GOSUB A10

...

A10:

OFF HOUR

...

ON HOUR GOSUB A10

RETURN

Description : To resume the event trigger, call ***ON HOUR GOSUB...***

## **OFF KEY**

Purpose : To terminate KEY event trigger.

Syntax : ***OFF KEY(number%)***

Example : ON KEY(1) GOSUB F1

ON KEY(2) GOSUB F2

...

F1:

OFF KEY(1)

...

ON KEY(1) GOSUB F1

RETURN

F2:

OFF KEY(2)

...

ON KEY(2) GOSUB F2

RETURN

Description : To resume the event trigger, call ***ON KEY... GOSUB...***  
***number%*** is an integer variable in the range of 1 to 6,  
indicating a function key of the keypad.

---

## **OFF MINUTE**

Purpose : To terminate MINUTE event trigger.

Syntax : ***OFF MINUTE***

Example : ON MINUTE GOSUB A10

...

A10:

OFF MINUTE

...

ON MINUTE GOSUB A10

RETURN

Description : To resume the event trigger, call ***ON MINUTE GOSUB...***

## **OFF READER**

Purpose : To terminate READER event trigger.

Syntax : ***OFF READER(N%)***

Example : ON READER(1) GOSUB GetData

...

GetData:

OFF READER(1)

CLS

A\$=GET\_READER\_DATA\$(1,4)

PRINT "DATA:"+A\$

LOCATE 0,2

A\$=GET\_READER\_DATA\$(1,1)

PRINT "Name:"+A\$

LOCATE 0,4

PRINT GET\_READER\_DATALEN

...

ON READER(1) GOSUB GetData

RETURN

Description : To resume the event trigger, call ***ON READER... GOSUB...***  
*N%* is an integer variable, indicating the reader port (now we only can choose 1).

---

## OFF TIMER

Purpose : To terminate TIMER event trigger.

Syntax : ***OFF TIMER(N%)***

Example : ON TIMER(1,200) GOSUB A1

ON TIMER(2,300) GOSUB A2

...

A1:

OFF TIMER(1)

...

RETURN

A2:

OFF TIMER(2)

...

RETURN

Description : To resume the event trigger, call ***ON TIMER... GOSUB...***  
*N%* is an integer variable in the range of 1 to 5, indicating the timer ID.

## ON COM GOSUB

Purpose : To activate COM event trigger.

Syntax : ***ON COM(N%) GOSUB SubLabel / SubName***

Example : ON COM(1) GOSUB READ1

...

READ1:

OFF COM(1)

...

ON COM(1) GOSUB READ1

RETURN

Description : When data is received from the COM port, a specific subroutine will be executed.  
*N%* is an integer variable, indicating the COM port (now we only can choose 1).



---

## **ON ESC GOSUB**

Purpose : To activate ESC event trigger.

Syntax : ***ON ESC GOSUB SubLabel / SubName***

Example : ON ESC GOSUB ESC\_PRESS

...

ESC\_PRESS:

OFF ESC

...

ON ESC GOSUB ESC\_PRESS

RETURN

Description : When ESC key is pressed, a specific subroutine will be executed.

## **ON HOUR GOSUB**

Purpose : To activate HOUR event trigger.

Syntax : ***ON HOUR GOSUB SubLabel / SubName***

Example : ON HOUR GOSUB OnHourAlarm

...

OnHourAlarm:

CurrentTime\$=TIME\$

H%=VAL(LEFT\$(CurrentTime\$,2))

FOR I%=1 TO H%

BEEP(30,20,0,0)

WAIT(100)

NEXT

RETURN

Description : When the system time is on the hour, a specific subroutine will be executed.

---

## ON KEY GOSUB

Purpose : To activate KEY event trigger.

Syntax : ***ON KEY(number%) GOSUB SubLabel / SubName***

Example : ON KEY(1) GOSUB F1

ON KEY(2) GOSUB F2

...

F1:

OFF KEY(1)

...

RETURN

F2:

OFF KEY(2)

...

RETURN

Description : When a function key is pressed, a specific subroutine will be executed.

***number%*** is an integer variable in the range of 0 to 9, indicating a function key of the keypad.

## ON MINUTE GOSUB

Purpose : To activate MINUTE event trigger.

Syntax : ***ON MINUTE GOSUB SubLabel / SubName***

Example : ON MINUTE GOSUB AMINUTE

...

AMINUTE:

CurrentTime\$=TIME\$

CurrentMin%=VAL(MID\$(CurrentTime\$,3,2))

IF CurrentMin%=30 THEN

BEEP(30,50,0,0)

WAIT(200)

END IF

RETURN

Description : When the system time is on the minute, a specific subroutine will be executed.

---

## **ON READER GOSUB**

Purpose : To activate READER event trigger.

Syntax : ***ON READER(N%) GOSUB SubLabel / SubName***

Example : ON READER(1) GOSUB GetData

...

GetData:

OFF READER(1)

CLS

A\$=GET\_READER\_DATA\$(1,4)

PRINT "DATA:"+A\$

LOCATE 0,2

A\$=GET\_READER\_DATA\$(1,1)

PRINT "Name:"+A\$

LOCATE 0,4

PRINT GET\_READER\_DATALEN

...

ON READER(1) GOSUB GetData

RETURN

Description : When data is received from reader port, a specific subroutine will be executed.

*N%* is an integer variable, indicating the reader port (now we only can choose 1).

---

## **ON TIMER GOSUB**

Purpose : To activate TIMER event trigger.

Syntax : ***ON TIMER(N%, duration%) GOSUB SubLabel / SubName***

Example : ON TIMER(1,200) GOSUB TimeOut

...

TimeOut:

OFF TIMER(1)

...

RETURN

Description : When the system runs out of the time duration specified by user, a specific subroutine will be executed.

Up to five timers can be set in a BASIC program. Be sure the timer ID's are properly differentiated. Otherwise, the latter created timer will overwrite the former one.

*N%* is an integer variable in the range of 1 to 5, indicating the ordinal number of timer.

*duration%* is an integer variable, indicating a specified period of time in units of 10 ms.

---

## **LOCK**

Purpose : To hold all the activated event triggers until they are released by UNLOCK.

Syntax : **LOCK**

Example : ON KEY(1) GOSUB F1  
ON KEY(2) GOSUB F2

...

F1:

LOCK

PRINT "press F1"

UNLOCK

RETURN

F2:

PRINT "press F2"

RETURN

In this example, the BASIC program can trap the KEY(1) and KEY(2) events and reroute to the subroutines F1 and F2 respectively. In F1, the command LOCK disable all the activated event triggers so that the subroutine F1 will not be interrupted by a new upcoming KEY(1) and KEY(2) event. On the other hand, since LOCK is not called in F2, any new coming KEY(1) and KEY(2) event will interrupt the ongoing F2, and therefore, may affect the expected results.

Description : This command can prevent nesting of event triggers. All the activated event triggers will be disabled until UNLOCK is called.

---

## UNLOCK

Purpose : To release all the activated event triggers held by LOCK.

Syntax : ***UNLOCK***

Example : ON KEY(1) GOSUB F1  
ON KEY(2) GOSUB F2

...

F1:

LOCK

PRINT "press F1"

UNLOCK

RETURN

F2:

PRINT "press F2"

RETURN

Description : This command resumes event processing.

---

## 3.6 System commands

### AUTO\_OFF

- Purpose : To set auto power off timer.
- Syntax : ***AUTO\_OFF(N%)***
- Example : **AUTO\_OFF(56)**
- Description : ***N%*** is an integer variable in the range from 30 to 65535, indicating a specified period of time in units of 1 second. If the time interval is set to zero, this function will be disabled.

### DEVICE\_ID\$

- Purpose : To get the serial number of the terminal.
- Syntax : ***A\$ = DEVICE\_ID\$***
- Example : **PRINT "S/N:" + DEVICE\_ID\$**
- Description : ***A\$*** is a string variable to be assigned to the result. That is a string of the target terminal serial number to be returned.

### GET\_TARGET\_MACHINE\$

- Purpose : To get the model name of the target terminal.
- Syntax : ***A\$ = GET\_TARGET\_MACHINE\$***
- Example : **PRINT "Model Name:" + GET\_TARGET\_MACHINE\$**
- Description : ***A\$*** is a string variable to be assigned to the result. That is a string of the model name of the target terminal to be returned.

### MENU

- Purpose : To create a menu.
- Syntax : ***A% = MENU(Item\$)***
- Example : **MENU\_STR\$="1.Auto off"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"2.System Info"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"3.Power on"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"4.Suspend"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"5.Restart"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"6.Exit"+CHR\$(13)**  
**MENU\_STR\$=MENU\_STR\$+"@SYSTEM**  
**TEST"+CHR\$(13)**  
**...**  
**S%=MENU(MENU\_STR\$)**  
**ON S% GOTO 10,20,30,40,50,60**  
**...**

---

Description : *A%* is an integer variable to be assigned to the result, it is the ordinal number of the menu item that user has selected.  
*Item\$* is a string variable, indicating the menu item that are separated and ended by carriage return (CR, 0xd).  
This command allows user to select an item by using the UP/DOWN arrow keys (or the shortcut keys), and then the ENTER key to confirm the selection. Also it allows the use of ESC key to cancel the current operation.

■ Menu title : @ (the title can be put anywhere in the menu string)

## **POWER\_ON**

Purpose : To determine whether to restart or resume the program upon powering on.

Syntax : ***POWER\_ON(N%)***

Example : POWER\_ON(0)     'Resume

Description : *N%* can be set 0 or 1.

<i>N%</i>	<i>Meaning</i>
0	Resume
1	Reset

## **RESTART**

Purpose : To restart the system.

Syntax : ***RESTART***

Example : ON ESC GOSUB ESC\_PRESS  
...  
ESC\_PRESS:  
    RESTART  
    RETURN

Description : This command will terminate the execution of the BASIC program and restart the system.

## **SYSTEM\_INFORMATION**

Purpose : To get information on components.

Syntax : ***A\$=SYSTEM\_INFORMATION(index%)***

Example : PRINT "Kernel:"+SYSTEM\_INFORMATION\$(1)  
          PRINT "BASIC:"+SYSTEM\_INFORMATION\$(2)  
          PRINT "SCANNER:"+SYSTEM\_INFORMATION\$(3)



Description : *A\$* is a string variable to be assigned to the result.  
*index%* is an integer variable, indicating a specific category of information.

<i>index%</i>	<i>Meaning</i>
1	Kernel version
2	BASIC version
3	Scanner version

## **SYS\_SUSPEND**

Purpose : To shut down the system.

Syntax : ***SYS\_SUSPEND***

Example : **SYS\_SUSPEND**

Description : This command will shut down the system.

## **CHECK\_AID**

Purpose : To check the agency ID is correct or not.

Syntax : ***A%=CHECK\_AID(S1\$, S2\$)***

Example : IF CHECK\_AID("6421","08724") THEN  
PRINT "AID OK..."  
ELSE  
PRINT "AID NG..."  
END IF  
WHILE INKEY\$=""  
WEND

Description : *A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	AID not correct.
1	AID correct.

*S1\$* is a string variable, indicating the UserID that needs 4~8 characters.

*S2\$* is a string variable, indicating the password that needs 4~8 characters.

## **COPYAPPTOBIOS**

Purpose : To copy the setting from APP to BIOS.

Syntax : ***COPYAPPTOBIOS***

Example : **COPYAPPTOBIOS**

Description : This command will copy the APP settings to BIOS.

## **SET\_DCIN\_ALWAYS\_ON**

Purpose : To set the state of DC in always power on.

---

Syntax : **SET\_DCIN\_ALWAYSON(N%)**  
Example : SET\_DCIN\_ALWAYSON(N%)  
Description : **N%** can be set 0 or 1.

<b>N%</b>	<b>Meaning</b>
0	Disable
1	Enable

## **GET\_DCIN\_ALWAYSON**

Purpose : To get the state of DC in always power on.  
Syntax : **A% = GET\_DCIN\_ALWAYSON**  
Example : A% = GET\_DCIN\_ALWAYSON  
Description : **A%** is an integer variable to be assigned to the result.

<b>A%</b>	<b>State</b>
0	Disable
1	Enable

---

## 3.7 Reader commands

### **DISABLE READER**

Purpose : To disable the reader ports of the terminal.

Syntax : ***DISABLE READER(N%)***

Example : DISABLE READER(1)

Description : *N%* is an integer variable, indicating the reader port (now we only can choose 1).

### **ENABLE READER**

Purpose : To enable the reader ports of the terminal.

Syntax : ***ENABLE READER(N%)***

Example : ON READER(1) GOSUB SCAN  
ENABLE READER(1)

...

SCAN:

OFF READER(1)

CLS

A\$=GET\_READER\_DATA\$(1,4)

PRINT "DATA:"+A\$

LOCATE 0,2

A\$=GET\_READER\_DATA\$(1,1)

PRINT "Name:"+A\$

LOCATE 0,4

PRINT GET\_READER\_DATALEN

LOOP1:

S1\$=INKEY\$

IF S1\$="" THEN

GOTO LOOP1

END IF

ON READER(1) GOSUB SCAN

RETURN

Description : *N%* is an integer variable, indicating the reader port (now we only can choose 1).

---

## **SLEEP\_READER**

Purpose : To set scanner module to sleep.

Syntax : ***SLEEP\_READER(N%)***

Example : SLEEP\_READER (1)      ‘Scanner to sleep

Description : ***N%*** is an integer variable.

<b><i>N%</i></b>	<b><i>Meaning</i></b>
0	Not sleep
1	To sleep

## **GET\_READER\_DATA\$**

Purpose : To get data that is read from a specified reader ports.

Syntax : ***A\$ = GET\_READER\_DATA\$(N1%,N2%)***

Example : ON READER(1) GOSUB SCAN

ENABLE READER(1)

...

SCAN:

...

A\$=GET\_READER\_DATA\$(1,4)

...

RETURN

Description : This command will get reader port data.

***A\$*** is a string variable to be assigned to the result.

***N1%*** is an integer variable, indicating the reader port (now we only can choose 1).

***N2%*** is an integer variable, indicating what kind of data to be retrieved.

<b><i>N2%</i></b>	<b><i>Meaning</i></b>
1	Code Name
2	Full Code
3	Code ID
4	Data

■ The format of Full Code as follows:

Code name	Preamble	ID *	Code Length	Barcode data	ID *	Postamble	Suffix
--------------	----------	---------	----------------	-----------------	---------	-----------	--------

The ID position depends on “Code ID position” setting.

---

## **GET READER DATALEN**

Purpose : To get data length that is read from a specified reader ports.

Syntax : *A%=GET\_READER\_DATALEN*

Example : A% = GET\_READER\_DATALEN

Description : A% is an integer variable to be assigned to the result.

## **GET READER TYPE**

Purpose : To get scanner type.

Syntax : *A% = GET\_READER\_TYPE*

Example : A% = GET\_READER\_TYPE

Description : A% is an integer variable to be assigned to the result.

<i>A%</i>	<i>Type</i>
0	CCD (only PI-10X0)
2	2D (only PI-12X0)

## **READER CONFIG START**

Purpose : To start scanner setting procedure.

Syntax : *READER\_CONFIG\_START*

Example : READER\_CONFIG\_START

A%=READER\_SENDCMD(11,1, CHR\$(1))      ‘Code-39  
can read

...

