



PA-20/PA-2010

PA-21/PA-2110 Basic

Programming Manual

Version:1.01

Copyright © 2012 by ARGOX Information Co., Ltd.

<http://www.argox.com>

Preface

To satisfy the user's customization needs, PA-20 / PA-2010 / PA-21 / PA-2110 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.

PA-20 / PA-2010 / PA-21 / PA-2110 Basic interpreter provides a platform for users to develop application programs to be executed on the PA2 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 PA-20 / PA-2010 / PA-21 / PA-2110 Basic and PA2 series and the productivity they can boost for you in your application.

Table of Contents

Preface.....	1
Table of Contents	2
1 How to run BASIC program.....	5
1.1 BASIC Menu	5
1.1.1 Run program	5
1.1.2 Communication.....	5
1.1.3 Information	7
2 Program Structure	8
2.1 Constants.....	8
2.1.1 String.....	8
2.1.2 Numeric.....	8
2.2 Variables.....	8
2.2.1 Variable Names and Declaration Characters.....	9
2.2.2 Array Variables	9
2.3 Expression and Operators	9
2.3.1 Assignment Operator	10
2.3.2 Arithmetic Operator	10
2.3.3 Relational Operator	10
2.3.4 Logical Operator	10
2.4 Operator Precedence	11
2.5 Labels.....	11
2.6 Subroutines	12
2.7 Exit program	13
2.8 Special notes	13
3 Command Sets.....	14
3.1 General commands.....	14
3.2 Commands for decision structures.....	18
3.3 Commands for looping structures.....	21
3.4 Commands for string processing	23
3.5 Commands for event trapping.....	29
3.6 System commands	39
3.7 Reader commands.....	43
3.8 Beeper commands.....	51
3.9 Calendar and timer commands.....	53

3.10	LED Command	55
3.11	Keypad commands	56
3.12	LCD Commands	65
3.13	Font	69
3.13.1	User font commands	69
3.14	TextBlock	71
3.14.1	TextBlock commands	72
3.15	File manipulation commands	77
3.15.1	Standard Commands	77
3.15.2	DBMS Commands	84
3.16	Vibrator commands	89
3.17	Communication port commands	90
3.18	Memory commands	96
3.19	Bluetooth commands (Only for PA-2010/2110)	97
3.20	USB commands	105
3.21	Simulator (Only for PC simulator) commands	106
4	Appendices.....	107
	Appendix A	107
	PT-Basic Commands list	107
A1.	General commands.....	107
A2.	Commands for decision structures.....	107
A3.	Commands for looping structures.....	108
A4.	Commands for string processing	109
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	114
A11.	Keypad commands.....	114
A12.	LCD Commands	115
A13.	User font commands	115
A15.	File manipulation commands.....	117
A16.	Vibrator commands.....	118
A17.	Communication port commands.....	118
A18.	Memory commands	118
A19.	Bluetooth commands	119
A20.	USB commands	119

A21. Simulator (Only for PC simulator) commands	119
Appendix B	120
Scan Module (CCD) Configuration Table	120
Appendix C	136
Scan Module (Laser) Configuration Table	136
Appendix D	146
Parameter for Color	146

1 How to run BASIC program

1.1 BASIC Menu



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

1.1.1 Run program

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

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



1.1.2 Communication

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.

A%	Meaning
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 elsestatements] 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\$="ABCDEFGHIIJK"

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\$="ABCGEFGHIIJK"

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\$ = RIGHT\$(Str\$, N%)**
- Example : Str\$ = "ABCDEFGHJK"
PRINT RIGHT\$(Str\$,3)
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\$ = TRIM_LEFT\$(Str\$)**
- Example : PRINT TRIM_LEFT\$(" 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\$ = TRIM_RIGHT\$(Str\$)**
- Example : 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% = ASC(Str\$)**
- Example : A%=ASC("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\$ = CHR\$(N%)**

Example : A\$=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\$ = HEX\$(N%)**

Example : A\$ = 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\$ = OCT\$(N%)**

Example : A\$ = 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\$ = LCASE\$(Str\$)**

Example : Str\$="ABCDEFGH"
PRINT LCASE\$(Str\$)
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 integer 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 1 to 6, 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(2)
- 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_MACHINES\$

- Purpose : To get the model name of the target terminal.
- Syntax : **A\$ = GET_TARGET_MACHINE\$**
- Example : PRINT "Model Name:" + GET_TARGET_MACHINES\$
- 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.

N%	Meaning
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>	Meaning
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.

A%	Meaning
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.

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.

N%	Meaning
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.

N2%	Meaning
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.

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.

A%	Meaning
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_SENDCMD_LASER

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

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

Example : READER_CONFIG_START

...

'Code-39 can read

A%=READER_SENDCMD_LASER(11,1, CHR\$(1))

'Preamble characters setting

A%=READER_SENDCMD_LASER (8,3, "abcde")

...

READER_CONFIG_END

Description : This command can change scanner status.

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

A%	Meaning
-----------	----------------

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 C](#)” 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.

READER_QUERY_LASERS

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

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

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

Value\$=READER_QUERY_LASER\$(13, 1)

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

Preamble\$=READER_QUERY_LASER\$(8, 3)

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 C](#)” 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\PA21.axs")

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

A%	Meaning
0	Setting fail
1	Setting OK

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

READER_GETTYPE

Purpose : To get scanner type for CCD or Laser.

