Chapter 13. EasyBuilder500 Macro Guide

This document contents the following chapters:
Macro description:
Macro usage description
Macro command and PLC communication (LocalBit, LocalWord)
Macro operation manual
Some notes about using Macro
Compile error message
Example source code

13.1 Macro description

1. Constants and variables
A. Constants
(1) Decimal constant
(2) Hexadecimal constant
(3) ASCII code (character constant)
(4) Boolean: True (not zero), False (zero)
B. Variables
(1) Naming rule:
A variable must starts with a alphabet, other letters can be characters or numbers. And it can not exceed 32 characters long.
(2) Variable types:

* char Character variable
* short Short integer variable (16 bit)
* int Integer variable (32 bit)
* float Floating point variable
* bool Boolean variable
* WORD word variable
* DWORD Double word variable

C. Operator
(1) Assignment operator

* Assignment ： =

(2) Arithmetic operator

* Addition ： +
* Subtraction ： -
* Multiplication ： *
* Division ： /
* Modulo ： %

(3) Comparison operator

* Less than ： <

* Less than or equal ： <=
* Greater than ： >
* Greater than or equal ： >=
* Equal ： = =
* Not equal ： <>

(4) Logic operator

* Conditional AND ： And
* Conditional OR ： Or
* Exclusive OR ： Xor
* Boolean NOT ： Not

(5) Bitwise and shift operator

(a) Shift operator

* Left shift ： <<
* Right shift ： >>

(b) Bitwise operator

* Bitwise AND ： &
* Bitwise OR ： |
* Bitwise XOR： ^
* Bitwise complement： ~

2. Priority of operators

The process order of many operators within an expression is called the priority of operators.

A. Priority of the same kind of operator (From left to right, from up to low)

* Arithmetic operator： ^ → (* , /) → % → (+ , -)
* Shift operator ： From left to right within the expression
* Comparison operator ： From left to right within the expression
* Logic operator ： Not → And → Or → Xor

B.

* Arithmetic operator is prior to Bitwise operator
* Bitwise operator is prior to Comparison operator
* Logic operator is prior to Assignment operator

3. Array

We only support fixed length, 1-D array which is:

1-D array: Array_Name[Array_Size]

The array size can be integer which from 0 to 4294967295

Minimum of array index = 0

Maximum of array index = Array size – 1

Example : Array[MAX] MAX = 100

Minimum of array index = 0

Maximum of array index = 99 (100 – 1)

4. Expression

A. Operation object

(1) Constants

(2) Variables

(3) Array

(4) Function

B. Components of expression

An expression is combined operation objects with operators by following specific rules.

5. Statement

A. Definition statement

(1) type name Define the type of name

* ex: int a

(2) type name[constant] Define the type of array name

* ex: int array[10]

B. Assignment statement

The form is ： Variable = Expression

ex: a = 2

C. Logic statement and branches

(1) One-line format

```
If Condition Then
        [Statements]
End If
```

ex:

```
If a == 2 Then
        b = 1
Else
        b = 2
End If
```

(2) Block format

```
If Condition Then
        [Statements]
[Else [If Condition – n Then
        [Else_If_Statements] ….
[Else
        [Else_Statements]]
] ]
```

End If

ex:

```
If a == 2 Then
        b = 1
Else If a == 3
        b = 2
Else
        b = 3
End If
```

Syntax description：

Condition	Necessary. This is a control statement. It will be FALSE when the value of condition is 0; and will be TRUE when the value of condition is 1.
Statements	It is optional in block format statement but necessary in one-line format without ELSE. The statement will be executed when the condition is TRUE.
Condition – n	Optional. See Condition.
Else_If_Statements	Optional in one-line or multi-line format statement. The elseifstatement will be executed when the relative Condition – n is TRUE.
Else_Statements	Optional. The elsestatement will be executed when Condition and Condition—n are both FALSE.

(3) Multi-branches: Select Case statement

```
Select Case Test_Expression
[Case Expression – n
        [Staremens – n]] ….
        Break
[Case Else
        [Else_Statements]]
        Break
End Select
```

ex:

```
Select Case a
Case 1
        b = 1
        Break
```

```
Case 2
        b = 2
        Break
Case Else
        b = 3
End Select
```

Syntax description：

Test_Expression	Necessary. This can be any value or char expression.
Expression – n	Necessary if there is Case statement. This can be char or integer value.
Statements – n	Optional. This can be one-line or multi-lines format. The Statements—n will be executed when TestExpression and the relative expression—n are equal.
Else_Statements	Optional. This can be one-line or multi-lines format. The Elsestatements will be executed when TestExpression is not equal to any one of expression—n.
Case Else	Necessary.
End Select	Necessary.

D. Looping control

(1) For –Next Statement

Use this for fixed execution counts. To means increase by step while Down means decrease by step.

```
For Counter = Start To (Down) End [Step step]
        [Statements]
Next [Counter]
```

ex:

```
For a = 0 To 10 Step 2
    b = a
Next a
```

Syntax description：

Counter	Necessary. The counter of looping control. It can be integer or character.
Start	Necessary. The initial value of Counter.
End	Necessary. The end value of Counter

step	Optional. The increment/decrement step of Counter. It can be integer and can be omitted when value is 1.
Statements	Optional. Statement block between For and Next which will be executed fixed counts.