READER\_CONFIG\_END

Description : This command can start scanner setting procedure.

## **READER CONFIG END**

Purpose : To terminate scanner setting procedure.

Syntax : *READER\_CONFIG\_END*

Example : READER\_CONFIG\_END

Description : This command can terminate scanner setting procedure.

---

## **READER\_SENDCMD**

Purpose : To send scanner (CCD) command to change scanner status.

Syntax : *A%=READER\_SENDCMD(N1% , N2% , S\$)*

Example : READER\_CONFIG\_START

...

‘Code-39 can read

*A%=READER\_SENDCMD(11,1, CHR\$(1))*

“Code-93 Checksum verification disable

*A%=READER\_SENDCMD(12,2, CHR\$(0))*

‘Preamble characters setting

*A%=READER\_SENDCMD(8,3, “abcde”)*

...

READER\_CONFIG\_END

Description : This command can change scanner status.

*A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	Change fail
1	Change OK

*N1%* is an integer variable, indicating the parameter1.

*N2%* is an integer variable, indicating the parameter2.

*S\$* is a string variable.

Refer to “[Appendix B](#)” for more details about the parameter setting.

## **READER\_QUERY\$**

Purpose : To query the scanner(CCD) current setting.

Syntax : *A\$=READER\_QUERY\$(N1% , N2%)*

Example : ‘To query the scanner status (Code-128/Read).

*Value\$=READER\_QUERY\$(13, 1)*

*PRINT "Value:",ASC(Value \$)*

*Preamble\$=READER\_QUERY\$(8, 3)* ‘Preamble characters

*PRINT " Preamble:"+ Preamble \$*

Description : *A\$* is a string variable to be assigned to the result.

*N1%* is an integer variable, indicating the parameter1.

*N2%* is an integer variable, indicating the parameter2.

Refer to “[Appendix B](#)” for more details about the parameter setting.

---

## DECODE

Purpose : To perform barcode decoding.

Syntax : **DECODE**

Example :       ENABLE READER(1)

...

MAIN:

    IF DECODE <>0 THEN

        CLS

        LOCATE 0,0

        A\$=GET\_READER\_DATA\$(1,4)

        PRINT "DATA:"+A\$

        LOCATE 0,2

        A\$=GET\_READER\_DATA\$(1,1)

        PRINT "Name:"+A\$

        LOCATE 0,4

        PRINT "Length:",GET\_READER\_DATALEN

        LOCATE 0,6

        A\$=GET\_READER\_DATA\$(1,2)

        PRINT "FULL:"+A\$

        LOCATE 0,8

        PRINT "ID:"+GET\_READER\_DATA\$(1,3)

    END IF

    IF INKEY\$=CHR\$(27) THEN

        DISABLE READER(1)

    END

END IF

GOTO MAIN

Description : Once the scanner port is initialized (by using ENABLE READER command), call this DECODE command to perform barcode decoding. This command should be called constantly in user's program loops when barcode decoding is required. If the barcode decoding is not required for a long period of time, it is recommended that the scanner port be disabled by using DISABLE READER command.

---

## **SIM\_SCANKEY\_PRESS**

Purpose : To simulator the “Scan” key press or release.

Syntax : ***SIM\_SCANKEY\_PRESS(N1%)***

Example : ‘Set the scan key pressed.  
SIM\_SCANKEY\_PRESS(1)

...

‘Set the scan key released.  
SIM\_SCANKEY\_PRESS(0)

Description : This command can simulator the scan key status for pressed or released.

## **READER\_SETFROMFILE**

Purpose : To set scanner setting by scanner setting file.

Syntax : ***A%=READER\_SETFROMFILE(FilePath\$)***

Example : A%=READER\_SETFROMFILE("c:\data\PI1030.axs")

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Setting fail
1	Setting OK

***FilePath \$*** is a string variable, indicating the Scanner setting file path.



---

## 3.8 Beeper commands

### **BEEP**

Purpose : To assign a beeper sequence to designate beeper operation.

Syntax : ***BEEP(freq%, duration% {, freq%, duration%})***

Example : BEEP(99,30,0,10,88,30,0,10,66,30,0,0)

Description : *freq%* is an integer variable, indicating the value of ***Beep frequency ( 76000 / Actual Frequency Desired )***.

A beep frequency is an integer used to specify the frequency (tone) when the beeper been activated. The actual frequency that the beeper gives is not the value specified to the beep frequency. It is calculated by the following formula.

For instance, to get a frequency of 2000Hz, the value of beep frequency should be 38. If no sound is desired (pause), the beep frequency should be set to 0. A beep with frequency 0 does not terminate the beeper sequence. Suitable frequency for the beeper ranges from 1 to 2700Hz, while peak volume is at around 2000Hz.

*Duration%* is an integer variable, indicating the value of beeping duration, which is specified in units of 10 ms.

### **STOP BEEP**

Purpose : To terminate beeper sequence.

Syntax : ***STOP BEEP***

Example : BEEP(99,100,0,30,88,100,66,100,0,0)  
WAIT(200)  
STOP BEEP

Description : The STOP BEEP statement terminates the beeping immediately if there is a beeper sequence in progress.

---

## **SET BUZZER VOL**

Purpose : To set the buzzer volume.

Syntax : ***SET\_BUZZER\_VOL(N%)***

Example : SET\_BUZZER\_VOL(2)

Description : *N%* is an integer variable to be assigned to the result.

<i>N%</i>	<i>Buzzer volume</i>
0	close
1	Low
2	Medium
3	High

---

## 3.9 Calendar and timer commands

### DATE\$

- Purpose : To set or to get the current date.
- Syntax : **DATE\$ = X\$**  
**Y\$ = DATE\$**
- Example : PRINT "NOW:" + DATE\$  
DATE\$ = "20090115"  
PRINT "SET:" + DATE\$
- Description : **X\$** is a string variable in the form of "yyyymmdd".  
**DATE\$ = X\$**, to set the current date.  
**Y\$** is a string variable to be assigned to the result.  
**Y\$ = DATE\$**, to get the current date, in the form "yyyymmdd".

### DAY OF WEEK

- Purpose : To get the day of the week.
- Syntax : **A% = DAY\_OF\_WEEK**
- Example : PRINT DAY\_OF\_WEEK
- Description : **A%** is an integer variable to be assigned to the result. A value of 1 to 7 represents Monday to Sunday respectively.

### TIME\$

- Purpose : To set or to get the current time.
- Syntax : **TIME\$ = X\$**  
**Y\$ = TIME\$**
- Example : PRINT TIME\$  
TIME\$ = "180831"  
PRINT TIME\$
- Description : **X\$** is a string variable in the form of "hhmmss".  
**TIME\$ = X\$**, to set the current time.  
**Y\$** is a string variable to be assigned to the result.  
**Y\$ = TIME\$**, to get the current time, in the form of "hhmmss".

### TIMER

- Purpose : To return the number of seconds elapsed since the terminal is powered on.
- Syntax : **A% = TIMER**
- Example : PRINT TIMER
- Description : **A%** is an integer variable to be assigned to the result.

---

## WAIT

Purpose : To set system delay time.

Syntax : **WAIT(duration%)**

Example : WAIT(1000) '5sec

Description : **duration%** is a positive integer variable, indicating the time duration for a hold. This argument is specified in units of 5 ms.

---

## 3.10 LED Command

### LED

Purpose : To set the LED indicators.

Syntax : ***LED(number%, mode%, duration%)***

Example : LED(2,2,100)

Description :

<b><i>number%</i></b>	<b><i>description</i></b>
1	LED displays green light.
2	LED displays red light.
3	LED displays orange light.

<b><i>mode%</i></b>	<b><i>description</i></b>
1	off for ( <b><i>duration%</i></b> X 0.01) seconds then on
2	on for ( <b><i>duration%</i></b> X 0.01) seconds then off
3	flash, on then off each for ( <b><i>duration%</i></b> X 0.01) seconds then repeat

---

## 3.11 Keypad commands

### CLR\_KBD

Purpose : To clear the keypad buffer.

Syntax : **CLR\_KBD**

Example : CLR\_KBD

Description : This command will clear keypad buffer.

### INKEY\$

Purpose : To read one character from the keypad buffer then remove it.

Syntax : **Str\$ = INKEY\$**

Example : START:

```
S$=INKEY$
IF S$<>"" THEN
    PRINT ASC(S$)
    IF ASC(S$)=27 THEN    'ESC key
        END
    END IF
END IF
GOTO START
...
```

Description : **Str\$** is a string variable to be assigned to character read.

### INPUT\_LEN

Purpose : To set or get input length limit when using "INPUT" or INPUT\_S" command.

Syntax : **X% = INPUT\_LEN**  
**INPUT\_LEN = A%**

Example : INPUT\_LEN=4  
PRINT "INPUT STRING:"  
A%=INPUT("",S\$)

...

PRINT "Input length: "; INPUT\_LEN

Description : **A%** is an integer variable. When using "INPUT" or "INPUT\_S" command, it can set limit on input length(When N%=0 indicating not limit).

**X%** is an integer variable, indicating the input length limit.

---

## INPUT

Purpose : To retrieve input from the keypad and store it in a variable.

Syntax : ***A%=INPUT(S\$, variable)***

Example : PRINT "INPUT STRING:"

Result%=INPUT("",String\$) 'Input a string variable

PRINT "INPUT NUMBER:"

Result %=INPUT("123",Number%) 'Input a numeric variable

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Press the ENT key and has not input any item.
1	Inputs correctly.
255	Press the ESC key.
-1	Input error.

***S\$*** is a string variable, indicating the input default value.

***variable*** is numeric or string variable that will receive the input data. The data entered must match the data type of the variable.

When the input task is properly ended with the ENTER key being pressed, the data string will be stored in a variable. Otherwise, press the ESC key to abort the task.

## INPUT S

Purpose : To retrieve input from the keypad, scanning and store it in a variable.

Syntax : ***A%=INPUT\_S(S\$, variable)***

Example : Result%=INPUT\_S("",String\$)

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Press the ENT key and has not input any item.
1	Inputs correctly.
255	Press the ESC key.

-1	Input error.
----	--------------

*S\$* is a string variable, indicating the input default value.

*variable* is numeric or string variable that will receive the input data. The data entered must match the data type of the variable.

When the input task is properly ended with the ENTER key being pressed, the data string will be stored in a variable. Otherwise, press the ESC key to abort the task.

## **INPUT S CARRYENT**

Purpose : To set ENT auto press on/off when using “INPUT\_S\_CARRYENT” command.

Syntax : **INPUT\_S\_CARRYENT(N%)**

Example : INPUT\_S\_CARRYENT(1)

Description : *N%* is an integer variable. When using “INPUT\_S\_CARRYENT” command, it can set auto press ENT on/off key after scanner reading.

<i>N%</i>	<i>Auto press ENT</i>
0	No
1	Yes

## **INPUT S VIBRATE**

Purpose : To set vibrator on or off when using “INPUT\_S\_VIBRATE” command.

Syntax : **INPUT\_S\_VIBRATE(N%)**

Example : INPUT\_S\_VIBRATE(1)

Description : *N%* is an integer variable. When using “INPUT\_S\_VIBRATE” command, it can set vibrator on or off after scanner reading.

<i>N%</i>	<i>Meaning</i>
0	Vibrate off
1	Vibrate on



---

## **INPUT S SLEEP**

Purpose : To set scanner sleep on or off when using “INPUT\_S\_SLEEP” command.

Syntax : **INPUT\_S\_SLEEP(N%)**

Example : INPUT\_S\_SLEEP(1)  
R%=INPUT\_S("",S1\$)      ‘Scanner to sleep  
...

Description : **N%** is an integer variable. After using “INPUT\_S\_SLEEP” command, the “INPUT\_S\_SLEEP” command can set scanner to sleep or not.  
If use this command and set “1”, when leaving “INPUT\_S” command, scanner will go to sleep.

<b>N%</b>	<b>Meaning</b>
0	Not sleep(scanner go to suspend)
1	To sleep

## **INPUT MODE**

Purpose : To set the display mode of the input data.

Syntax : **INPUT\_MODE(mode%)**

Example : INPUT\_MODE(2)

Description : **mode%** is an integer variable, indicating the input mode.

<b>mode%</b>	<b>Meaning</b>
0	Nothing will be displayed on the LCD.
1	The input characters will be displayed on the LCD (default).
2	“*” will be displayed instead of the input characters. Usually it is applied for password input.

## **KEY CLICK**

Purpose : To enable or disable the key click sound.

Syntax : **KEY\_CLICK(status%)**

Example : KEY\_CLICK(0)

Description : **status%** is an integer variable, indicating the key click status.

<b>status%</b>	<b>Key click sound</b>
0	Disable
1	Enable

---

## **ALPHA LOCK**

Purpose : To set the ALPHA state for input mode.

Syntax : ***ALPHA\_LOCK(status%)***

Example : ALPHA\_LOCK(1)

Description : ***status%*** is a string variable, indicating the Alpha status.

<b><i>status%</i></b>	<b><i>Alpha status</i></b>	<b><i>Default input</i></b>
0	Unlock	Numeric mode
1	Lock	Alpha mode (lower case)
2	Lock	Alpha mode (upper case)
3	Lock	Numeric mode

## **GET ALPHA LOCK**

Purpose : To get information of the ALPHA state for input mode.

Syntax : ***A% = GET\_ALPHA\_LOCK***

Example : Alpha\_lock% = GET\_ALPHA\_LOCK

Description : ***A%*** is an integer variable to be assigned to the result.

## **GET KEY CLICK**

Purpose : To get current key click status.

Syntax : ***A% = GET\_KEY\_CLICK***

Example : Key\_click% = GET\_KEY\_CLICK

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Key click sound</i></b>
0	Off
1	On

---

## **KEYPAD\_BL\_TIMER**

Purpose : To set or get keypad backlight timer.

Syntax : ***A% = KEYPAD\_BL\_TIMER***  
***KEYPAD\_BL\_TIMER = X%***

Example : KEYPAD\_BL(0)  
PRINT "K,B timer=",KEYPAD\_BL\_TIMER  
...  
KEYPAD\_BL\_TIMER=3      'Keypad backlight timer=3  
sec

Description : ***A%*** is an integer variable to be assigned to the keypad backlight timer.  
***X%*** is an integer variable indicating a period of time in units of 1-second.

## **KEYPAD\_BL**

Purpose : To set keypad backlight on or off.

Syntax : ***KEYPAD\_BL(N%)***

Example : KEYPAD\_BL(1)

Description : ***N%*** is an integer variable indicating the keypad backlight on or off.

<b><i>N%</i></b>	<b><i>Keypad backlight status</i></b>
0	Off
1	On

---

## **DEF\_PKEY**

Purpose : To change the definition of programmable key (P1 ~ P3) .