Syntax : **A%=READER_GETTYPE**

Example : A%=READER_GETTYPE

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

A%	Meaning
0	CCD
1	Laser

READER_ALL_SYMBOLOGIES

Purpose : To set all scan codes on or off.

Syntax : **A%=READER_ALL_SYMBOLOGIES(N%)**

Example : ‘Set all scan codes on.
A%=READER_ALL_SYMBOLOGIES(1)

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

A%	Meaning
0	Setting fail
1	Setting OK

N% is an integer variable indicating the set of all scan codes on or off.

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

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.

N%	Buzzer volume
0	close
1	Low
2	Medium
3	High

3.9 Calendar and timer commands

DATES

- 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.

TIMES

- 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)

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 :

number%	description
1	LED displays green light.
2	LED displays red light.
3	LED displays orange light.

mode%	description
1	off for (duration% X 0.01) seconds then on
2	on for (duration% X 0.01) seconds then off
3	flash, on then off each for (duration% 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 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.

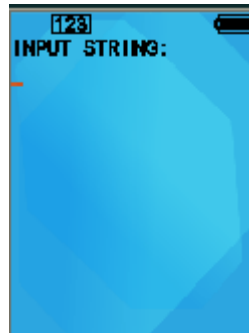
INPUT

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.

A%	Meaning
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.

<i>A%</i>	<i>Meaning</i>
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.

N%	Meaning
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.

mode%	Meaning
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.

status%	Key click sound
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.

<i>status%</i>	<i>Alpha status</i>	<i>Default input</i>
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.

<i>A%</i>	<i>Key click sound</i>
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.

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

DEF_PKEY

Purpose : To change the definition of programmable key (P1 & P2) .

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(1,5)	'P1 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(2,11)	'P2 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(1,127)	'P1 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 :

N1%	Meaning
1	Define P1 key
2	Define P2 key

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

SCANKEYPWON

Purpose : To set power on by scan key.

Syntax : **A%=SCANKEYPWON(N%)**

Example : Result%=SCANKEYPWON(1)

Description :

N%	Meaning
0	Normal
1	Set power on by scan key

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

A%	Meaning
0	Setting fail
1	Setting success

GET_SCANKEYPWON

Purpose : Get state for power on by scan key.

Syntax : **A%=GET_SCANKEYPWON**

Example : Result%=GET_SCANKEYPWON

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

A%	Meaning
0	Normal
1	Set power on by scan key

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 5. 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 with a user defined color.
- Syntax : ***FILL_RECT(color% , left% , top% , width% , height%)***
- Example : FILL_RECT(65280,100,100,100,100) ‘green rectangular area
- Description : Several the argument as follows:

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

status%	Meaning
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:

left %	Fill from the start point of X-axis (pixel).
top %	Fill from the start point of Y-axis (pixel).
width%	Fill the width from the start point (pixel).
height%	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:

left %	Fill from the start point of X-axis (pixel).
top %	Fill from the start point of Y-axis (pixel).
width%	Fill the width from the start point (pixel).
height%	Fill the height from the start point (pixel).
path\$	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.

A%	Meaning
0	Set font fail
1	Set font OK

Several key arguments as below:

FontID%	Font ID (2~9)
FontPath\$	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.

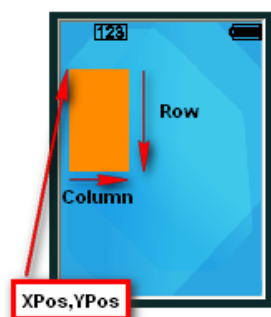
3.14.1 TextBlock commands

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.

A%	Meaning
0	Define TextBlock fail
1	Define TextBlock OK

Several key arguments as below:

BlockNo%	TextBlock number(1~15)
FontID%	Defined Font: 0~1: system font 2~9: user font.
BGType%	If 0 then using default background. If 1 then using user defined background.
Color%	Background color
Column%	TextBlock column number.
Row%	TextBlock row number.
XPos%	TextBlock left-top X position in pixel (0~239).

YPos%	TextBlock left-top Y position in pixel. StatusBar enable: 0~295. StatusBar disable: 0~319.
--------------	--

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:

BlockNo%	TextBlock number(1~15)
FontID%	Defined Font: 0~1: system font 2~9: user font.
BGType%	If 0 then using default background. If 1 then using bitmap file..
BitmapPath\$	Bitmap file path
Column%	TextBlock column number.
Row%	TextBlock row number.
XPos%	TextBlock left-top X position in pixel (0~239).
YPos%	TextBlock left-top Y position in pixel. StatusBar enable: 0~295. StatusBar disable: 0~319.

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:

BlockNo%	TextBlock number(1~15)
Save%	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:

BlockNo%	TextBlock number(0~15)
Column%	TextBlock column number.
Row%	TextBlock row number.
Str\$	Text data.
FontColor %	Text color.

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:

BlockNo%	TextBlock number(0~15)
Column%	TextBlock column number.
Row%	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:

BlockNo%	TextBlock number(0~15)
Show %	1:Show cursor 0:Hide cursor
Type %	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 ■.

TEXTBLOCK_SETBGCOLOR

- Purpose : To set default background color.
- Syntax : **TEXTBLOCK_SETBGCOLOR(Color%)**
- Example : TEXTBLOCK_SETBGCOLOR(16711680) 'Blue
- Description : **Color %** is an integer variable indicating which color you want to set.
After executing this command, all TextBlock will be reset.