(2) While – Wend statement

Loop controlled by Condition. When Condition is TRUE, the statements will be executed repetitively until the condition turns to FALSE.

```
While Condition
    [statements]
Wend
```

ex:

```
While a == 2
    b = b + 1
    GetData(a, LB_bin, 5, 1)
Wend
```

Syntax description：

Condition	Necessary. Logic expression which control the execution of statements.
Statements	Optional. Statement block.

(3) break

Used in looping control or select statement. It skips immediately to the end of the statement.

(4)continue

Used in looping control statement. It quits the current iteration of a loop and starts the next one.

(5) return

To stop executing the current method.

6. Reserved keywords

The following keywords are reserved for Macro which can not be used in function name, array name, or variable name.

+ , - ,* , / ,^, mod, >= ,>, < ,<=, <> ,== , And, Or, Xor, Not,<<,>>,= ,& ,|,^,~

If , Then, Else, End If, Select ,Case ,For, To, Down Step, Next, while, wend break ,continue, return

13.2 Macro usage description

1. Local variables and global variables

A. Local variables: Its value remains valid only within a specific statement.

B. Global variables: Its value always remains valid after declaration.

When local variable and global variable have the same declaration name, only the local variable will be valid.

2. Variable and constant initialization

A. Variable initialization

(1) Initialize a value of variable in the declaration statement directly. e.g: int h = 9

(2) Use assignment operator to initialize a value after declaration.

(3) Array initialization

Format: int g[10] = { 1,2,3, , 3 }

The initial values are written within the {} and divided by comma (,). These values are assigned orderly from left to right starting from array index=0.

B. Constants

Macro supports

(1) Decimal integer constant

(2) Hexadecimal integer constant: start with 0x

(3) Character constant

(4) Boolean constant: True / False

3. Boolean variables and Boolean expressions

A. Boolean variables

True or False. Not zero value means TRUE while zero value means FALSE.

B. Boolean expressions

The value of Boolean expression is zero means FALSE.

The value of Boolean expression is not zero means TRUE.

4. Declaration statement

A. Declaration outside a function is a global variable declaration.

B. Declaration inside a function is local variable declaration. This declaration must at the very beginning of a statement within a function. Other statements before declaration statements will cause compiler error.

For example：

```
Macro_Command main( )
char i

i = 9//    Assign statement within declaration area causes compiler error
int g[10]

For g[2] = 0 To 2
```

```
    g[3] = 4
    Next g[2]
    End Macro_Command
```

5. Function call and passing parameters

A. Function call

A function must be defined before its execution. Otherwise, a compiler error ‘Function not defined’ will occur.

For example：

```
Macro_Command main()
    int    i
    i = Func(i)//    call an undefined function causes compiler error
End Macro_Command

Sub int Func(int i)
    int h = 9
    i = 9 * h
    return i
End Sub
```

B. Passing parameters

(1) Passing by value through local variable.

(2) Through the same global variables

6. Main Function

Macro must have one and only one main function which is the execution start point of Macro. The format is:

```
Macro_Command Function_name( )



End Macro_Command
```

13.3 Macro command and PLC communication (LocalBit, LocalWord)

Usage: Communicate with PLC through a function library

In the command program, Macro can communicate with data in the PLC. The function GetData(…) can receive data from the PLC through EasyView. The function SetData(…) can set data to the PLC through EasyView. The Macro command handles the communication details.

1. GetData(short DestData, CString strAddrType, int nAddrOffset, int nDataCount)

Description: Get data from PLC by filling a dialog

Parameters：

DestData	The address of data to get
strAddrType	The type and encoding method of PLC address
nAddrOffset	The address offset of PLC
nDataCount	Number of data

Return value：

None

2. SetData(short DestData, CString strAddrType, int nAddrOffset, int nDataCount)

Parameters：

DestData	The address of data to set
strAddrType	The type and encoding method of PLC address
nAddrOffset	The address offset of PLC
nDataCount	Number of data