Syntax : ***DEF\_PKEY(N1% ,N2%)***

Example :

DEF_PKEY(1,13)	'P1 key define to ENT key
DEF_PKEY(2,49)	'P2 key define to '1' key
DEF_PKEY(1,21)	'P1 key define to P1 key
DEF_PKEY(2,22)	'P2 key define to P2 key
DEF_PKEY(3,5)	'P3 key define to UP key
DEF_PKEY(2,6)	'P2 key define to DOWN key
DEF_PKEY(1,7)	'P1 key define to LEFT key
DEF_PKEY(3,11)	'P3 key define to RIGHT key
DEF_PKEY(1,27)	'P1 key define to ESC key
DEF_PKEY(2,8)	'P2 key define to BS key
DEF_PKEY(3,127)	'P3 key define to DEL key
DEF_PKEY(2,32)	'P2 key define to SP key
DEF_PKEY(1,45)	'P1 key define to '- ' key

Description :

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
1	Define P1 key
2	Define P2 key
3	Define P3 key

N2% is an integer variable indicating the key to be defined.

---

## 3.12 LCD Commands

The following commands: `CURSOR`, `CURSOR_X`, `CURSOR_Y`, `LOCATE`, `FILL_RECT`, `PRINT`, `CLR_RECT`, `CLS`, `SHOW_IMAGE`, `CLR_EOL`, will only affect the current TextBlock on LCD screen. Parameters of these commands will be based on TextBlock's size and position.

### BACK LIGHT DURATION

Purpose : To specify how long the backlight will last once the terminal is turned on.

Syntax : ***BACK\_LIGHT\_DURATION(N%)***

Example : `BACK_LIGHT_DURATION(20)`

Description : *N%* is an integer variable indicating the LCD backlight timer in the range from 0 to 65535. It is specified in units of 1-sec.

■ If *N%=0*, then LCD backlight will always be on.

### LCD CONTRAST

Purpose : To set the contrast level of the LCD.

Syntax : ***LCD\_CONTRAST(N%)***

Example : `LCD_CONTRAST(5)`

Description : *N%* is an integer variable indicating the LCD contrast level in the range from 1 to 10. The higher value means higher contrast.

### CURSOR

Purpose : To turn on/off the cursor indication in the activated TextBlock.

Syntax : ***CURSOR(status%)***

Example : `CURSOR(1)`

Description : *status%* is an integer indicating the cursor on or off.

<i>status%</i>	<i>Meaning</i>
0	Cursor off
1	Cursor on

### CURSOR\_X

Purpose : To get the x coordinate of the current cursor position in the activated TextBlock.

Syntax : ***X% = CURSOR\_X***

Example : `X% = CURSOR_X`

Description : *X%* is an integer variable to be assigned to the X coordinate of the current cursor position.

---

## **CURSOR\_Y**

- Purpose : To get the y coordinate of the current cursor position in the activated TextBlock.
- Syntax : ***Y% = CURSOR\_Y***
- Example : ***Y% = CURSOR\_Y***
- Description : ***Y%*** is an integer variable to be assigned to the Y coordinate of the current cursor position.

## **LOCATE**

- Purpose : To move the cursor to a specified location in the activated TextBlock.
- Syntax : ***LOCATE X% , Y%***
- Example : LOCATE 0,0  
...  
LOCATE 2,3  
...
- Description : ***X%*** is an integer variable indicating the new X coordinate position of the cursor.  
***Y%*** is an integer variable indicating the new Y coordinate position of the cursor.

## **FILL\_RECT**

- Purpose : To fill a rectangular area in the activated TextBlock.
- Syntax : ***FILL\_RECT(left% , top% , width% , height%)***
- Example : ***FILL\_RECT(100,100,100,100)*** 'green rectangular area
- Description : Several the argument as follows:

<b><i>left %</i></b>	Fill form the start point of X-axis (pixel).
<b><i>top %</i></b>	Fill form the start point of Y-axis (pixel).
<b><i>width%</i></b>	Fill the width form the start point (pixel).
<b><i>height%</i></b>	Fill the high form the start point (pixel).

---

## ICON\_ZONE\_PRINT

Purpose : To enable or disable the status bar.

Syntax : **ICON\_ZONE\_PRINT**(*status%*)

Example : ICON\_ZONE\_PRINT(0)

Description : *status%* is an integer variable indicating the status bar is on or off.

If using this command, all of the TextBlock setting will be reset.

<i>status%</i>	<i>Meaning</i>
0	Status bar off
1	Status bar on

## PRINT

Purpose : To display data in the activated TextBlock.

Syntax : **PRINT** *expression*[{,/[*expression*]}]

Example : PRINT "Print data"

X% = CURSOR\_X

Y% = CURSOR\_Y

PRINT "Cur. Location=>(";X%;",";Y%;")"

Description : *expression* may be numeric or string expression.

The position of echo printed item is determined by the punctuation used to separate items in the list. In the list of expression, a comma causes the next character to be printed after the last character with a blank space. A semicolon causes the next character to be printed immediately after the last character. If the list of expressions terminates without a comma or semicolon, a carriage return is printed at the end of the line.

## CLR\_RECT

Purpose : To clear a rectangular area in the activated TextBlock. The cursor position is not affected after the operation.

Syntax : **CLR\_RECT**(*left%* , *top%* , *width%* , *height%*)

Example : CLR\_RECT(100,100,100,100)

Description : Several key argument as below:

<i>left %</i>	Fill from the start point of X-axis (pixel).
<i>top %</i>	Fill from the start point of Y-axis (pixel).
<i>width%</i>	Fill the width from the start point (pixel).
<i>height%</i>	Fill the height from the start point (pixel).

---

## CLS

Purpose : To clear the activated TextBlock.

Syntax : **CLS**

Example : CLS

Description : After executing this command, whatever being shown on the LCD will be erased and the cursor will be moved to (0,0).

## SHOW\_IMAGE

Purpose : To put a rectangular bitmap in the activated TextBlock.

Syntax : **SHOW\_IMAGE(left% , top% , width% , height% , path\$)**

Example : SHOW\_IMAGE(0,0,300,300,"d:\PROGRAM\test.bmp")

Description : Several key argument as below:

<b>left %</b>	Fill from the start point of X-axis (pixel).
<b>top %</b>	Fill from the start point of Y-axis (pixel).
<b>width%</b>	Fill the width from the start point (pixel).
<b>height%</b>	Fill the height from the start point (pixel).
<b>path\$</b>	Bitmap file path (Must be on Disk D).

## CLR\_EOL

Purpose : To clear from where the cursor is to the end of the line. The cursor position is not affected after the operation.

Syntax : **CLR\_EOL**

Example : PRINT "TEST BASIC"

LOCATE 3,0

CLR\_EOL

Description : The CLR\_EOL command clears from where the cursor is to the end of the line and then moves the cursor to the original place.



---

## 3.13 Font

This utility “**SDK Tool**” can be used as the following:

When you need a font file for your application, you can make the font file by “SDK Tool”, the font generator can help you making a font file.

### 3.13.1 User font commands

#### **DISPFONT SETFONT**

Purpose : To set user font from font file.

Syntax : ***A%=DISPFONT\_SETFONT(FontID% ,FontPath\$)***

Example : ***A%=DISPFONT\_SETFONT(2,"D:\Fonts\Font16.cft")***

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Set font fail
1	Set font OK

Several key arguments as below:

<b><i>FontID%</i></b>	Font ID (2~9)
<b><i>FontPath\$</i></b>	Font file path

#### **DISPFONT INFO TYPE**

Purpose : To get font type.

Syntax : ***A%=DISPFONT\_INFO\_TYPE(FontID%)***

Example : ***A%=DISPFONT\_INFO\_TYPE(2)***

Description : ***A%*** is an integer variable to be assigned to the result.

***FontID%*** is an integer variable in the range from 2 to 9.

#### **DISPFONT INFO HEIGHT**

Purpose : To get font height.

Syntax : ***A%=DISPFONT\_INFO\_HEIGHT(FontID%)***

Example : ***C%=DISPFONT\_INFO\_HEIGHT(2)***

Description : ***A%*** is an integer variable to be assigned to the result.

***FontID%*** is an integer variable in the range from 2 to 9.

---

## **DISPFONT INFO WIDTH**

Purpose : To get font width.

Syntax : *A%=DISPFONT\_INFO\_WIDTH(FontID %)*

Example : B%=DISPFONT\_INFO\_WIDTH(3)

Description : *A%* is an integer variable to be assigned to the result.  
*FontID%* is an integer variable in the range from 2 to 9.

---

## 3.14 TextBlock

TextBlock is a floating text printing rectangle area on LCD screen. TextBlock defines activated area anywhere within LCD screen display. An out of display area definition is not allowed.

Each TextBlock has individual attribute definition for position, size, font, background color or bmp. There are total 16 TextBlocks. TextBlock(0) is system default block. The setting of TextBlock(0) can't be executed. TextBlock(1~15) are user defined.

### 3.14.1 TextBlock commands

#### **DEFINETEXTBLOCK COLOR**

Purpose : To define the TextBlock setting and the background using default background color or user defined color.

Syntax : ***A%=DEFINETEXTBLOCK\_COLOR***  
***(BlockNo% , FontID% , BGType% ,Color% , Column% , Row% , XPos% , YPos%)***

Example : Orange%=36095  
A%=DEFINETEXTBLOCK\_COLOR(1,0,1,Orange%,6,5,10,30)  
...  
A%=SETTEXTBLOCK(1,0)  
...

Description ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Define TextBlock fail
1	Define TextBlock OK

Several key arguments as below:

<b><i>BlockNo%</i></b>	TextBlock number(1~15)
<b><i>FontID%</i></b>	Defined Font: 0~1: system font 2~9: user font.
<b><i>BGType%</i></b>	If 0 then using default background. If 1 then using user defined background. (PI-1X always set 2 or 3)
<b><i>Color%</i></b>	Background color (PI-1X always set 0)
<b><i>Column%</i></b>	TextBlock column number.
<b><i>Row%</i></b>	TextBlock row number.
<b><i>XPos%</i></b>	TextBlock left-top X position in pixel (0~159).
<b><i>YPos%</i></b>	TextBlock left-top Y position in pixel. StatusBar enable: 0~143. StatusBar disable: 0~159.

#### **DEFINETEXTBLOCK IMAGE**

Purpose : To define the TextBlock setting and the background using bitmap file or default background color.

Syntax : ***A%=DEFINETEXTBLOCK\_IMAGE(BlockNo% ,FontID% , BGType% ,BitmapPath\$ ,Column% ,Row% ,XPos% ,YPos%)***

---

Example : A%=DEFINETEXTBLOCK\_IMAGE(2,0,1,"d:\PROGRAM\5.bmp"  
,8,6,120,30)

Description A% is an integer variable to be assigned to the result.

:

A%	Meaning
0	Define TextBlock fail
1	Define TextBlock OK

Several key arguments as below:

<b>BlockNo%</b>	TextBlock number(1~15)
<b>FontID%</b>	Defined Font: 0~1: system font 2~9: user font.
<b>BGType%</b>	If 0 then using default background. If 1 then using bitmap file..
<b>BitmapPath\$</b>	Bitmap file path
<b>Column%</b>	TextBlock column number.
<b>Row%</b>	TextBlock row number.
<b>XPos%</b>	TextBlock left-top X position in pixel (0~159).
<b>YPos%</b>	TextBlock left-top Y position in pixel. StatusBar enable: 0~143. StatusBar disable: 0~159.

## SETTEXTBLOCK

Purpose : To enable specific TextBlock.

Syntax : A%=SETTEXTBLOCK(BlockNo% ,Save%)

Example : A%=SETTEXTBLOCK(1,0)

Description : A% is an integer variable to be assigned to the result.

A%	Meaning
0	Set TextBlock fail
1	Set TextBlock OK

Several key arguments as below:

<b>BlockNo%</b>	TextBlock number(1~15)
<b>Save%</b>	Save flag to save screen (Save%=1) or not (Save%=0).

## RESETTEXTBLOCK

Purpose : To disable specific TextBlock.

Syntax : RESETTEXTBLOCK(BlockNo%)

Example : RESETTEXTBLOCK(1)

Description : **BlockNo%** is an integer in the range from 1 to 15 indicating TextBlock number.

### **PRINTTEXTBLOCK**

Purpose : To print Text to specific TextBlock.

Syntax : **PRINTTEXTBLOCK**  
(**BlockNo%**, **Column%**, **Row%**, **Str\$**, **FontColor%**)

Example : PRINTTEXTBLOCK(2,5,5,"Hello",0) 'font color is black

Description : Several key arguments as below:

<b>BlockNo%</b>	TextBlock number(0~15)
<b>Column%</b>	TextBlock column number.
<b>Row%</b>	TextBlock row number.
<b>Str\$</b>	Text data.
<b>FontColor%</b>	Text color. (PI-1X always set 0)

### **GETTEXTBLOCKCUR\_X**

Purpose : To get the x coordinate of the current TextBlock position.

Syntax : **A% =GETTEXTBLOCKCUR\_X(BlockNo%)**

Example : PRINT "X=",GETTEXTBLOCKCUR\_X(1)

Description : **A%** is an integer variable to be assigned to the result.

**BlockNo%** is an integer variable in the range from 0 to 15.

### **GETTEXTBLOCKCUR\_Y**

Purpose : To get the y coordinate of the current TextBlock position.

Syntax : **A% =GETTEXTBLOCKCUR\_Y(BlockNo%)**

Example : PRINT "Y=",GETTEXTBLOCKCUR\_Y(1)

Description : **A%** is an integer variable to be assigned to the result.

**BlockNo%** is an integer variable in the range from 0 to 15.

### **SETTEXTBLOCKCUR**

Purpose : To set specific TextBlock as active TextBlock and set position.

Syntax : **SETTEXTBLOCKCUR(BlockNo%, Column%, Row%)**

Example : SETTEXTBLOCKCUR(0,0,0)

Description : Several key arguments as below:

<b>BlockNo%</b>	TextBlock number(0~15)
<b>Column%</b>	TextBlock column number.
<b>Row%</b>	TextBlock row number.

### **SHOWTEXTBLOCKCURSOR**

Purpose : To show or hide TextBlock cursor.

Syntax : **SHOWTEXTBLOCKCURSOR(BlockNo%, Show%, Type%)**

Example : SHOWTEXTBLOCKCURSOR(1,1,1)

---

Description : Several key arguments as below:

<b><i>BlockNo%</i></b>	TextBlock number(0~15)
<b><i>Show%</i></b>	1:Show cursor 0:Hide cursor
<b><i>Type%</i></b>	0: Cursor off. 1: Cursor on, and cursor type is a line as _. 2: Cursor on, and cursor type is a line as  . 3: Cursor on, and cursor type is a block as ■.

## **SWITCHTEXTBLOCK**

Purpose : To switch TextBlock.

Syntax : ***A%=SWITCHTEXTBLOCK(BlockNo%)***

Example : A%=SWITCHTEXTBLOCK(1)

Description : ***A%*** is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Switch fail.
1	Switch success.

***BlockNo%*** is an integer variable in the range from 0 to 15.

---

## 3.15 File manipulation commands

### 3.15.1 Standard Commands

#### *Access mode string    Meaning*

r	Opens file for reading operation only. Error will be returned if target file does not exist.
r+	Opens existing files for both reading and writing operations. Error will be returned if target file does not exist.
w+	Create a file and open it for both reading and writing. If target file does exist, current contents are destroyed.

#### **OPENIN**

Purpose : To open (r mode) a file and get the file for further processing.