TEXTBLOCK_SETBGIMAGE

Purpose : To set default background image for bitmap file.

Syntax : ***A%=TEXTBLOCK_SETBGIMAGE(FilePath\$)***

Example : R%=TEXTBLOCK_SETBGIMAGE("d:\program\test.bmp")

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

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

FilePath \$ is a string variable, indicating the bitmap file path.

After executing this command, all TextBlock will be reset.

TEXTBLOCK_MODE

Purpose : To set or get TextBlock mode for single or multi layer.

Syntax : ***TEXTBLOCK_MODE = A%***

X% = TEXTBLOCK_MODE

Example : TEXTBLOCK_MODE=1

...

PRINT "TEXTBLOCK_MODE : "; TEXTBLOCK_MODE

Description : ***A%*** is an integer variable to be assigned as the TextBlock mode..

X% is an integer variable indicating the TextBlock mode.

<i>A% or X%</i>	<i>Meaning</i>
0	single layer
1	multi layer

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.

<i>A%</i>	<i>Meaning</i>
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.

<i>F%</i>	<i>Meaning</i>
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:

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

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.

<i>F%</i>	<i>Meaning</i>
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:

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

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.

<i>F%</i>	<i>Meaning</i>
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:

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

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:

<i>GET_FILE_ERROR</i>	<i>Meaning</i>
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:

GET_FILE_ERROR	Meaning
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:

GET_FILE_ERROR	Meaning
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:

GET_FILE_ERROR	Meaning
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%=168
 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:

GET_FILE_ERROR	Meaning
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.

E%	Meaning
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:

GET_FILE_ERROR	Meaning
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:

GET_FILE_ERROR	Meaning
-----------------------	----------------

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:

GET_FILE_ERROR	Meaning
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\variable.DAT",1,"",1,0,3)

Result

%=DBMS_INIT_SEARCH("C:\DATA\fix.DAT",2,"6,8,8",0,22,3)

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>	<p>This argument has two kinds of meanings.</p> <ul style="list-style-type: none"> ■ When search for regular length, it needs to insert the unsigned char array; the array represents the length of every field. ■ When search for variable length, it needs to insert one character to represent separate symbol.
<i>N1%</i>	<ul style="list-style-type: none"> ■ When N1%=0, search for regular length. It has no separate symbols between different fields. ■ When N1%=1, search for variable length. It needs a separate symbol between different fields.
<i>N2%</i>	<p>This argument is each record's length.</p> <ul style="list-style-type: none"> ■ When N1%=0, needs to insert this value,

	not including the symbol of line feed. ■ When N1%=1, this field can insert any value.
N3%	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,"6,8,8","1,3",2,0,22,3)

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.

A%	Meaning
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:

<i>FilePath\$</i>	DBMS file path
<i>DBMSID%</i>	DBMS ID (1~10)
<i>S1\$</i>	This argument has two kinds of meanings. <ul style="list-style-type: none"> ■ When search for regular length, it needs to insert the unsigned char array; the array represents the length of every field. ■ When search for variable length, it needs to insert one character to represent separate symbol.
<i>S2\$</i>	This argument can give max. 8 key fields for search. We will make a checksum index file for these key fields.
<i>N1%</i>	This argument can give the sum of key fields size.
<i>N2%</i>	<ul style="list-style-type: none"> ■ When $N1\%=0$, search for regular length. It has no separate symbols between different fields. ■ When $N1\%=1$, search for variable length. It needs a separate symbol between different fields.
<i>N3%</i>	This argument is each record's length. <ul style="list-style-type: none"> ■ When $N1\%=0$, needs to insert this value, not including the symbol of line feed. ■ When $N1\%=1$, this field can insert any value.
<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.

A%	Meaning
0	Search defeat.
Other value	Match the record position of data

Several key arguments as below:

DBMSID%	DBMS ID (1~10)
field%	Search wanted field.
key \$	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.

A%	Meaning
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:

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

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.

<i>N%</i>	<i>Meaning</i>
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:

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

A%	Meaning
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.

N2%	Meaning
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 Argolink 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.

N%	Meaning
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.

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

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=0 '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.

<i>FILE_TRANS_INTERFACE</i>	<i>Interface</i>
0	RS-232
1	USB

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.

FREE_MEMORY

- Purpose : To check the free space in disk C or disk D.
- 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).

DISK_USEDSIZE

- Purpose : To check the occupied space in disk C or disk D.
- 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).

3.19 Bluetooth commands (Only for PA-2010/2110)

These commands only for PA-2010/2110, and our Bluetooth module only support SPP mode(Serial Port Profile).

BT_START

Purpose : Bluetooth module power enable.

Syntax : **BT_START**

Example : BT_START

...

...

BT_STOP

Description : This command can enable Bluetooth module power. After use this command, the left led will flash blue light.If you want to use other Bluetooth command, you must run this command first.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Bluetooth module power enable.
6	The terminal is not PA-2010/2110.
8	Bluetooth module has been power enable, please run.

BT_STOP

Purpose : Bluetooth module power disable.

Syntax : **BT_STOP**

Example : BT_START

...

...

BT_STOP

Description : This command can disable Bluetooth module power. The left led will stop flashing.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Bluetooth module power disable.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please

	run BT_SART.
--	--------------

BT_OPEN

Purpose : Bluetooth connect.

Syntax : **BT_OPEN**

Example : BT_OPEN

...

...

BT_CLOSE

Description : This command can connect to other Bluetooth device. Before use this command, you have to set the target Bluetooth MAC address by using “BT_SETLOCALSET” command.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Bluetooth connect ok.
5	Connect timeout.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_SART.
10	Bluetooth already connect.

BT_CLOSE

Purpose : Bluetooth disconnect.

Syntax : **BT_CLOSE**

Example : BT_OPEN

...

...

BT_CLOSE

Description : This command can disconnect Bluetooth. If you want to disconnect, you can use “BT_STOP” or this command.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Bluetooth disconnect ok.
5	Connect timeout.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_SART.

BT_WRITE

Purpose : Write characters to Bluetooth module.

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

Example : BT_START

```
...
BT_OPEN
IF GET_BT_ERROR=1 THEN
  CLS
  PRINT "    BlueTooth test"
  WHILE 1
    A$=INKEY$
    IF A$<>"" THEN
      IF(ASC(A$)=27) THEN EXIT

      N1%=0
      N1%=BT_WRITE(A$,1)
      IF N1%=1 THEN PRINT A$;

    END IF
    STR1$=BT_READ$(1)
    IF LEN(STR1$)<>0 THEN PRINT STR1$;
  WEND
  BT_CLOSE
ELSE
  PRINT "Link error!!!"
  WHILE INKEY$=""
  WEND
END IF
...
BT_STOP
...
```

Description : If Bluetooth is connected, this command can write characters to other Bluetooth device.

N1% is an integer variable. It will tell you how many characters send to other Bluetooth device.

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

N2% is an integer variable indicating number of bytes to be written to other Bluetooth device.

You can use the “GET_BT_ERROR” command to get the error

code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Write OK.
2	Parameter error, please check your parameter.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_SART.
9	Bluetooth not connect to other bluetooth device, please run BT_OPEN.

BT_READ\$

Purpose : Read characters from Bluetooth module.

Syntax : **A\$=BT_READ\$(N%)**

Example : STR1\$=BT_READ\$(1)

Description : If Bluetooth is connected, this command can read characters from Bluetooth module.

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

N% is an integer variable indicating number of bytes to be read from other Bluetooth device.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Read OK.
2	Parameter error, please check your parameter.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_SART.
9	Bluetooth not connect to other bluetooth device, please run BT_OPEN.

BT_GETLOCALINFO\$

Purpose : Get Bluetooth information.

Syntax : **A\$=BT_GETLOCALINFO\$**

Example : ...

S1\$=BT_GETLOCALINFO\$

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

```
PRINT "LocAdd: ";LocalAddress$
```