Syntax : ***F%*** = ***OPENIN filename\$***

Example : FilePath\$="C:\DATA\Test.DAT"  
fileID%=OPENIN FilePath \$

Description : ***F%*** is an integer variable to assigned to the result.

<b><i>F%</i></b>	<b><i>Meaning</i></b>
0	Open file fail.
Other	Open successfully. It returns the file handle.

***filename\$*** is a string variable indicating the file path.

In case of error, open will return an integer value of 0. You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
1	Filename is a NULL string.
6	<ul style="list-style-type: none"><li>■ Can't create file because the maximum number of files allowed in the system is exceeded.</li><li>■ File path error.</li></ul>



---

## **OPENOUT**

Purpose : To open (w+) a file and get the file for further processing.

Syntax : ***F%=OPENOUT filename\$***

Example : FilePath\$="C:\DATA\Test.DAT"  
fileID%=OPENOUT FilePath\$

Description : ***F%*** is an integer variable to be assigned to the result.

<b><i>F%</i></b>	<b><i>Meaning</i></b>
0	Open file failed.
Other	Open successfully. It returns the file.

***filename\$*** is a string variable indicating the file path.

In case of error, open will return an integer value of 0. You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
1	Filename is a NULL string.
6	<ul style="list-style-type: none"><li>■ Can't create file because the maximum number of files allowed in the system is exceeded.</li><li>■ File path error.</li></ul>

## **OPENUP**

Purpose : To open (r+) a file and get the file for further processing.

Syntax : ***F% = OPENUP filename\$***

Example : FilePath\$="C:\DATA\Test.DAT"  
fileID%=OPENUP FilePath\$

---

Description : **F%** is an integer variable to be assigned to the result.

<b>F%</b>	<b>Meaning</b>
0	Open file failed.
Other	Open successfully. It returns the file.

**filename\$** is a string variable, indicating the file path.

In case of error, open will return an integer value of 0. You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b>GET_FILE_ERROR</b>	<b>Meaning</b>
1	Filename is a NULL string.
6	■ Can't create file because the maximum number of files allowed in the system is exceeded. ■ File path error.

## **MKDIR**

Purpose : To create a folder.

Syntax : **M% = MKDIR foldername\$**

Example : FolderPath\$="C:\ARGOX\"

Result%=MKDIR FolderPath\$

Description : **M%** is an integer variable to be assigned to the result.

<b>M%</b>	<b>Meaning</b>
0	Create folder failed.
1	Create folder succeed.

**foldername\$** is a string variable, indicating the folder path.  
(It is able to create only two level of subfolder)

---

## **RMDIR**

Purpose : To delete a folder.

Syntax : ***R% = RMDIR foldername\$***

Example : FolderPath\$="C:\ARGOX\"

Result%=RMDIR FolderPath\$

Description : ***R%*** is an integer variable to be assigned to the result.

<b><i>R%</i></b>	<b><i>Meaning</i></b>
0	Delete folder failed.
1	Delete folder successfully.

***foldername\$*** is a string variable, indicating the folder path.

## **CLOSE**

Purpose : To close a file.

Syntax : ***CLOSE # F%***

Example : CLOSE # FILEID%

Description : ***F%*** is an integer indicating the file handle.

You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
8	File not opened

## **BGET**

Purpose : To read a byte from a file. The current position is updated after reading.

Syntax : ***STR% = BGET # FILEID%***

Example : STRING1%=BGET # FILEID%

PRINT CHR\$(STRING1%)

Description : ***STR%*** is an integer variable to be returned to the result.

***FILEID%*** is an integer variable indicating the file handle.

You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
7	File not opened

---

## **BGETEXT**

Purpose : To read a specified number of bytes from a file. The current position is updated after reading.

Syntax : ***STR\$ = BGETEXT(N%) # FILEID%***

Example : **STRING1\$=BGETEXT(5)#FILEID%**  
**PRINT STRING1\$**  
**PRINT "STRING LEN=",LEN(STRING1\$)**

Description : ***STR\$*** is a string to be returned to the result.  
***N%*** is an integer indicating the number of bytes to be read.  
***FILEID%*** is an integer variable indicating the file handle.  
You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
7	File not opened

---

## **GET\$**

Purpose : Read a line terminated by a null character “\0” from a file.

Syntax : ***FileData\$ = GET\$ # FILEID%***

Example : WHILE (EOF#FILEID% <> -1)  
          Str\$=GET\$ # FILEID%  
          PRINT Str\$  
          WEND

Description : ***FileData\$*** is a string to be returned to the result.  
***FILEID%*** is an integer variable indicating the file handle.  
You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
7	File not opened

## **BPUT**

Purpose : To write data to a file.

Syntax : ***BPUT # FILEID% , <expr 1> , <expr 2> , ... , <expr n>***

Example : AAA%=566  
          BPUT # FILEID%,STR\$(AAA%),"HELLO"

Description : ***FILEID%*** is an integer variable, indicating the file handle.  
***expr 1 ~ expr n*** is string expression indicating the string data to write to file.  
You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
7	File not opened
10	Not enough memory to write to file.

---

## **EOF**

Purpose : To check if file pointer of a file reaches end of file.

Syntax : ***E%=EOF # FILEID%***

Example : WHILE (EOF#FILEID% <> -1)  
          Str\$=GET\$ # FILEID%  
          PRINT Str\$  
          WEND

Description : ***E%*** is an integer to be assigned to the result.

<b><i>E%</i></b>	<b><i>Meaning</i></b>
0 (False)	Not end-of-file.
-1 (True)	End-of-file

***FILEID%*** is an integer variable indicating the file handle.  
You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
8	File not opened

## **PTR**

Purpose : To get or move the file pointer position of a file.

Syntax : ***TELLPTR% = PTR # FILEID%***  
***PTR # FILEID% = NPTR%***

Example : ...  
          TELLPTR%=PTR # FILEID%  
          ...  
          PTR # FILEID% = 40

Description : ***TELLPTR %*** is an integer variable to be assigned to the result.

TELLPTR% = PTR # FILEID%, to get the file pointer position of a file.

***NPTR %*** is an integer variable indicating the offset bytes address been specified.

***FILEID%*** is an integer variable indicating the file handle.  
You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
------------------------------	-----------------------

2	File specified does not exist.
9	Illegal offset value.
15	New position is beyond end-of-file.

## **EXT**

Purpose : To get or change file length of a file.

Syntax : ***FILESIZE% = EXT # FILEID%***  
***EXT # FILEID% = SIZE%***

Example : ***FILESIZE%=EXT # FILEID%***  
***PRINT FILESIZE%***

...

***EXT # FILEID% = 20***

Description : ***FILESIZE%*** is an integer variable to be returned the file length.

***SIZE%*** is an integer variable indicating the length to be changed of the file.

***FILEID%*** is an integer variable indicating the file handle.

You can use the GET\_FILE\_ERROR command to get the file error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILE_ERROR</i></b>	<b><i>Meaning</i></b>
2	File specified does not exist.
8	File not opened

## **GET FILE ERROR**

Purpose : To get the file error code.

Syntax : ***A%=GET\_FILE\_ERROR***

Example : ***A%=GET\_FILE\_ERROR***  
***PRINT "File error code:",A%***

Description : ***A%*** is an integer to be assigned to the result. If there is no error, it returns 0. If it returns a value other than 0, it's file error code.

---

### 3.15.2 DBMS Commands

#### **DBMS INIT SEARCH**

Purpose : To initiate the file search in disk.

Syntax : *A%=DBMS\_INIT\_SEARCH(FilePath\$, DBMSID%, S\$,  
N1%,N2%,N3%)*

Example : Result%=DBMS\_INIT\_SEARCH("C:\DATA\fix.DAT",1,"5,6,6",0  
,17,3)

Description : *A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	DBMS initialization fail
1	DBMS initialization OK
5	Open file error
6	The DBMS ID is illegal.
7	DBMS ID already used.
8	The record type is illegal.
9	The field number exceeds 20.

Several key arguments as below:

<i>FilePath\$</i>	DBMS file path
<i>DBMSID%</i>	DBMS ID (1~10)
<i>S\$</i>	It needs to insert the unsigned char array; the array represents the length of every field.
<i>N1%</i>	It has no separate symbols between different fields. (now we only can set 0)
<i>N2%</i>	This argument is each record's length. Needs to insert this value, not including the symbol of line feed.
<i>N3%</i>	This argument is the field's quantity of each record (1~20).

#### **DBMS INIT SEARCHADV**

Purpose : To initiate the advance file search in disk.

Syntax : *A%=DBMS\_INIT\_SEARCHADV(FilePath\$, DBMSID%, S1\$,  
S2\$, N1%,N2%,N3%,N4%)*

Example : Result%=DBMS\_INIT\_SEARCHADV("C:\DATA\fix.DAT",1,"5,  
6,6","1,3",2,0,17,3)

Description : This command can initialize a work of advance searching file. After inserting every argument, you can to search files.

When using this command to initial a DBMS search, you have to



---

take care for:

1. This command cannot support Variable field length search.
2. When initial, we will make a index file in C disk, so it has to take a few time.
3. The index filename will be similar to origin file. For example, the lookup file name is “AAA.txt”, the index filename will be “AAA.idx”. So, you have to check the duplicate filename to avoid error for making index file.
4. You have to reserve some space for the function to make index file in C disk.

**A%** is an integer variable to be assigned to the result.

<b>A%</b>	<b>Meaning</b>
0	DBMS initialization fail
1	DBMS initialization OK
5	Open file error
6	The DBMS ID is illegal.
7	DBMS ID already used.
8	The record type is illegal.
9	The field number exceeds 20.
-1	Argument S2\$ or N1% is error, please check it.
-2	Cannot make a IDX file, please check your lookup filename or C disk size.

Several key arguments as below:

<b>FilePath\$</b>	DBMS file path
<b>DBMSID%</b>	DBMS ID (1~10)
<b>S1\$</b>	It needs to insert the unsigned char array; the array represents the length of every field.
<b>S2\$</b>	This argument can give max. 8 key fields for search. We will make a checksum index file for these key fields.
<b>N1%</b>	This argument can give the sum of key fields size.
<b>N2%</b>	It has no separate symbols between different fields. (now we only can set 0)
<b>N3%</b>	This argument is each record's length.

	Needs to insert this value, not including the symbol of line feed.
<i>N4%</i>	This argument is the field's quantity of each record (1~20).

### **DBMS\_CLOSE\_SEARCH**

Purpose : To close the file search in disk.

Syntax : ***DBMS\_CLOSE\_SEARCH(DBMSID%)***

Example : DBMS\_CLOSE\_SEARCH(1)

Description : ***DBMSID%*** is an integer variable in the range from 1 to 10.

### **DBMS\_APPEND\_DATA**

Purpose : To increase one record on the file end.

Syntax : ***DBMS\_APPEND\_DATA(DBMSID%,data\$)***

Example : data\$ = "Happy, TEST, DBMS"  
DBMS\_APPEND\_DATA(1,data\$)

Description : ***DBMSID%*** is an integer variable in the range from 1 to 10.  
***data\$*** is a string variable indicating the data of record introduced.

### **DBMS\_DEL\_DATA**

Purpose : To delete the appointed record in the file.

Syntax : ***DBMS\_DEL\_DATA(DBMSID%,record%)***

Example : DBMS\_DEL\_DATA(1,2)

Description : ***DBMSID%*** is an integer variable in the range from 1 to 10.  
***Record%*** is an integer variable indicating the appointed record to be deleted.

### **DBMS\_EMPTY**

Purpose : To remove all records in the file.

Syntax : ***DBMS\_EMPTY(DBMSID%)***

Example : DBMS\_EMPTY(1)

Description : ***DBMSID%*** is an integer variable in the range from 1 to 10.

### **DBMS\_FIND\_RECORD**

Purpose : To search the designated field.

Syntax : ***A%=DBMS\_FIND\_RECORD(DBMSID% , field% , key\$)***

Example : A% = DBMS\_FIND\_RECORD(1, 2, " TEST3")  
PRINT A%

Description : *A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	Search defeat.
Other value	Match the record position of data

Several key arguments as below:

<i>DBMSID%</i>	DBMS ID (1~10)
<i>field%</i>	Search wanted field.
<i>key \$</i>	Match wanted string data.

※This command only supports forward search.

## **DBMS\_FIND\_RECORD\_B**

Purpose : To search the designated field.

Syntax : *A%=DBMS\_FIND\_RECORD\_B(DBMSID% , field% , key\$)*

Example : *A% = DBMS\_FIND\_RECORD\_B(1, 2, " TEST3")*  
*PRINT A%*

Description : *A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	Search defeat.
Other value	Match the record position of data

Several key arguments as below:

<i>DBMSID%</i>	DBMS ID (1~10)
<i>field%</i>	Search wanted field.
<i>key \$</i>	Match wanted string data.

※This command only supports backward search.

## **DBMS\_GET\_COUNT**

Purpose : To obtain the figure of all records in the file.

Syntax : *A%=DBMS\_GET\_COUNT(DBMSID%)*

Example : *A% = DBMS\_GET\_COUNT(1)*  
*PRINT A%*

Description : *A%* is an integer variable to be assigned to the result.

*DBMSID%* is an integer variable in the range from 1 to 10.

## **DBMS\_GET\_DATA\$**

Purpose : To read the data of appointed field in the appointed record.

Syntax : *A\$=DBMS\_GET\_DATA\$(DBMSID% , record% , field%)*

Example : *A\$ = DBMS\_GET\_DATA\$(1, 3, 3)*  
*PRINT A\$*

Description : *A\$* is a string variable to be assigned to the result.

Several key arguments as below:

<i>DBMSID%</i>	DBMS ID (1~10)
----------------	----------------

<i>record %</i>	Read record position.
<i>field %</i>	Read field position.

## **DBMS\_UPDATE\_DATA**

Purpose : To revise the data of appoint field in appointed field record.

Syntax : ***DBMS\_UPDATE\_DATA(DBMSID%, record%, field%, key\$)***

Example : DBMS\_UPDATE\_DATA(1, 3, 3, "SONG")

Description : Several key arguments as below:

<b><i>DBMSID%</i></b>	DBMS ID (1~10)
<i>record %</i>	Read record position.
<i>field %</i>	Read field position.
<i>key\$</i>	Update string data wanted.

## **DBMS\_SEARCH\_FIELD**

Purpose : To search the designated field.

Syntax : ***A%=DBMS\_FIND\_RECORD(DBMSID% , field% , record% , key\$, flag%)***

Example : A% = DBMS\_FIND\_RECORD(1, 2, 3, " TEST3", 1)  
PRINT A%

Description : A% is an integer variable to be assigned to the result.

<b><i>A%</i></b>	<b><i>Meaning</i></b>
0	Search defeat.
Other value	Match the record position of data

Several key arguments as below:

<b><i>DBMSID%</i></b>	DBMS ID (1~10)
<b><i>field%</i></b>	Search wanted field.
<b><i>record%</i></b>	Search wanted record.
<b><i>key \$</i></b>	Match wanted string data.
<b><i>flag%</i></b>	Search from forward or backward. 1 => Search from forward to backward 2 => Search from backward to forward

---

## 3.16 Vibrator commands

### VIBRATOR\_TIMER

Purpose : To set or get the vibrator timer.

Syntax : ***A% = VIBRATOR\_TIMER***  
***VIBRATOR\_TIMER = X%***

Example : VIBRATOR\_TIMER=5

...

PRINT "Vibrator timer:",VIBRATOR\_TIMER

Description : ***A%*** is an integer variable to be assigned as the vibrator timer.  
***X%*** is an integer variable indicating a period of time in units of 100ms.

### VIBRATOR

Purpose : To set the vibrator on/off.

Syntax : ***VIBRATOR(N%)***

Example : VIBRATOR(1)      'Vibrator on

'Wait 0.5 sec

WAIT(100)

VIBRATOR(0)      'Vibrator off

Description : ***N%*** is an integer variable indicating vibrator on or off.

<b><i>N%</i></b>	<b><i>Meaning</i></b>
0	Vibrator off
1	Vibrator on

---

## 3.17 Communication port commands

### CLOSE\_COM

Purpose : To terminate communication and disable a specified COM port.

Syntax : **CLOSE\_COM** (*N%*)

Example : CLOSE\_COM(1)

Description : *N%* is an integer indicating which COM port is to be disabled (now we only can choose 1).

### OPEN\_COM

Purpose : To enable a specified COM port and initialize communication.

Syntax : **OPEN\_COM** (*N%*)

Example : OPEN\_COM(1)

Description : *N%* is an integer variable indicating which COM port is to be enabled (now we only can choose 1).

### SET\_COM

Purpose : To set parameters of a specified COM port.

Syntax : **SET\_COM**(*N%*, *Baudrate%*, *Parity%*, *Data%*, *Handshake%*)

Example : SET\_COM(1, 1, 1, 2, 1)

Description : Several key arguments as below:

<b><i>N%:</i></b>	1: RS-232 (now we only can choose 1)
<b><i>Baudrate%:</i></b>	Baud rate
	1: 115200                      2-3: 57600
	4: 38400                      5: 19200
	6: 9600                      7-8: 4800
<b><i>Parity%:</i></b>	Parity
	1:None                      2:Odd
	3:Even
<b><i>Data%:</i></b>	Data bits
	1: 7 bits                      2: 8 bits
<b><i>Handshake%:</i></b>	Flow control
	1: None
	2: Auto Flow control

---

## READ\_COM\$

Purpose : To read data from a specified COM port.

Syntax : *A\$ = READ\_COM\$(N%)*

Example : ON COM(1) GOSUB READ1

CLS

PRINT "==COM TEST=="

LOCATE 0,1

PRINT "ENT TO WRITE"

SET\_COM(1,1,1,2,1)

OPEN\_COM(1)

CLEAR\_COM(1)

SET\_RTS(1,1)

LOOP2:

IF INKEY\$="" THEN

GOTO LOOP2

END IF

CLOSE\_COM(1)

END

READ1:

A\$=READ\_COM\$(1)

PRINT A\$

RETURN

Description :

*A\$* is a string variable to be assigned to the data.

*N%* is an integer variable indicating which COM port the data is to be read (now we only can choose 1).

If the receiver buffer is empty, an empty string will be returned.

---

## WRITE\_COM

Purpose : To send a string to the host through a specified COM port.

Syntax : **WRITE\_COM(N%, A\$)**

Example : CLS

```
PRINT "===COM TEST==="
```

```
PRINT "ENT TO WRITE"
```

```
SET_COM(1,1,1,2,1)
```

```
OPEN_COM(1)
```

```
WHILE INKEY$<>CHR$(13)
```

```
WEND
```

```
STR1$="Hello!!"
```

```
WHILE GET_CTS(1)=0
```

```
WEND
```

```
WRITE_COM(1,STR1$)
```

```
...
```

```
CLOSE_COM(1)
```

```
END
```

Description : *N%* is an integer variable indicating which COM port the data is to be sent to (now we only can choose 1).

*A\$* is a string variable indicating the string to be sent.

## GET\_CTS

Purpose : To get CTS level.

Syntax : **A% = GET\_CTS(N%)**

Example : PRINT "CTS Status:",GET\_CTS(1)

Description : *A%* is an integer variable to be assigned to the result.

<i>A%</i>	<i>Meaning</i>
0	Negated (Space)
1	Asserted (Mark)

*N%* is an integer variable indicating which COM port to get CTS level (now we only can choose 1).



---

## **SET RTS**

Purpose : To set RTS level.

Syntax : **SET\_RTS(N1%, N2%)**

Example : SET\_RTS(1, 1)

Description : **N1%** is an integer variable indicating which COM port to set RTS level (now we only can choose 1).

**N2%** is an integer variable indicating the RTS state.

<b>N2%</b>	<b>Meaning</b>
0	Negated (Space)
1	Asserted (Mark)

## **CLEAR COM**

Purpose : To clear receiver buffer.

Syntax : **CLEAR\_COM(N%)**

Example : CLEAR\_COM(1)

Description : **N%** is an integer variable indicating which COM port to clear receive buffer (now we only can choose 1).

## **COM DELIMITER**

Purpose : To change delimiter of sending and receiving string for a specified COM port.

Syntax : **COM\_DELIMITER(N%, C%)**

Example : COM\_DELIMITER(1,13)      'use carriage return as delimiter

COM\_DELIMITER(1,38)      'use '&' character as delimiter

COM\_DELIMITER(1,-1)      'no delimiter

Description : **N%** is an integer variable indicating which COM port is to be set (now we only can choose 1).

**C%** is an integer variable indicating the ASCII code of the delimiter character, in the range from 0 to 255. If it is other value, no delimiter will be applied.

The default COM\_DELIMITER is 0xd.

---

## FILE\_TRANS

Purpose : Using FILE\_TRANS to upload or download files.

Syntax : **FILE\_TRANS**

Example : FILE\_TRANS



Description : The FILE\_TRANS command provides the transmission environment to link with Voler/Everlink and make file uploading or downloading.

Pressing ESC key can quit the transmission operation.

## FILE\_TRANS\_REALTIME

Purpose : Using FILE\_TRANS\_REALTIME to upload or download files immediately.

Syntax : **FILE\_TRANS\_REALTIME(N%)**

Example : FILE\_TRANS\_REALTIME(1)

Description : N% is an integer variable indicating the transmission state.

<i>N%</i>	<i>Meaning</i>
0	Transmission, not real-time.
1	Real-time transmission.

## FILE\_TRANS\_BAUD

Purpose : To get or set the transmission baud rate.

Syntax : **A% = FILE\_TRANS\_BAUD**

**FILE\_TRANS\_BAUD = X%**

Example : N%=FILE\_TRANS\_BAUD

...

FILE\_TRANS\_BAUD=2      'baud rate is 38400 bps

Description : *A%* is an integer variable to be assigned for the transmission baud rate.

*X%* is an integer variable indicating baud rate to be set.

<b><i>FILE_TRANS_BAUD</i></b>	<b><i>Baud rate (bps)</i></b>
0	115200
1	57600
2	38400
3	19200
4	9600
5	4800

You can use the GET\_FILETRANS\_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILETRANS_ERROR</i></b>	<b><i>Meaning</i></b>
-1	Set OK.
-2	Selected LinkingPort is using.
-4	Parameter error.

## **FILE TRANS INTERFACE**

Purpose : To get or set the transmission interface.

Syntax : *A% = FILE\_TRANS\_INTERFACE*  
*FILE\_TRANS\_INTERFACE = X%*

Example : N%=FILE\_TRANS\_INTERFACE

...

FILE\_TRANS\_INTERFACE=1      'RS-232

Description : *A%* is an integer variable to be assigned for the transmission interface.

*X%* is an integer variable indicating interface to be set.

<b><i>FILE_TRANS_INTERFACE</i></b>	<b><i>Interface</i></b>
0	None
1	RS-232
2	USB
3	BT
4	WIFI

You can use the GET\_FILETRANS\_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

<b><i>GET_FILETRANS_ERROR</i></b>	<b><i>Meaning</i></b>
-----------------------------------	-----------------------

---

-1	Set OK.
-2	Selected LinkingPort is using.
-4	Parameter error.

## **FILE TRANS GETBT**

Purpose : Get transmission Bluetooth information.

Syntax : ***A\$=FILE\_TRANS\_GETBT\$***

Example : S1\$=FILE\_TRANS\_GETBT\$

LocalAddress\$=LEFT\$(S1\$,16)

PRINT "LocAdd: ";LocalAddress\$

LocalName\$=MID\$(S1\$,17,20)

PRINT "LocName: ";LocalName\$

LocalSec%=ASC(MID\$(S1\$,45,4))

PRINT "LocalSec: ";LocalSec%

LocalEnc%=ASC(MID\$(S1\$,49,4))

PRINT "LocalEnc: ";LocalEnc%

LocalTimeout%=ASC(MID\$(S1\$,37,4))

PRINT "LocalTimeout: ";LocalTimeout%

LocalRes%=ASC(MID\$(S1\$,42,4))

PRINT "LocalRes: ";LocalRes%

LinkAddress\$=MID\$(S1\$,53,16)

PRINT "LinkAddress: ";LinkAddress\$

PinCode\$=MID\$(S1\$,69,20)

PRINT "PinCode: ";PinCode\$

...

Description : Use this command can get transmission's Bluetooth settings.

A\$ is a string variable indicating the PI-1010/1030 Bluetooth information. Format of string as show below:

<i>A\$(Length)</i>	<i>Meaning</i>
<b>1~16</b>	PI-1X Bluetooth MAC address.(Cannot change.)
<b>17~36</b>	PI-1X Bluetooth device name
<b>45~48</b>	PI-1X Bluetooth inquiry timeout, the value from 1(1.28 seconds) to 48(61.44 seconds).
<b>49~52</b>	PI-1X Bluetooth inquiry max response, the value from 1 to 10.
<b>37~40</b>	PI-1X Bluetooth security mode, if 1(on) else 0(off)
<b>41~44</b>	PI-1X Bluetooth encryption mode, if 1(on) else 0(off)
<b>53~68</b>	To linking device address.
<b>69~88</b>	PIN code.

## **FILE\_TRANS\_GETWIFI**

Purpose : Get transmission WIFI information.

Syntax : **A\$=FILE\_TRANS\_GETWIFI\$**

Example : ...

```
S1$=FILE_TRANS_GETWIFI$
Dhcp%=ASC(LEFT$(S1$,4))
PRINT "Dhcp:";Dhcp%
LOCATE 1,4
IpAddress$=MID$(S1$,5,20)
PRINT "IpAdd:";IpAddress$    ...
```

Description : Use this command can get transmission's WIFI settings.

A\$ is a string variable indicating the PI-1030 WIFI information. Format of string as show below:

<i>A\$(Length)</i>	<i>Meaning</i>
<b>1~4</b>	PI-1X WIFI Dhcp mode.
<b>5~20</b>	PI-1X WIFI IP address.
<b>21~36</b>	PI-1X WIFI subnet mask.
<b>37~52</b>	PI-1X WIFI gateway.
<b>53~88</b>	WIFI accesspoint SSID

	name.
<b>89~92</b>	PI-1X WIFI TX power.
<b>93~96</b>	PI-1X WIFI power saving mode.
<b>97~100</b>	WIFI security key type.
<b>101~116</b>	TCP connect IP address.
<b>117~120</b>	TCP connect port.
<b>121~184</b>	WIFI security key.
<b>185~196</b>	WIFI module MAC address.(Read Only)
<b>197~202</b>	WIFI module F/W version. (Read Only)

## **FILE TRANS SETBT**

Purpose : Set transmission Bluetooth information.

Syntax : **FILE\_TRANS\_SETBT(S1\$,N1%,N2%,N3%,N4%,S2\$,S3\$)**

Example : FILE\_TRANS\_SETBT(LocalName\$,1,1,3,10,DeviceAddress\$,PIN\$)

Description : Several key arguments as below:

<b>S1\$</b>	PI-1X Bluetooth device name(Allow 1~16 characters)
<b>N1%</b>	PI-1X Bluetooth security mode, set 1(on) or 0(off)
<b>N2%</b>	PI-1X Bluetooth encryption mode, set 1(on) or 0(off)
<b>N3%</b>	PI-1X Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).
<b>N4%</b>	PI-1X Bluetooth inquiry max response, the value from 1 to 10.
<b>S2\$</b>	Set link device address(Allow 1~12 characters)
<b>S3\$</b>	Set PIN code(Allow 4~16 characters)

You can use the GET\_FILETRANS\_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

<b>GET_FILETRANS_ERROR</b>	<b>Meaning</b>
-1	Set OK.
-2	Selected LinkingPort is using.
-4	Parameter error.

## **FILE TRANS SETWIFI**

Purpose : Set transmission WIFI information.

Syntax : **FILE\_TRANS\_SETWIFI**(*N1%*,*S1\$*,*S2\$*,*S3\$*,*S4\$*,*N2%*,*N3%*,*S5\$*,*N4%*,*S6\$*)

Example : FILE\_TRANS\_SETWIFI(0,IP\$,MK\$,GW\$,SSID\$,1,2,  
CONNIP\$,PORT\$,KEY\$)  
FILE\_TRANS\_SETWIFI(1,"","","", SSID\$,1,2,  
CONNIP\$,PORT\$,KEY\$) 'Use DHCP

Description : Several key arguments as below:

<i>N1%</i>	PI-1X WIFI Dhcp mode, set 1(enable) or 0(disable)
<i>S1\$</i>	WIFI module IP address.(xxx.xxx.xxx.xxx)
<i>S2\$</i>	WIFI module subnet mask.(xxx.xxx.xxx.xxx)
<i>S3\$</i>	WIFI module gateway.(xxx.xxx.xxx.xxx)
<i>S4\$</i>	WIFI accesspoint SSID name.
<i>N2%</i>	PI-1X WIFI TX power, set 0(Low) 1(Medium) 2(High)
<i>N3%</i>	PI-1X WIFI security key type, set 0(disable) 1(WEP) or 2(WPA2)
<i>S5\$</i>	Remote TCP connect IP address.(xxx.xxx.xxx.xxx)
<i>N4%</i>	Remote TCP connect port. (allowed range 1024 to 49151)
<i>S6\$</i>	WIFI security key. (1~63 characters)

You can use the GET\_FILETRANS\_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

<b>GET_FILETRANS_ERROR</b>	<b>Meaning</b>
-1	Set OK.
-2	Selected LinkingPort is using.
-4	Parameter error.

## **FILE\_TRANS\_SENDMSG**

Purpose : Send the message to PC.

Syntax : **FILE\_TRANS\_SENDMSG**(*S1\$*, *N1%*,*S2\$*)

Example : FILE\_TRANS\_SENDMSG("Send MSG!", 1, EID\$) ' message type  
FILE\_TRANS\_SENDMSG(ScanData\$, 0, "") 'barcode data type

Description : Several key arguments as below:

<i>S1\$</i>	Send data string.
<i>N1%</i>	Send data type. set 1(message) or 0(barcode data)
<i>S2\$</i>	Target terminal EID.

You can use the GET\_FILETRANS\_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

---

<b><i>GET_FILETRANS_ERROR</i></b>	<b><i>Meaning</i></b>
1	Send OK.
0	Send false.

## **GET FILETRANS ERROR**

Purpose : To get the FILE\_TRANS error code.

Syntax : ***N%=GET\_FILETRANS\_ERROR***

Example : N%=GET\_FILETRANS\_ERROR

Description : ***N%*** is an integer to be assigned to the result.