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

```
PRINT "LocName: ";LocalName$
```

```
LocalSec%=ASC(MID$(S1$,37,4))
```

```
PRINT "LocalSec: ";LocalSec%
```

```
LocalEnc%=ASC(MID$(S1$,41,4))
```

```
PRINT "LocalEnc: ";LocalEnc%
```

```
LocalTimeout%=ASC(MID$(S1$,45,4))
```

```
PRINT "LocalTimeout: ";LocalTimeout%
```

```
LocalRes%=ASC(MID$(S1$,49,4))
```

```
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 PA-2010/2110 Bluetooth information.Format of string as show below:

<i>A\$(Length)</i>	<i>Meaning</i>
<i>1~16</i>	PT-20B Bluetooth MAC address.(Cannot change.)
<i>17~36</i>	PT20B Bluetooth device name
<i>37~40</i>	PT-20B Bluetooth security mode, if 1(on) else 0(off)
<i>41~44</i>	PT-20B Bluetooth encryption mode, if 1(on) else 0(off)
<i>45~48</i>	PT-20B Bluetooth inquiry timeout, the value from 1(1.28 seconds) to 48(61.44 seconds).
<i>49~52</i>	PT-20B Bluetooth inquiry max response, the value from 1 to 10.
<i>53~68</i>	To linking device address.

69~88	PIN code.
--------------	-----------

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Read OK.
6	The terminal is not PA-2010/2110.

BT_SETLOCALSET

Purpose : Set Bluetooth information.

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

Example : BT_SETLOCALSET(LocalName\$,1,1,3,10,DeviceAddress\$,PIN\$)

Description : Several key arguments as below:

S1\$	PT20B Bluetooth device name(Allow 1~16 characters)
N1%	PT-20B Bluetooth security mode, set 1(on) or 0(off)
N2%	PT-20B Bluetooth encryption mode, set 1(on) or 0(off)
N3%	PT-20B Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).
N4%	PT-20B Bluetooth inquiry max response, the value from 1 to 10.
S2\$	Set link device address(Allow 1~12 characters)
S3\$	Set PIN code(Allow 4~16 characters)

You can use the GET_BT_ERROR command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Get information OK.
2	Parameter error, please check your parmeter.
5	Timeout.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_ SART.
10	Bluetooth already connect.

BT_INQUIRY

Purpose : Inquiry other Bluetooth module for PA-2010/2110 to connect.

Syntax : **N1%=BT_INQUIRY(N2%)**

Example : Num%=BT_INQUIRY(1)

Description : This command can search other Bluetooth device for PA-2010/2110.You can use “BT_GETDEVICEINFO” command to get other Bluetooth devices information.

N1% is an integer variable. It will return how many devices are found.

N2% is an integer variable. It will tell the command that will return device name or not.

You can use the “GET_BT_ERROR” command to get the error code. Possible error codes and their interpretation are listed below:

GET_BT_ERROR	Meaning
1	Search OK.
5	Search fail, please search again.
6	The terminal is not PA-2010/2110.
7	Bluetooth module power disable, please run BT_ SART.
10	Bluetooth already connect.

BT_GETDEVICEINFO

Purpose : To get other bluetooth device information after the search.

Syntax : **A\$=BT_GETDEVICEINFO(N%)**

Example : ...

S\$=BT_GETDEVICEINFO(1)

CLS

DEVICEAddress\$=LEFT\$(S\$,16)

PRINT "DEVICEAddress:";DEVICEAddress\$

DEVICENAME\$=MID\$(S\$,17,20)

PRINT "DEVICENAME:";DEVICENAME\$

...

Description : **A\$** is a string variable indicating the PA-2010/2110 Bluetooth information. Format of string as show below:

A\$(Length)	Meaning
1~16	Device MAC addresses.
17~36	Device name.

N% is an integer variable indicating which device information you want to get.

BT_DISCONNECTALERT

Purpose : For bluetooth disconnect alert.

Syntax : **BT_DISCONNECTALERT(N1%,N2%)**

Example : BT_DISCONNECTALERT(1,0)

‘Only beep one time for disconnect

Description : This command can prompt you that bluetooth disconnect after connect.

If **N1%** is 1, the buzzer will sound for disconnect after connect,
else, the buzzer will do nothing for disconnect after connect.
And, if **N1%** is 1, the **N2%** will be set for buzzer sound time gap.
For example, if **N2%** is 2, the buzzer will sound for each 2 seconds.

GET_BT_ERROR

Purpose : To get the bluetooth error code.
Syntax : **N%=GET_BT_ERROR**
Example : N%=GET_BT_ERROR
Description : **N%** is an integer to be assigned to the result.

3.20 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_READS

- 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.

3.21 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.

SIM_SETSCANNERTYPE

- Purpose : To setting scan module type.
- Syntax : **SIM_SETSCANNERTYPE(N%)**
- Example : SIM_SETSCANNERTYPE(1)
- Description : N% is an integer variable indicating the scan module type.

N%	Meaning
0	CCD module
1	Laser module

4 Appendices

Appendix A

PT-Basic Commands list

A1. General commands

Command	description
ABS	To return the absolute value of a numeric expression.
DIM	To specify the maximum value of variable subscripts and to allocate storage accordingly.
GOSUB	To call a specified subroutine.
GOTO	To branch unconditionally to a specified line number or line label from the normal program sequence.
INT	To return the largest integer that is less than or equal to the given numeric expression.
REM	To insert explanatory remarks in a program.
SET_PRECISION	To set the precision of the decimal points for printing real number expression.
SGN	To return an indication of the mathematical sign (+ or -) of a given numeric expression.

A2. Commands for decision structures

Command	description
IF ... THEN ... {ELSE IF...} [ELSE...] END IF	To provide a decision structure for multiple-line conditional execution.
ON ... GOSUB ...	To call one of the several specified subroutines depending on the value of the expression.
ON ... GOTO ...	To branch to one of several specified Line Labels depending on the value of an expression.

A3. Commands for looping structures

Command	description
EXIT	To provide an alternative exit for looping structures, such as FOR...NEXT and WHILE...WEND statements.