---

## 3.18 Memory commands

### RAM\_SIZE

- Purpose : To check the total space in disk C.
- Syntax : **RAMSIZE% = RAM\_SIZE**
- Example : PRINT "RAM\_SIZE=",RAM\_SIZE
- Description : **RAMSIZE%** is an integer variable to be assigned for the total space in disk C.

### ROM\_SIZE

- Purpose : To check the total space in disk D.
- Syntax : **ROMSIZE% = ROM\_SIZE**
- Example : PRINT "ROM\_SIZE=",ROM\_SIZE
- Description : **ROMSIZE%** is an integer variable to be assigned for the total space in disk D.

### SD\_SIZE

- Purpose : To check the total space in disk E.
- Syntax : **SDSIZE% = SD\_SIZE**
- Example : PRINT "SD\_SIZE=",SD\_SIZE
- Description : **SDSIZE%** is an integer variable to be assigned for the total space in disk E.

### FREE MEMORY

- Purpose : To check the free space in disk C/ D/ E.
- Syntax : **FREESIZE% = FREE\_MEMORY(N%)**
- Example : PRINT "Free on disk C:";FREE\_MEMORY(0)  
PRINT "Free on disk D:";FREE\_MEMORY(1)  
...
- Description : **FREESIZE%** is an integer variable to be assigned for the free space in disk C(N%=0) or disk D (N%=1) or disk E (N%=2).

### DISK\_USEDSIZE

- Purpose : To check the occupied space in disk C/ D/ E.
- Syntax : **USED\_SIZE% = DISK\_USEDSIZE(N%)**
- Example : PRINT "USED C SIZE:",DISK\_USEDSIZE(0)  
PRINT "USED D SIZE:",DISK\_USEDSIZE(1)
- Description : **USED\_SIZE%** is an integer variable to be assigned for the occupied space in disk C (N%=0) or disk D (N%=1) or disk E (N%=2).

---

## 3.19 USB commands

### USB\_OPEN

- Purpose : To initialize and enable USB port.
- Syntax : ***USB\_OPEN***
- Example : USB\_OPEN
- Description : Using USB\_OPEN command can initialize and enable the USB port.

### USB\_CLOSE

- Purpose : To close the USB port.
- Syntax : ***USB\_CLOSE***
- Example : USB\_CLOSE
- Description : Using USB\_CLOSE command can disable and suspend the USB port.

### USB\_READ\$

- Purpose : To read specific number of bytes from USB port.
- Syntax : ***A\$=USB\_READ\$(N%)***
- Example : KEY\$=USB\_READ\$(1)
- Description : ***A\$*** is a string variable to be assigned to the data.  
***N%*** is an integer variable indicating number of bytes to be read from USB port.

### USB\_WRITE

- Purpose : To write specific number of bytes to the PC side.
- Syntax : ***USB\_WRITE(A\$, N%)***
- Example : USB\_WRITE(A\$,100)
- Description : ***A\$*** is a string variable indicating the data is to be sent.  
***N%*** is an integer variable indicating number of bytes to be written to USB port.

---

## 3.20 LinkingPort commands

### LINKPORT\_OPEN

Purpose : Start a LinkingPort.

Syntax : *N1%=LINKPORT\_OPEN(N2%)*

Example : Result%=LINKPORT\_OPEN(Port%)

...

Result%=LINKPORT\_CLOSE(Port%)

Description : Use this command can start a LinkingPort. Before use this command, you have to set LinkingPort's setting by using "LINKPORT\_SETxxx" command.

*N1%* is an integer variable to be assigned to the result.

<i>N1%</i>	<i>Meaning</i>
-1	Open LinkingPort success.
-2	Selected LinkingPort is using.
-3	Selected LinkingPort's connect interface is using.
-4	Parameter error.
-7	LinkingPort is not set.

*N2%* is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

### LINKPORT\_COSE

Purpose : Stop a LinkingPort.

Syntax : *N1%=LINKPORT\_CLOSE(N2%)*

Example : Result%=LINKPORT\_OPEN(Port%)

...

Result%=LINKPORT\_CLOSE(Port%)

Description : Use this command can stop a LinkingPort.

*N1%* is an integer variable to be assigned to the result.

<i>N1%</i>	<i>Meaning</i>
-1	Close LinkingPort success.
-4	Parameter error.
-6	LinkingPort is not open.
-7	LinkingPort is not set.

*N2%* is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

---

## **LINKPORT\_SELECTIF**

Purpose : Set LinkingPort interface select setting.

Syntax : ***N1%=LINKPORT\_SELECTIF(N2%,N3%)***

Example : Result%=LINKPORT\_SELECTIF(Port%, Interface%)

Description : Use this command can select a LinkingPort's interface. Before use this command, you have to close LinkingPort.

***N1%*** is an integer variable to be assigned to the result.

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
-1	Set LinkingPort interface success.
-2	Selected LinkingPort is using.
-4	Parameter error.

***N2%*** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

***N3%*** is an integer variable indicating which interface is to be selected.

<b><i>N3%</i></b>	<b><i>Meaning</i></b>
0	None.
1	RS232.
2	USB.
3	Bluetooth
4	WIFI

## **LINKPORT\_GETIF**

Purpose : Get LinkingPort interface select setting.

Syntax : ***N1%=LINKPORT\_GETIF(N2%)***

Example : Result%=LINKPORT\_GETIF(Port%)

Description : Use this command can get a LinkingPort interface.

***N1%*** is an integer variable to be assigned to the result.

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
0	None.
1	RS232.
2	USB.
3	Bluetooth
4	WIFI
-4	Parameter error.

***N2%*** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

---

## **LINKPORT\_SETCOM**

Purpose : Set LinkingPort COM baudrate setting.

Syntax : ***N1%=LINKPORT\_SETCOM(N2%, N3%)***

Example : Result%=LINKPORT\_SETCOM(Port%, Baud%)

Description : Use this command can set LinkingPort's COM baudrate. Before use this command, you have to close LinkingPort.

***N1%*** is an integer variable to be assigned to the result.

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
-1	Success.
-2	Selected LinkingPort is using.
-4	Parameter error.

***N2%*** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

***N3%*** is an integer variable indicating LinkingPort's COM baudrate. The value is form 0 to 5.

<b><i>N3%</i></b>	<b><i>Meaning</i></b>
0	115200 bps
1	57600 bps
2	38400 bps
3	19200 bps
4	9600 bps
5	4800 bps

## **LINKPORT\_GETCOM**

Purpose : Get LinkingPort COM baudrate seting.

Syntax : ***N1%=LINKPORT\_GETCOM(N2%)***

Example : Result%=LINKPORT\_GETCOM(Port%)

Description : Use this command can get LinkingPort's baudrate.

***N1%*** is an integer variable to be assigned to the result.

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
0	115200 bps
1	57600 bps
2	38400 bps
3	19200 bps
4	9600 bps
5	4800 bps
-4	Parameter error.

---

**N2%** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

### **LINKPORT SETBT**

Purpose : Set LinkingPort Bluetooth function setting.

Syntax : **N1%=LINKPORT\_SETBT(N2%, S1\$, N3%, N4%, N5%, N6%, S2\$, S3\$)**

Example : **Result%=LINKPORT\_SETBT(Port%, LocalName\$, 1, 1, 3, 10, DeviceAddress\$, PIN\$)**

Description : Use this command can set LinkingPort's BT setting. Before use this command, you have to close LinkingPort.

**N1%** is an integer variable to be assigned to the result.

<b>N1%</b>	<b>Meaning</b>
-1	Success.
-2	Selected LinkingPort is using.
-4	Parameter error.

**N2%** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

Several key arguments as below:

<b>S1\$</b>	PI-1X Bluetooth device name(Allow 1~16 characters)
<b>N3%</b>	PI-1X Bluetooth security mode, set 1(on) or 0(off)
<b>N4%</b>	PI-1X Bluetooth encryption mode, set 1(on) or 0(off)
<b>N5%</b>	PI-1X Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).
<b>N6%</b>	PI-1X Bluetooth inquiry max response, the value from 1 to 10.
<b>S2\$</b>	Set link device address(Allow 1~12 characters)
<b>S3\$</b>	Set PIN code(Allow 4~16 characters)

### **LINKPORT GETBT**

Purpose : Get LinkingPort Bluetooth function setting.

Syntax : **A\$=LINKPORT\_GETBT\$(N1%)**

Example : **S1\$=LINKPORT\_GETBT\$(Port%)**

**LocalAddress\$=LEFT\$(S1\$,16)**

**PRINT "LocAdd:";LocalAddress\$**

**LocalName\$=MID\$(S1\$,17,20)**

---

```
PRINT "LocName:";LocalName$
```

```
LocalSec%=ASC(MID$(S1$,45,4))
```

```
PRINT "LocalSec:";LocalSec%
```

```
LocalEnc%=ASC(MID$(S1$,49,4))
```

```
PRINT "LocalEnc:";LocalEnc%
```

```
LocalTimeout%=ASC(MID$(S1$,37,4))
```

```
PRINT "LocalTimeout:";LocalTimeout%
```

```
LocalRes%=ASC(MID$(S1$,42,4))
```

```
PRINT "LocalRes:";LocalRes%
```

```
LinkAddress$=MID$(S1$,53,16)
```

```
PRINT "LinkAddress:";LinkAddress$
```

```
PinCode$=MID$(S1$,69,20)
```

```
PRINT "PinCode:";PinCode$
```

```
...
```

Description : *A\$* is a string variable indicating the PI-1010/1030 Bluetooth information.Format of string as show below:

<i><b>A\$(Length)</b></i>	<i><b>Meaning</b></i>
<i><b>1~16</b></i>	PI-1X Bluetooth MAC address.(Cannot change.)
<i><b>17~36</b></i>	PI-1X Bluetooth device name
<i><b>45~48</b></i>	PI-1X Bluetooth inquiry timeout, the value from 1(1.28 seconds) to 48(61.44 seconds).
<i><b>49~52</b></i>	PI-1X Bluetooth inquiry max response, the value from 1 to 10.
<i><b>37~40</b></i>	PI-1X Bluetooth security mode, if 1(on) else 0(off)
<i><b>41~44</b></i>	PI-1X Bluetooth encryption mode, if 1(on) else 0(off)
<i><b>53~68</b></i>	To linking device address.
<i><b>69~88</b></i>	PIN code.

*NI%* is an integer variable indicating LinkingPort's port number.  
The value is form 1 to 4.

## LINKPORT\_SETWIFI

Purpose : Set LinkingPort WIFI setting.

Syntax : ***N1%=LINKPORT\_SETWIFI(N2%,N3%,S1\$,S2\$,S3\$,S4\$,N4%,N5%, S5\$,N6%,S6\$)***

Example : LINKPORT\_SETWIFI(Port%,0,IP\$,MK\$,GW\$,SSID\$,1,2,  
CONNIP\$,PORT%,KEY\$)  
LINKPORT\_SETWIFI(Port%,1,"","","", SSID\$,1,2,  
CONNIP\$,PORT%,KEY\$) 'Use DHCP

Description : Use this command can set LinkingPort's WIFI setting. Before use this command, you have to close LinkingPort.

***N1%*** is an integer variable to be assigned to the result.

<b><i>N1%</i></b>	<b><i>Meaning</i></b>
-1	Success.
-2	Selected LinkingPort is using.
-4	Parameter error.

***N2%*** is an integer variable indicating LinkingPort's port number.

The value is form 1 to 4.

Several key arguments as below:

<b><i>N3%</i></b>	PI-1X WIFI Dhcp mode, set 1(enable) or 0(disable)
<b><i>S1\$</i></b>	WIFI module IP address.(xxx.xxx.xxx.xxx)
<b><i>S2\$</i></b>	WIFI module subnet mask.(xxx.xxx.xxx.xxx)
<b><i>S3\$</i></b>	WIFI module gateway.(xxx.xxx.xxx.xxx)
<b><i>S4\$</i></b>	WIFI accesspoint SSID name.
<b><i>N4%</i></b>	PI-1X WIFI TX power, set 0(Low) 1(Medium) 2(High)
<b><i>N5%</i></b>	PI-1X WIFI security key type, set 0(disable) 1(WEP) or 2(WPA2)
<b><i>S5\$</i></b>	Remote TCP connect IP address.(xxx.xxx.xxx.xxx)
<b><i>N6%</i></b>	Remote TCP connect port. (allowed range 1024 to 49151)
<b><i>S6\$</i></b>	WIFI security key. (1~63 characters)

## LINKPORT\_GETWIFI

Purpose : Get LinkingPort WIFI setting.

Syntax : ***A\$=LINKPORT\_GETWIFI\$(N1%)***

Example : ...

```
S1$=LINKPORT_GETWIFI$  
Dhcp%=ASC(LEFT$(S1$,4))  
PRINT "Dhcp: ";Dhcp%  
LOCATE 1,4
```



```

IpAddress$=MID$(S1$,5,20)
PRINT "IpAdd:";IpAddress$    ...

```

Description : Use this command can get LinkingPort's WIFI settings.

*A\$* is a string variable indicating the PI-1030 WIFI information.Format of string as show below:

<i>A\$(Length)</i>	<i>Meaning</i>
<i>1~4</i>	PI-1X WIFI Dhcp mode.
<i>5~20</i>	PI-1X WIFI IP address.
<i>21~36</i>	PI-1X WIFI subnet mask.
<i>37~52</i>	PI-1X WIFI gateway.
<i>53~88</i>	WIFI accesspoint SSID name.
<i>89~92</i>	PI-1X WIFI TX power.
<i>93~96</i>	PI-1X WIFI power saving mode.
<i>97~100</i>	WIFI security key type.
<i>101~116</i>	TCP connect IP address.
<i>117~120</i>	TCP connect port.
<i>121~184</i>	WIFI security key.
<i>185~196</i>	WIFI module MAC address.(Read Only)
<i>197~202</i>	WIFI module F/W version. (Read Only)

*N1%* is an integer variable indicating LinkingPort's port number.  
The value is form 1 to 4.

## **LINKPORT WRITE**

Purpose : Write characters to LinkingPort.

Syntax : *N1%=LINKPORT\_WRITE(N2%,A\$,N3%)*

Example : A%=LINKPORT\_OPEN

```

...
WHILE 1
    A$=INKEY$
    IF A$<>"" THEN
        IF(ASC(A$)=27) THEN
            EXIT
        ELSE
            AA%=LINKPORT_WRITE(1,A$,1)
            IF AA%>0 THEN PRINT A$;
        END IF
    END IF
END IF

```

```

STR1$=LINKPORT_READ$(1,1)
IF LEN(STR1$)<>0 THEN
    PRINT STR1$;
END IF
WEND
...
A%=LINKPORT_CLOSE(1)
...

```

Description : After opening LinkingPort, you can write characters to that LinkingPort.

*N1%* is an integer variable.It will tell you how many characters send to that LinkingPort device and other mean as bellow:

<i>N1%</i>	<i>Meaning</i>
>=0	How many characters send to that LinkingPort.
-4	Parameter error.
-5	Connect fail.
-6	LinkingPort is not open.

*N2%* is an integer variable indicating LinkingPort's port number.  
The value is form 1 to 4.