FOR ... NEXT	To repeat the execution of a block of statements for a specified number of times.
WHILE ... WEND	To repeat the execution of a block of statements while a certain condition is TRUE.

A4. Commands for string processing

Command	description
LEN	To return the length of a string.
INSTR	To search if one string exists inside another one.
LEFT\$	To retrieve a given number of characters from the left side of the target string.
MID\$	To retrieve a given number of characters from anywhere of the target string.
RIGHT\$	To retrieve a given number of characters from the right side of the target string.
TRIM_LEFT\$	To return a copy of a string with leading blank spaces stripped.
TRIM_RIGHT\$	To return a copy of a string with trailing blank spaces stripped.
ASC	To return the decimal value for the ASCII code for the first character of a given string.
CHR\$	To return the character for a given ASCII value.
HEX\$	To return a string that represents the hexadecimal value (base 16) of the decimal argument.
OCT\$	To return a string that represents the octal value (base 8) of the decimal argument.
LCASE\$	To return a copy of a string in which all uppercase letters will be converted to lowercase letters.
UCASE\$	To return a copy of a string in which all lowercase letters will be converted to uppercase letters.
STR\$	To convert a numeric expression to a string.
VAL	To return the numeric value of a string expression in integer form.
VALR	To convert a string expression to a real number.
STRING\$	To return a string containing the specified number of the requested character.

A5. Commands for event trapping

Command	description
OFF ALL	To terminate all the event triggers.
OFF ESC	To terminate ESC event trigger.
OFF COM	To terminate COM event trigger.
OFF HOUR	To terminate HOUR event trigger.
OFF KEY	To terminate KEY event trigger.
OFF MINUTE	To terminate MINUTE event trigger.
OFF READER	To terminate READER event trigger.
OFF TIMER	To terminate TIMER event trigger.
ON COM GOSUB	To activate COM event trigger.
ON ESC GOSUB	To activate ESC event trigger.
ON HOUR GOSUB	To activate HOUR event trigger.
ON KEY GOSUB	To activate KEY event trigger.
ON MINUTE GOSUB	To activate MINUTE event trigger.
ON READER GOSUB	To activate READER event trigger.
ON TIMER GOSUB	To activate TIMER event trigger.
LOCK	To hold all the activated event triggers until they are released by UNLOCK.
UNLOCK	To release all the activated event triggers held by LOCK.

A6. System commands

Command	description
AUTO_OFF	To set auto power off timer.
DEVICE_ID\$	To get the serial number of the terminal.
GET_TARGET_MACHINES	To get the model name of the target terminal.
MENU	To create a menu.
POWER_ON	To determine whether to restart or resume the program upon powering on.
RESTART	To restart the system.
SYSTEM_INFORMATION\$	To get information on components.
SYS_SUSPEND	To shut down the system.
CHECK_AID	To verify if the agency ID is correct or not.

A7. Reader commands

Command	description
DISABLE READER	To disable the reader ports of the terminal.
ENABLE READER	To enable the reader ports of the terminal.
SLEEP_READER	To set scanner module to sleep.
GET_READER_DATA\$	To get data that is read from a specified reader port.
GET_READER_DATALEN	To get data length that is read from a specified reader port.
READER_CONFIG_START	To start scanner setting procedure.
READER_CONFIG_END	To end scanner setting procedure.
READER_SENDCMD	To send scanner (CCD) command to change scanner status.
READER_SENDCMD_LASER	To send scanner (Laser) command to change scanner status.
READER_QUERY\$	To query the scanner (CCD) current setting.
READER_QUERY_LASERS	To query the scanner (Laser) current setting.
DECODE	To perform barcode decoding.
SIM_SCANKEY_PRESS	To simulator the “Scan” key press or release.
READER_SETFROMFILE	To set scanner setting by scanner setting file.
READER_GETTYPE	To get scanner type for CCD or Laser.
READER_ALL_SYMBOLOGIES	To set all scan codes on or off.

A8. Buzzer commands

Command	description
BEEP	To assign a beeper sequence to beeper operation.
STOP BEEP	To terminate beeper sequence.
SET_BUZZER_VOL	To set the buzzer volume.

A9. Calendar and timer commands

Command	description
DATES\$	To set or to get the current date.

DAY_OF_WEEK

To get the day of the week.

TIMES

To set or to get the current time.

TIMER

**To return the number of seconds elapsed
since the terminal been powered on.**

WAIT

To set system delay time.

A10. LED command

Command	description
LED	To set the LED indicators.

A11. Keypad commands

Command	description
CLR_KBD	To clear the keypad buffer.
INKEYS	To read one character from the keypad buffer and then remove it.
INPUT_LEN	To set or get input length when used "INPUT" or INPUT_S" command.
INPUT	To take user input from the keypad and store it in a variable.
INPUT_S	To take user input from the keypad, scanning and store it in a variable.
INPUT_S_CARRYENT	To set ENT auto press on/off when use "INPUT_S_CARRYENT" command.
INPUT_S_VIBRATE	To set vibrator on or off when use "INPUT_S_VIBRATE" command.
INPUT_S_SLEEP	To set scanner sleep on or off when use "INPUT_S_SLEEP" command.
INPUT_MODE	To set the display mode of the input data.
KEY_CLICK	To enable or disable the key click sound.
ALPHA_LOCK	To set the ALPHA state for input mode.
GET_ALPHA_LOCK	To get information of the ALPHA state for input mode.
GET_KEY_CLICK	To get current key click status.
KEYPAD_BL_TIMER	To set or get keypad backlight timer.
KEYPAD_BL	To set keypad backlight on/off.
DEF_PKEY	To change the definition of programmable key (P1 & P2) .
SCANKEYPWON	To set power on by scan key.
GET_SCANKEYPWON	Get state for power on by scan key.

A12. LCD Commands

Command	description
BACK_LIGHT_DURATION	To specify how long the backlight will last once the terminal been turned on.
LCD_CONTRAST	To set the contrast level of the LCD.
CURSOR	To turn on/off the cursor indication in the activated TextBlock.
CURSOR_X	To get the x coordinate of the current cursor position in the activated TextBlock.
CURSOR_Y	To get the y coordinate of the current cursor position in the activated TextBlock.
LOCATE	To move the cursor to a specified location in the activated TextBlock.
FILL_RECT	To fill a user defined color rectangular area in the activated TextBlock.
ICON_ZONE_PRINT	To enable or disable the statusbar.
PRINT	To display data in the activated TextBlock.
CLR_RECT	To clear a rectangular area in the activated TextBlock. The cursor position is not affected after the operation.
CLS	To clear the activated TextBlock.
SHOW_IMAGE	To put a rectangular bitmap in the activated TextBlock.
CLR_EOL	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
DISPFONT_SETFONT	To set user font from font file.
DISPFONT_INFO_TYPE	To get font type.
DISPFONT_INFO_HEIGHT	To get font height.
DISPFONT_INFO_WIDTH	To get font width.

A14. TextBlock commands

Command	description
DEFINETEXTBLOCK_COLOR	To define the TextBlock setting and the background using color or default background color.
DEFINETEXTBLOCK_IMAGE	To define the TextBlock setting and the background using bitmap file or default background color.
SETTEXTBLOCK	To enable the specific TextBlock.
RESETTEXTBLOCK	To disable the specific TextBlock.
PRINTTEXTBLOCK	To print Text to specific TextBlock.
GETTEXTBLOCKCUR_X	To get the x coordinate of the current TextBlock position.
GETTEXTBLOCKCUR_Y	To get the y coordinate of the current TextBlock position.
SETTEXTBLOCKCUR	To set specific TextBlock as active TextBlock and set position.
SHOWTEXTBLOCKCURSOR	To show or hide TextBlock cursor.
TEXTBLOCK_SETBGCOLOR	To set default background color.
TEXTBLOCK_SETBGIMAGE	To set default background image for bitmap file.
TEXTBLOCK_MODE	To set or get TextBlock mode for single or multi layer.
SWITCHTEXTBLOCK	To switch TextBlock.

A15. File manipulation commands

Command	description
OPENIN	To open (r) a file and get the header of the file for further processing.
OPENOUT	To open (w+) a file and get the header of the file for further processing.
OPENUP	To open (r+) a file and get the header of the file for further processing.
CLOSE	To close a file.
BGET	To read a byte from a file. The current position is updated after reading.
BGETEXT	To read a specified number of bytes from a file. The current position is updated after reading.
GET\$	Read a line terminated by a null character “\0” from a file.
BPUT	To write data to a file.
EOF	To check if file pointer of a file reaches end of file.
PTR	To get or move the file pointer position of a file.
EXT	To get or change file length of a file.
GET_FILE_ERROR	To get the file error code.
DBMS_INIT_SEARCH	To initiate the file search in disk.
DBMS_INIT_SEARCHADV	To initiate the advance file search in disk.
DBMS_CLOSE_SEARCH	To close the file search in disk.
DBMS_APPEND_DATA	To increase one record on the file end.
DBMS_DEL_DATA	To delete the appointed record in the file.
DBMS_EMPTY	To remove all the record in the file.
DBMS_FIND_RECORD	To search the designated field. This command only supports forward search.
DBMS_FIND_RECORD_B	To search the designated field. This command only supports backward search.
DBMS_GET_COUNT	To obtain the figure of all records in the file.
DBMS_GET_DATAS	To read the data of appointed field in the appointed record.
DBMS_UPDATE_DATA	To revise the data of appointed field in appointed field record.

A16. Vibrator commands

Command	description
VIBRATOR_TIMER	To set or get the vibrator timer.
VIBRATOR	To set the vibrator on/off.

A17. Communication port commands

Command	description
CLOSE_COM	To terminate communication and disable a specified COM port.
OPEN_COM	To enable a specified COM port and initialize communication.
SET_COM	To set parameters of a specified COM port.
READ_COM\$	To read data from a specified COM port.
WRITE_COM	To send a string to the host through a specified COM port.
GET_CTS	To get CTS level.
SET_RTS	To set RTS level.
CLEAR_COM	To clear receiver buffer.
COM_DELIMITER	To change delimiter of sending and receiving string of a specified COM port.
FILE_TRANS	Using FILE_TRANS to upload or download files.
FILE_TRANS_REALTIME	Using FILE_TRANS_REALTIME to upload or download files immediately.
FILE_TRANS_BAUD	To get or set the FILE_TRANS baud rate.
FILE_TRANS_INTERFACE	To get or set the FILE_TRANS interface.

A18. Memory commands

Command	description
RAM_SIZE	To check the total space in disk C.
ROM_SIZE	To check the total space in disk D.
FREE_MEMORY	To check the free space in disk C or disk D.
DISK_USED SIZE	To check the occupied space in disk C or disk D.

A19. Bluetooth commands

Command	description
BT_START	Bluetooth module power enable.
BT_STOP	Bluetooth module power disable.
BT_OPEN	Bluetooth connect.
BT_CLOSE	Bluetooth disconnect.
BT_WRITE	Write characters to Bluetooth module.
BT_READS	Read characters from Bluetooth module.
BT_GETLOCALINFOS	Get Bluetooth information.
BT_SETLOCALSET	Set Bluetooth information.
BT_INQUIRY	Inquiry other Bluetooth module for PA-2010/2110 to connect.
BT_GETDEVICEINFO	To get other bluetooth device information after the search.
BT_DISCONNECTALERT	For bluetooth disconnect alert.
GET_BT_ERROR	To get the bluetooth error code.

A20. USB commands

Command	description
USB_OPEN	To initialize and enable USB port.
USB_CLOSE	To close the USB port.
USB_READS	To read specific number of bytes from USB port.
USB_WRITE	To write specific number of bytes to the PC side.

A21. Simulator (Only for PC simulator) commands

Command	description
COPYFILETOPDT	To copy a file from PC side to PDT.
BACKUPDATAFILETOPC	To backup a file from PDT to PC.
SIM_SETSCANNERTYPE	To setting scan module type.

Appendix B

Scan Module (CCD) Configuration Table

Parameter1	Parameter2	Alphanumeric Entry
5 Indication	1 Power on alert	0:On * 1:None 2:Off