*A\$* is a string variable indicating the characters is to be sent.

*N3%* is an integer variable indicating number of bytes to be written to that LinkingPort device.

## **LINKPORT\_READ\$**

Purpose : Read characters from LinkingPort.

Syntax : *A\$=LINKPORT\_READ\$(N1%,N2%)*

Example : STR1\$=LINKPORT\_READ\$(1)

```

A%=LINKPORT_OPEN
...
WHILE 1
    A$=INKEY$
    IF A$<>"" THEN
        IF(ASC(A$)=27) THEN
            EXIT
        ELSE
            AA%=LINKPORT_WRITE(1,A$,1)
            IF AA%>0 THEN PRINT A$;

```

---

```

        END IF
    END IF
END IF
STR1$=LINKPORT_READ$(1,1)
IF LEN(STR1$)<>0 THEN
    PRINT STR1$;
END IF
WEND
...
A%=LINKPORT_CLOSE(1)
...

```

Description : After opening LinkingPort, you can read characters from that LinkingPort.  
**A\$** is a string variable to be assigned the characters is read from that LinkingPort.  
**N1%** is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.  
**N2%** is an integer variable indicating number of bytes to be read from LinkingPort.

## **LINKPORT FLUSH**

Purpose : Flush the LinkingPort data buffer.

Syntax : **N1%=LINKPORT\_FLUSH(N2%)**

Example : **Result%=LINKPORT\_FLUSH(Port%)**

Description : After opening LinkingPort, you can flush the LinkingPort's read and write data buffer.

N1% is an integer variable to be assigned to the result.

N1%	Meaning
-1	Close LinkingPort success.
-4	Parameter error.
-6	LinkingPort is not open.
-7	LinkingPort is not set.

N2% is an integer variable indicating LinkingPort's port number. The value is form 1 to 4.

---

## 3.21 Simulator (Only for PC simulator) commands

### COPYFILETOPDT

Purpose : To copy a file from PC side to PDT.

Syntax : ***COPYFILETOPDT(PCPath\$, PDTPath\$)***

Example : COPYFILETOPDT("D:\Code\BASIC\5.BMP", "D:\PROGRAM\5.BMP")

Description : The COPYFILETOPDT command copies the PC file path specified by ***PCPath\$*** to the simulator path specified by ***PDTPath\$***.

### BACKUPDATAFILETOPC

Purpose : To backup a file from PDT to PC.

Syntax : ***BACKUPDATAFILETOPC(PDTPath\$, PCPath\$)***

Example : BACKUPDATAFILETOPC("D:\PROGRAM\5.BMP", "d:\test.bmp")

Description : The BACKUPDATAFILETOPC command copies the simulator datafile path specified by ***PDTPath\$*** to the ***PCPath\$*** in PC and stored in PC with the same file name.

---

## 4 Appendices

### Appendix A

#### PT-Basic Commands list

##### A1. General commands

Command	description
<u><a href="#">ABS</a></u>	To return the absolute value of a numeric expression.
<u><a href="#">DIM</a></u>	To specify the maximum value of variable subscripts and to allocate storage accordingly.
<u><a href="#">GOSUB</a></u>	To call a specified subroutine.
<u><a href="#">GOTO</a></u>	To branch unconditionally to a specified line number or line label from the normal program sequence.
<u><a href="#">INT</a></u>	To return the largest integer that is less than or equal to the given numeric expression.
<u><a href="#">REM</a></u>	To insert explanatory remarks in a program.
<u><a href="#">SET PRECISION</a></u>	To set the precision of the decimal points for printing real number expression.
<u><a href="#">SGN</a></u>	To return an indication of the mathematical sign (+ or -) of a given numeric expression.

##### A2. Commands for decision structures

Command	description
<u><a href="#">IF ... THEN ... {ELSE IF...}</a></u> <u><a href="#">[ELSE...] END IF</a></u>	To provide a decision structure for multiple-line conditional execution.
<u><a href="#">ON ... GOSUB ...</a></u>	To call one of the several specified subroutines depending on the value of the expression.
<u><a href="#">ON ... GOTO ...</a></u>	To branch to one of several specified Line Labels depending on the value of an expression.

---

### A3. Commands for looping structures

Command	description
<u><a href="#">EXIT</a></u>	To provide an alternative exit for looping structures, such as FOR...NEXT and WHILE...WEND statements.
<u><a href="#">FOR ... NEXT</a></u>	To repeat the execution of a block of statements for a specified number of times.
<u><a href="#">WHILE ... WEND</a></u>	To repeat the execution of a block of statements while a certain condition is TRUE.

---

#### A4. Commands for string processing

Command	description
<a href="#"><u>LEN</u></a>	To return the length of a string.
<a href="#"><u>INSTR</u></a>	To search if one string exists inside another one.
<a href="#"><u>LEFT\$</u></a>	To retrieve a given number of characters from the left side of the target string.
<a href="#"><u>MID\$</u></a>	To retrieve a given number of characters from anywhere of the target string.
<a href="#"><u>RIGHT\$</u></a>	To retrieve a given number of characters from the right side of the target string.
<a href="#"><u>TRIM LEFT\$</u></a>	To return a copy of a string with leading blank spaces stripped.
<a href="#"><u>TRIM RIGHT\$</u></a>	To return a copy of a string with trailing blank spaces stripped.
<a href="#"><u>ASC</u></a>	To return the decimal value for the ASCII code for the first character of a given string.
<a href="#"><u>CHR\$</u></a>	To return the character for a given ASCII value.
<a href="#"><u>HEX\$</u></a>	To return a string that represents the hexadecimal value (base 16) of the decimal argument.
<a href="#"><u>OCT\$</u></a>	To return a string that represents the octal value (base 8) of the decimal argument.
<a href="#"><u>LCASE\$</u></a>	To return a copy of a string in which all uppercase letters will be converted to lowercase letters.
<a href="#"><u>UCASE\$</u></a>	To return a copy of a string in which all lowercase letters will be converted to uppercase letters.
<a href="#"><u>STR\$</u></a>	To convert a numeric expression to a string.
<a href="#"><u>VAL</u></a>	To return the numeric value of a string expression in integer form.
<a href="#"><u>VALR</u></a>	To convert a string expression to a real number.
<a href="#"><u>STRING\$</u></a>	To return a string containing the specified number of the requested character.

---

## A5. Commands for event trapping

Command	description
<a href="#"><u>OFF ALL</u></a>	To terminate all the event triggers.
<a href="#"><u>OFF ESC</u></a>	To terminate ESC event trigger.
<a href="#"><u>OFF COM</u></a>	To terminate COM event trigger.
<a href="#"><u>OFF HOUR</u></a>	To terminate HOUR event trigger.
<a href="#"><u>OFF KEY</u></a>	To terminate KEY event trigger.
<a href="#"><u>OFF MINUTE</u></a>	To terminate MINUTE event trigger.
<a href="#"><u>OFF READER</u></a>	To terminate READER event trigger.
<a href="#"><u>OFF TIMER</u></a>	To terminate TIMER event trigger.
<a href="#"><u>ON COM GOSUB</u></a>	To activate COM event trigger.
<a href="#"><u>ON ESC GOSUB</u></a>	To activate ESC event trigger.
<a href="#"><u>ON HOUR GOSUB</u></a>	To activate HOUR event trigger.
<a href="#"><u>ON KEY GOSUB</u></a>	To activate KEY event trigger.
<a href="#"><u>ON MINUTE GOSUB</u></a>	To activate MINUTE event trigger.
<a href="#"><u>ON READER GOSUB</u></a>	To activate READER event trigger.
<a href="#"><u>ON TIMER GOSUB</u></a>	To activate TIMER event trigger.
<a href="#"><u>LOCK</u></a>	To hold all the activated event triggers until they are released by UNLOCK.
<a href="#"><u>UNLOCK</u></a>	To release all the activated event triggers held by LOCK.

## A6. System commands

Command	description
<a href="#"><u>AUTO OFF</u></a>	To set auto power off timer.
<a href="#"><u>DEVICE ID\$</u></a>	To get the serial number of the terminal.
<a href="#"><u>GET TARGET MACHINES\$</u></a>	To get the model name of the target terminal.
<a href="#"><u>MENU</u></a>	To create a menu.
<a href="#"><u>POWER ON</u></a>	To determine whether to restart or resume the program upon powering on.
<a href="#"><u>RESTART</u></a>	To restart the system.
<a href="#"><u>SYSTEM INFORMATION\$</u></a>	To get information on components.
<a href="#"><u>SYS SUSPEND</u></a>	To shut down the system.
<a href="#"><u>CHECK AID</u></a>	To verify if the agency ID is correct or not.
<a href="#"><u>COPYAPPTOBIO\$</u></a>	To copy setting from APP to BIOS.
<a href="#"><u>SET DCIN ALWAYS ON</u></a>	To set the state of DC in always power on.
<a href="#"><u>GET DCIN ALWAYS ON</u></a>	To get the state of DC in always power on.



---

## A7. Reader commands

Command	description
<u><a href="#">DISABLE READER</a></u>	To disable the reader ports of the terminal.
<u><a href="#">ENABLE READER</a></u>	To enable the reader ports of the terminal.
<u><a href="#">SLEEP READER</a></u>	To set scanner module to sleep.
<u><a href="#">GET READER DATA\$</a></u>	To get data that is read from a specified reader port.
<u><a href="#">GET READER DATALEN</a></u>	To get data length that is read from a specified reader port.
<u><a href="#">GET READER TYPE</a></u>	To get scanner type.
<u><a href="#">READER CONFIG START</a></u>	To start scanner setting procedure.
<u><a href="#">READER CONFIG END</a></u>	To end scanner setting procedure.
<u><a href="#">READER SENDCMD</a></u>	To send scanner (CCD) command to change scanner status.
<u><a href="#">READER QUERY\$</a></u>	To query the scanner (CCD) current setting.
<u><a href="#">DECODE</a></u>	To perform barcode decoding.
<u><a href="#">SIM SCANKEY PRESS</a></u>	To simulator the “Scan” key press or release.
<u><a href="#">READER SETFROMFILE</a></u>	To set scanner setting by scanner setting file.

## A8. Buzzer commands

Command	description
<u><a href="#">BEEP</a></u>	To assign a beeper sequence to beeper operation.
<u><a href="#">STOPBEEP</a></u>	To terminate beeper sequence.
<u><a href="#">SET BUZZER VOL</a></u>	To set the buzzer volume.

## A9. Calendar and timer commands

Command	description
<u><a href="#">DATE\$</a></u>	To set or to get the current date.
<u><a href="#">DAY OF WEEK</a></u>	To get the day of the week.
<u><a href="#">TIME\$</a></u>	To set or to get the current time.
<u><a href="#">TIMER</a></u>	To return the number of seconds elapsed since the terminal been powered on.
<u><a href="#">WAIT</a></u>	To set system delay time.

---

## A10. LED command

Command	description
<a href="#"><u>LED</u></a>	To set the LED indicators.

## A11. Keypad commands

Command	description
<a href="#"><u>CLR_KBD</u></a>	To clear the keypad buffer.
<a href="#"><u>INKEY\$</u></a>	To read one character from the keypad buffer and then remove it.
<a href="#"><u>INPUT_LEN</u></a>	To set or get input length when used "INPUT" or INPUT_S" command.
<a href="#"><u>INPUT</u></a>	To take user input from the keypad and store it in a variable.
<a href="#"><u>INPUT_S</u></a>	To take user input from the keypad, scanning and store it in a variable.
<a href="#"><u>INPUT_S_CARRYENT</u></a>	To set ENT auto press on/off when use "INPUT_S_CARRYENT" command.
<a href="#"><u>INPUT_S_VIBRATE</u></a>	To set vibrator on or off when use "INPUT_S_VIBRATE" command.
<a href="#"><u>INPUT_S_SLEEP</u></a>	To set scanner sleep on or off when use "INPUT_S_SLEEP" command.
<a href="#"><u>INPUT_MODE</u></a>	To set the display mode of the input data.
<a href="#"><u>KEY_CLICK</u></a>	To enable or disable the key click sound.
<a href="#"><u>ALPHA_LOCK</u></a>	To set the ALPHA state for input mode.
<a href="#"><u>GET_ALPHA_LOCK</u></a>	To get information of the ALPHA state for input mode.
<a href="#"><u>GET_KEY_CLICK</u></a>	To get current key click status.
<a href="#"><u>KEYPAD_BL_TIMER</u></a>	To set or get keypad backlight timer.
<a href="#"><u>KEYPAD_BL</u></a>	To set keypad backlight on/off.
<a href="#"><u>DEF_PKEY</u></a>	To change the definition of programmable key (P1 ~ P3) .

---

## A12. LCD Commands

Command	description
<u><a href="#">BACK_LIGHT_DURATION</a></u>	To specify how long the backlight will last once the terminal been turned on.
<u><a href="#">LCD_CONTRAST</a></u>	To set the contrast level of the LCD.
<u><a href="#">CURSOR</a></u>	To turn on/off the cursor indication in the activated TextBlock.
<u><a href="#">CURSOR_X</a></u>	To get the x coordinate of the current cursor position in the activated TextBlock.
<u><a href="#">CURSOR_Y</a></u>	To get the y coordinate of the current cursor position in the activated TextBlock.
<u><a href="#">LOCATE</a></u>	To move the cursor to a specified location in the activated TextBlock.
<u><a href="#">FILL_RECT</a></u>	To fill a user defined color rectangular area in the activated TextBlock.
<u><a href="#">ICON_ZONE_PRINT</a></u>	To enable or disable the statusbar.
<u><a href="#">PRINT</a></u>	To display data in the activated TextBlock.
<u><a href="#">CLR_RECT</a></u>	To clear a rectangular area in the activated TextBlock. The cursor position is not affected after the operation.
<u><a href="#">CLS</a></u>	To clear the activated TextBlock.
<u><a href="#">SHOW_IMAGE</a></u>	To put a rectangular bitmap in the activated TextBlock.
<u><a href="#">CLR_EOL</a></u>	To clear from where the cursor is to the end of the line. The cursor position is not affected after the operation.

## A13. User font commands

Command	description
<u><a href="#">DISPFONT_SETFONT</a></u>	To set user font from font file.
<u><a href="#">DISPFONT_INFO_TYPE</a></u>	To get font type.
<u><a href="#">DISPFONT_INFO_HEIGHT</a></u>	To get font height.
<u><a href="#">DISPFONT_INFO_WIDTH</a></u>	To get font width.

---

#### A14. TextBlock commands

Command	description
<a href="#"><u>DEFINETEXTBLOCK COLOR</u></a>	To define the TextBlock setting and the background using color or default background color.
<a href="#"><u>DEFINETEXTBLOCK IMAGE</u></a>	To define the TextBlock setting and the background using bitmap file or default background color.
<a href="#"><u>SETTEXTBLOCK</u></a>	To enable the specific TextBlock.
<a href="#"><u>RESETTEXTBLOCK</u></a>	To disable the specific TextBlock.
<a href="#"><u>PRINTTEXTBLOCK</u></a>	To print Text to specific TextBlock.
<a href="#"><u>GETTEXTBLOCKCUR X</u></a>	To get the x coordinate of the current TextBlock position.
<a href="#"><u>GETTEXTBLOCKCUR Y</u></a>	To get the y coordinate of the current TextBlock position.
<a href="#"><u>SETTEXTBLOCKCUR</u></a>	To set specific TextBlock as active TextBlock and set position.
<a href="#"><u>SHOWTEXTBLOCKCURSOR</u></a>	To show or hide TextBlock cursor.
<a href="#"><u>SWITCHTEXTBLOCK</u></a>	To switch TextBlock.