	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	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 ~ 64 4 *
	7 Global max. code length	0 ~ 64 63 *
	8 Inverted image scan	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	<0%> 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	 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

Appendix C

Scan Module (Laser) Configuration Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	7 Code ID position	0: Before code data * 1: After code data
	8 Code ID transmission	0: Disable * 1: Enable
	9 Code length transmission	0: Disable * 1: Enable
	10 Code name transmission	0: Disable * 1: Enable
	12 AIM ID	0: Disable * 1: Enable
7 Scan	6: Global min. code length	0~80 (0: Disable) 3*
	9: Global lock code length	0~80 (0: Disable) 0*
	10: Configurable code length #1	See Note1 (4 characters)
	11: Configurable code length #2	See Note1 (4 characters)
	12: Configurable code length #3	See Note1 (4 characters)
	13:	See Note1

	Configurable code length #4	(4 characters)
	14: Configurable code length #5	See Note1 (4 characters)
	15: Configurable code length #6	See Note1 (4 characters)
	16: Configurable code length #7	See Note1 (4 characters)
	17 : Scan Timeout (Sec)	1~30 See Note2 2*
	18: Idle Timeout (Sec)	0~255 (0: Disable) See Note2 5*
8 String setting	3 Preamble characters settings	0x00 ~ 0xff ASCII code (1-10 characters) 00 *
	4 Postamble characters settings	0x00 ~ 0xff ASCII code (1-10 characters) 00 *
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check Digit	1: One digit * 2: Two digits
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8	0: Disable*

	Code ID setting	0x20 ~ 0xff ASCII code(1 bytes)
	10 Full ASCII	0: Disable * 1: Enable
	13 Transmit Start/Stop Characters	0: Disable * 1: Enable
	14 Italian Pharmacode	0: Disable * 1: Enable
	15 Italian Pharmacode Only	0: Disable 1: Enable *
12 Code 93	1 Read	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
13 Code 128	1 Read	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	10 UCC/EAN 128	0: Disable * 1: Enable
	14 ISBT 128	0: Disable * 1: Enable
	15 ISBT 128 Transmit Identifier Data	0: Disable 1: Enable *
	16 ISBT 128 Pre-Defined Concatenation	0: Disable 1: Enable *

	17 ISBT 128 Form:	0: =A + =% * 1: =A + &; 2: =A + &! 3: =< + => 4: =< + &> 5: &< + => 6: &< + &>
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	11 Transmit Start/Stop Characters	0: Disable * 1: Enable
	12 Dual Field	0: Disable * 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	13 Convert to EAN-13	0: Disable * 1: Enable
16 EAN 13	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)

	12 ISBN conversion	0: Disable * 1: Enable
	13 ISBN Supplement Required	0: Disable * 1: Enable
	14 ISMN conversion	0: Disable * 1: Enable
	15 ISMN Supplement Required	0: Disable * 1: Enable
17 Industrial 2 of 5	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable 1: Mod 10 * 2: Mod 10/10
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable

	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
22 Telepen	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	10 Alpha Telepen	0: Disable * 1: Enable
23 UPCA	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	12 Convert to EAN-13	0: Disable * 1: Enable
	13 Coupon	0: Disable * 1: Enable See Note 3
	14 Coupon Transmit "]C1"	0: Disable 1: Enable *
	15 Transmit Number System	0: Disable 1: Enable *
24 UPCE	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	14 Convert to UPC-A	0: Disable * 1: Enable
	15 Transmit Number System	0: Disable 1: Enable *

25 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
35 UPC/EAN General	1 Supplements Required	0: Disable * 1: Enable
	2 Two Digit Supplements	0: Disable * 1: Enable
	3 Five Digit Supplements	0: Disable * 1: Enable
	4 Two Digit Redundancy	0: Disable * 1: Enable
	5 Five Digit Redundancy	0: Disable * 1: Enable
	6 GTIN Formatting	0: Disable * 1: Enable
36 IATA 2 of 5	1 Read	0: Disable * 1: Enable See Note 4
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
37 TRI-OPTIC	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	10 Conversion	0: Disable * 1: Enable
38 RSS	1 Read	0: Disable * 1: Enable
	2 Transmit Application ID	0: Disable 1: Enable *
	3	0: Disable

	Transmit Check Digit	1: Enable *
	4 Transmit Symbology ID "je0"	0: Disable * 1: Enable

Note1:

There are twelve bar code lock lengths available. Specific code type can be assigned to a lock length.

Code type:

	Hex
CODE11	64
CODE39	65
CODE93	66
CODE128	67
CODABAR	68
INDUSTRIAL_25	6B
INTERLEAVED_25	6C
MSI_PLESSEY	6E
UK_PLESSEY	6F
TELEPEN	70
MATRIX_25	73
TRIOPTIC	7E

Length:

0~80 (0: Disable)

Default setting:

“0000”

Example:

Cmd1	Cmd2	Value
7	10	“670C”
7	11	“6514”
7	12	“6710”

Code 128: Can only read 12 or 16 the length of the barcode.

Code39: Can only read 20 the length of the barcode.

Note2:

ScanTimeout:

The maximum time, in seconds, during which the laser remains on without decoding any barcode.

IdleTimeout:

The maximum time, in seconds, during which the scanner remains idle without any action.

Note3:

When this setting is enable the UCC / EAN 128 barcode can not be read.

Note4:

IATA 2 of 5 only support 13, 15 Digit.

Appendix D

Parameter for Color

In our BASIC SDK function, some have “color” parameter, this appendix will tell you how to set color parameter.

1. Value:

The BASIC parameter value doesn’t support Hex., so you have to set it in DEC.

Our sample will use HEX and when you want to set color, please conversion Dec from Hex value.

2. Example:

Color	Hex value	Dec value
Red	0x0000FF	255
Green	0x00FF00	65280
Blue	0xFF0000	16711680
White	0xFFFFFFFF	16777215
Black	0x000000	0

Red, Green, and Blue are the Primary color of light. If you want to set color “Yellow”, for example, you have to use “Red” and “Green” color.

So, the yellow color hex value is “0x00FFFF”. You can set the color you want by this rule.