---

## A15. File manipulation commands

Command	description
<a href="#"><u>OPENIN</u></a>	To open (r) a file and get the header of the file for further processing.
<a href="#"><u>OPENOUT</u></a>	To open (w+) a file and get the header of the file for further processing.
<a href="#"><u>OPENUP</u></a>	To open (r+) a file and get the header of the file for further processing.
<a href="#"><u>MKDIR</u></a>	To create a folder.
<a href="#"><u>RMDIR</u></a>	To delete a folder.
<a href="#"><u>CLOSE</u></a>	To close a file.
<a href="#"><u>BGET</u></a>	To read a byte from a file. The current position is updated after reading.
<a href="#"><u>BGETEXT</u></a>	To read a specified number of bytes from a file. The current position is updated after reading.
<a href="#"><u>GET\$</u></a>	Read a line terminated by a null character “\0” from a file.
<a href="#"><u>BPUT</u></a>	To write data to a file.
<a href="#"><u>EOF</u></a>	To check if file pointer of a file reaches end of file.
<a href="#"><u>PTR</u></a>	To get or move the file pointer position of a file.
<a href="#"><u>EXT</u></a>	To get or change file length of a file.
<a href="#"><u>GET FILE ERROR</u></a>	To get the file error code.
<a href="#"><u>DBMS INIT SEARCH</u></a>	To initiate the file search in disk.
<a href="#"><u>DBMS INIT SEARCHADV</u></a>	To initiate the advance file search in disk.
<a href="#"><u>DBMS CLOSE SEARCH</u></a>	To close the file search in disk.
<a href="#"><u>DBMS APPEND DATA</u></a>	To increase one record on the file end.
<a href="#"><u>DBMS DEL DATA</u></a>	To delete the appointed record in the file.
<a href="#"><u>DBMS EMPTY</u></a>	To remove all the record in the file.
<a href="#"><u>DBMS FIND RECORD</u></a>	To search the designated field. This command only supports forward search.
<a href="#"><u>DBMS FIND RECORD B</u></a>	To search the designated field. This command only supports backward search.
<a href="#"><u>DBMS SEARCH FIELD</u></a>	To search the designated field.
<a href="#"><u>DBMS GET COUNT</u></a>	To obtain the figure of all records in the file.
<a href="#"><u>DBMS GET DATA\$</u></a>	To read the data of appointed field in the appointed record.
<a href="#"><u>DBMS UPDATE DATA</u></a>	To revise the data of appointed field in appointed

---

field record.

#### A16. Vibrator commands

Command	description
<a href="#"><u>VIBRATOR_TIMER</u></a>	To set or get the vibrator timer.
<a href="#"><u>VIBRATOR</u></a>	To set the vibrator on/off.

#### A17. Communication port commands

Command	description
<a href="#"><u>CLOSE_COM</u></a>	To terminate communication and disable a specified COM port.
<a href="#"><u>OPEN_COM</u></a>	To enable a specified COM port and initialize communication.
<a href="#"><u>SET_COM</u></a>	To set parameters of a specified COM port.
<a href="#"><u>READ_COM\$</u></a>	To read data from a specified COM port.
<a href="#"><u>WRITE_COM</u></a>	To send a string to the host through a specified COM port.
<a href="#"><u>GET_CTS</u></a>	To get CTS level.
<a href="#"><u>SET_RTS</u></a>	To set RTS level.
<a href="#"><u>CLEAR_COM</u></a>	To clear receiver buffer.
<a href="#"><u>COM_DELIMITER</u></a>	To change delimiter of sending and receiving string of a specified COM port.
<a href="#"><u>FILE_TRANS</u></a>	Using FILE_TRANS to upload or download files.
<a href="#"><u>FILE_TRANS_REALTIME</u></a>	Using FILE_TRANS_REALTIME to upload or download files immediately.
<a href="#"><u>FILE_TRANS_BAUD</u></a>	To get or set the FILE_TRANS baud rate.
<a href="#"><u>FILE_TRANS_INTERFACE</u></a>	To get or set the FILE_TRANS interface.
<a href="#"><u>FILE_TRANS_GETBT</u></a>	To get the FILE_TRANS BT information.
<a href="#"><u>FILE_TRANS_GETWIFI</u></a>	To get the FILE_TRANS WIFI information.
<a href="#"><u>FILE_TRANS_SETBT</u></a>	To set the FILE_TRANS BT information.
<a href="#"><u>FILE_TRANS_SETWIFI</u></a>	To set the FILE_TRANS WIFI information.
<a href="#"><u>FILE_TRANS_SENDMSG</u></a>	Send the message to PC.
<a href="#"><u>GET_FILETRANS_ERROR</u></a>	To get the FILE_TRANS error code.

#### A18. Memory commands

Command	description
<a href="#"><u>RAM_SIZE</u></a>	To check the total space in disk C.
<a href="#"><u>ROM_SIZE</u></a>	To check the total space in disk D.q

---

<a href="#"><u>SD_SIZE</u></a>	To check the total space in disk E.
<a href="#"><u>FREE_MEMORY</u></a>	To check the free space in disk C/ D/ E.
<a href="#"><u>DISK_USED\$SIZE</u></a>	To check the occupied space in disk C/ D/ E.

#### A19. USB commands

Command	description
<a href="#"><u>USB_OPEN</u></a>	To initialize and enable USB port.
<a href="#"><u>USB_CLOSE</u></a>	To close the USB port.
<a href="#"><u>USB_READ\$</u></a>	To read specific number of bytes from USB port.
<a href="#"><u>USB_WRITE</u></a>	To write specific number of bytes to the PC side.

#### A20. LinkingPort commands

Command	description
<a href="#"><u>LINKPORT_OPEN</u></a>	Start a linkingPort.
<a href="#"><u>LINKPORT_CLOSE</u></a>	Stop a linkingport.
<a href="#"><u>LINKPORT_SELECTIF</u></a>	Set LinkingPort interface select setting.
<a href="#"><u>LINKPORT_GETIF</u></a>	Get LinkingPort interface select setting.
<a href="#"><u>LINKPORT_SETCOM</u></a>	Set LinkingPort COM baudrate setting.
<a href="#"><u>LINKPORT_GETCOM</u></a>	Get LinkingPort COM baudrate setting.
<a href="#"><u>LINKPORT_SETBT</u></a>	Set LinkingPort Bluetooth function setting.
<a href="#"><u>LINKPORT_GETBT</u></a>	Get LinkingPort Bluetooth function setting.
<a href="#"><u>LINKPORT_SETWIFI</u></a>	Set LinkingPort WIFI function setting.
<a href="#"><u>LINKPORT_GETWIFI</u></a>	Get LinkingPort WIFI function setting.
<a href="#"><u>LINKPORT_WRITE</u></a>	Write characters a linkingport.
<a href="#"><u>LINKPORT_READ\$</u></a>	Read characters from a linkingport.
<a href="#"><u>LINKPORT_FLUSE</u></a>	Flush the LinkingPort data buffer.

#### A21. Simulator (Only for PC simulator) commands

Command	description
<a href="#"><u>COPYFILETOPDT</u></a>	To copy a file from PC side to PDT.
<a href="#"><u>BACKUPDATAFILETOPC</u></a>	To backup a file from PDT to PC.

---

## Appendix B

### Scan Module (CCD) Configuration Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	1 Preamble transmission	0: Disable * 1: Enable
	2 Postamble transmission	0: Disable * 1: Enable
	7 Code ID position	0: Before code data * 1: After code data
	8 Code ID transmission	0: Disable * 1: Proprietary ID 2: AIM ID
	9 Code length transmission	0: Disable * 1: Enable
	10 Code name transmission	0: Disable * 1: Enable
	11 Case conversion	0: Disable * 1: Upper case 2: Lower case
7 Scan	4 Double confirm	0 ~ 9 0 *
	6 Global min. code length	0 ~ 99 4 *
	7 Global max. code length	0 ~ 99 63 *
	8 Inverted image scan	0: Disable * 1: Enable
8 String setting	1 Prefix characters setting	0 * 0x00 ~ 0xff ASCII code 12 characters.



	2 Suffix characters setting	0 * 0x00 ~ 0xff ASCII code 12 characters.
	3 Preamble characters settings	0 * 0x00 ~ 0xff ASCII code 12 characters.
	4 Postamble characters settings	0 * 0x00 ~ 0xff ASCII code 12 characters.
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/One digit * 2:Disable/Two digits 3:Enable/One digit 4:Enable/Two digits
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<O> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 1 *
	6 Truncate leading	0 ~ 20 0 *

	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<*> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: Full ASCII
	13 Start/stop transmission	0: Disable * 1: Enable
12 Code 93	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<&> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
13 Code 128	1 Read	0: Disable 1: Enable *
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 1 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *

	8 Code ID setting	<#> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: UCC.EAN 128
	12 UCC/EAN 128 ID setting	<#> 0x00 ~ 0xff ASCII code(1 bytes)
	13 Concatenation code	0x1D * 0x00 ~ 0xff ASCII code(1 bytes)
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<%> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Start/stop type	0: ABCD/ABCD * 1: abcd/abcd 2: ABCD/TN*E 3: abcd/tn*e
	11 Start/stop transmission	0: Disable * 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable *
	2 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7	0 ~ 15

	Truncate ending	0 *
	8 Code ID setting	<FF> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncation/expansion	0: None * 1: Truncate leading zero 2: Expand to EAN 13
	12 Expansion	0: Disable * 1: Enable
16 EAN 13	1 Read	0: Disable 1: Enable *
	2 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<F> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	12 ISBN/ISSN conversion	0: Disable * 1: Enable

17 Industrial 2 of 5	1 Read	0:Disable * 1:Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
19 Standard 2 of 5	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *

	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:N/disable * 1:N/MOD 10 2:N/Mod 10,10 3:N/mod 11,10 4:Y/ Mod10 5:Y/ Mod 10,10 6:Y/ Mod 11/10
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<@> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *

	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<@> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
22 Telepen	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<S> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Numeric * 1: Full ASCII
23 UPCA	1 Read	0: Disable 1: Enable *
	2 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<A> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2. 5 digitis 3: 2, 5 digits

		4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None 1: Truncate leading zero * 2: Expand to EAN 13
24 UPCE	1 Read	0: Disable 1: Enable *
	2 Check-sum transmission	0: Disable 1: Enable *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<E> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None * 1: Truncate leading zero 2: Expand to EAN 13 3: Expand to UPCA
	12 Expansion	0: Disable * 1: Enable
	13 UPCE-1	0: Disable * 1: Enable
25 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable
	4	0 ~ 64



	Max. code length	0 *
	5	0 ~ 64
	Min. code length	0 *
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
28 China post	Truncate ending	0 *
	8	<B>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
	1	0: Disable *
	Read	1: Enable
	4	0 ~ 64
29 RSS 14	Max. code length	11 *
	5	0 ~ 64
	Min. code length	11 *
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
30 RSS Limited	Truncate ending	0 *
	8	<t>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
	1	0: Disable *
	Read	1: Enable
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
	Truncate ending	0 *
	8	<R4>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11	0: Disable *
	UCC/EAN 128 emulation	1: Enable
	1	0: Disable *
30 RSS Limited	Read	1: Enable
	6	0 ~ 15
	Truncate leading	0 *

	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<RL> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable
31 RSS Expanded	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 99 99 *
	5 Min. code length	0 ~ 99 1 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<RX> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable
32 Italian Pharmacode 39	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 12 *
	5 Min. code length	0 ~ 64 9 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<p> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Leading "A"	0: Disable * 1: Enable

---

Parameter1	Parameter2	Alphanumeric Entry
8 String setting	2 Suffix characters setting	0 * 0x00 ~ 0xff ASCII code 22 characters.
	3 Preamble characters settings	0 * 0x00 ~ 0xff ASCII code 22 characters.
	4 Postamble characters settings	0 * 0x00 ~ 0xff ASCII code 22 characters.
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable 1: One digit * 2: Two digits
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<O> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

---

Parameter1	Parameter2	Alphanumeric Entry
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 20 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<*> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: Full ASCII
	13 Start/stop transmission	0: Disable * 1: Enable

---

Parameter1	Parameter2	Alphanumeric Entry
12 Code 93	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<&> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

Parameter1	Parameter2	Alphanumeric Entry
13 Code 128	1 Read	0: Disable 1: Enable *
	2 Check-sum verification	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<#> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: UCC.EAN 128
	12 UCC/EAN 128 ID setting	<#> 0x00 ~ 0xff ASCII code(1 bytes)
	13 Concatenation code	0x1D * 0x00 ~ 0xff ASCII code(1 bytes)

---

Parameter1	Parameter2	Alphanumeric Entry
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<%> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Start/stop type	0: ABCD/ABCD * 1: abcd/abcd 2: ABCD/TN*E 3: abcd/tn*e
	11 Start/stop transmission	0: Disable * 1: Enable

---

Parameter1	Parameter2	Alphanumeric Entry
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<FF> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncation/expansion	0: None * 1: Truncate leading zero 2: Expand to EAN 13
	12 Expansion	0: Disable * 1: Enable



Parameter1	Parameter2	Alphanumeric Entry
16 EAN 13	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<F> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	12 ISBN/ISSN conversion	0: Disable * 1: Enable
17 Industrial 2 of 5	1 Read	0:Disable * 1:Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

Parameter1	Parameter2	Alphanumeric Entry
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
19 Standard 2 of 5	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

Parameter1	Parameter2	Alphanumeric Entry
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable 1: Mod 10 * 2: Mod 10/10 3: Mod 11/10
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<@> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8	<@>

---

	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
--	-----------------	--------------------------------------

Parameter1	Parameter2	Alphanumeric Entry
22 Telepen	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<S> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Numeric * 1: Full ASCII

---

Parameter1	Parameter2	Alphanumeric Entry
23 UPCA	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<A> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2. 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None 1: Truncate leading zero * 2: Expand to EAN 13

Parameter1	Parameter2	Alphanumeric Entry
24 UPCE	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<E> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None * 1: Truncate leading zero 2: Expand to EAN 13 3: Expand to UPCA
	12 Expansion	0: Disable * 1: Enable
	13 UPCE-1	0: Disable * 1: Enable

Parameter1	Parameter2	Alphanumeric Entry
25 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<B> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
28 China post	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 11 *
	5 Min. code length	0 ~ 64 11 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<t> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

---

Parameter1	Parameter2	Alphanumeric Entry
29 RSS 14	1 Read	0: Disable * 1: Enable
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<R4> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable
30 RSS Limited	1 Read	0: Disable * 1: Enable
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<RL> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable



Parameter1	Parameter2	Alphanumeric Entry
31 RSS Expanded	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 99 99 *
	5 Min. code length	0 ~ 99 1 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<RX> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable
32 Italian Pharmacode	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 12 *
	5 Min. code length	0 ~ 64 9 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<p> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Leading "A"	0: Disable * 1: Enable