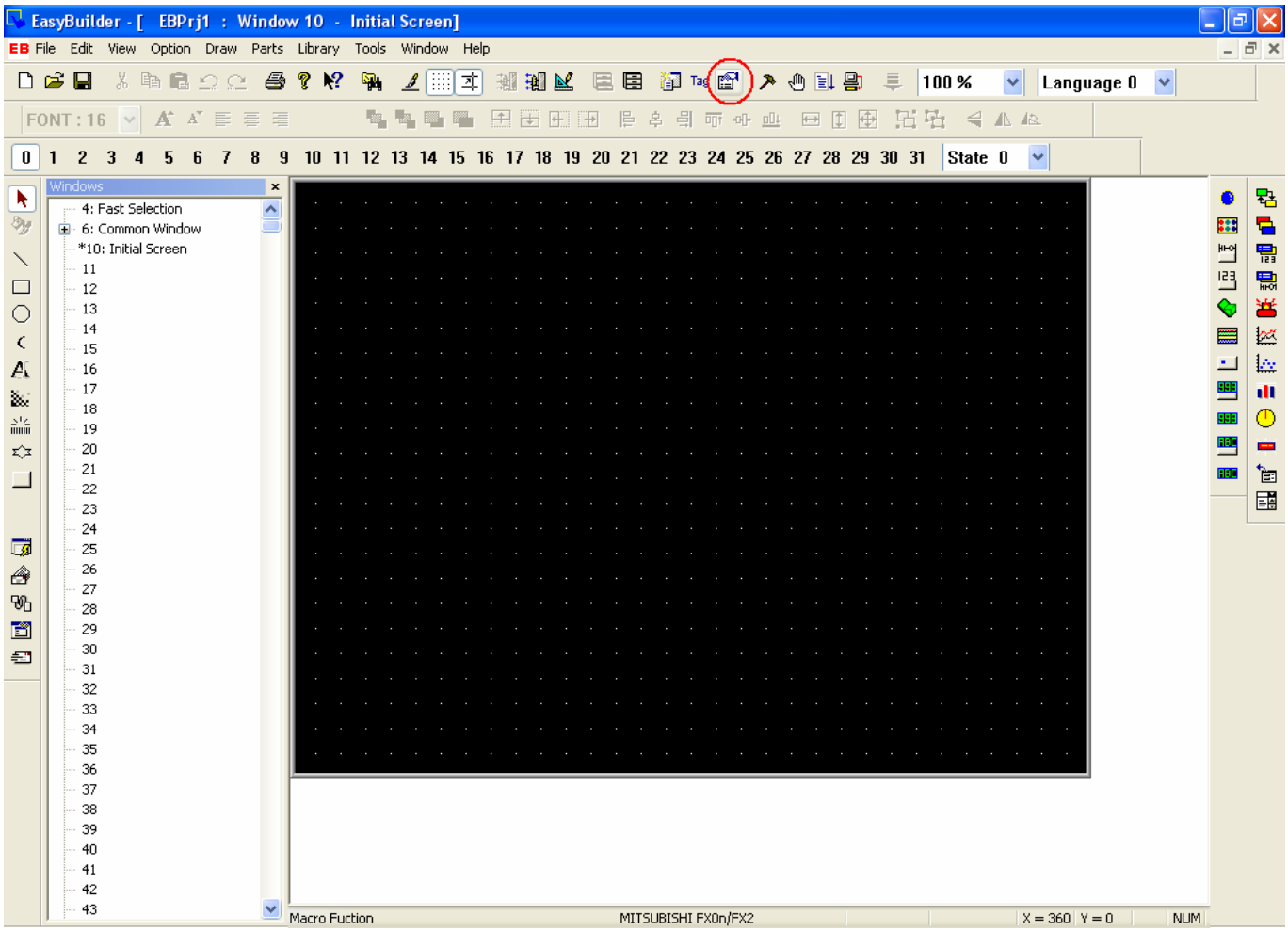
Return value：

None

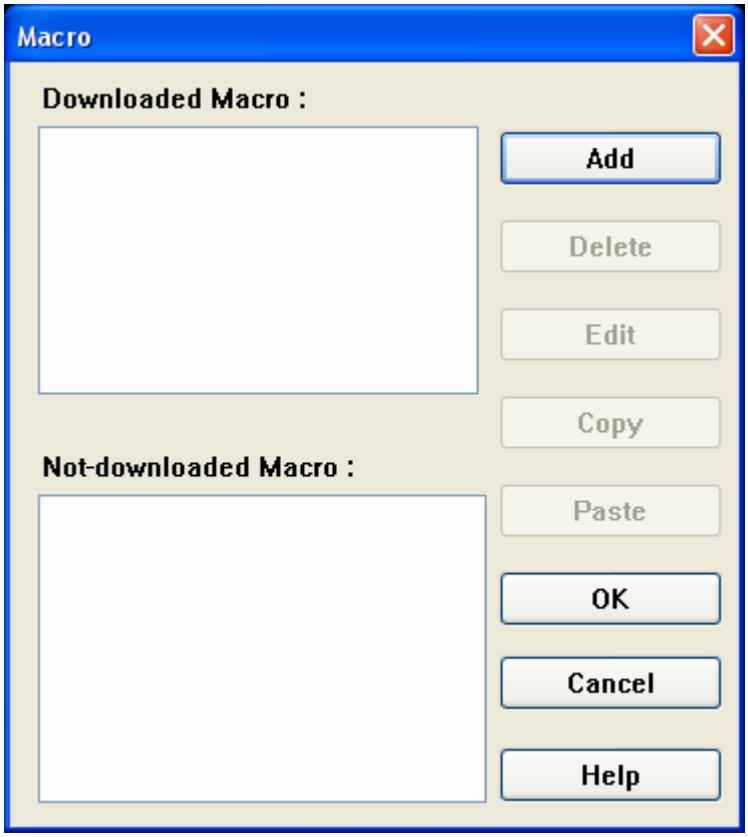
13.4 Macro operation manual

1. Macro programming can be divided into three steps

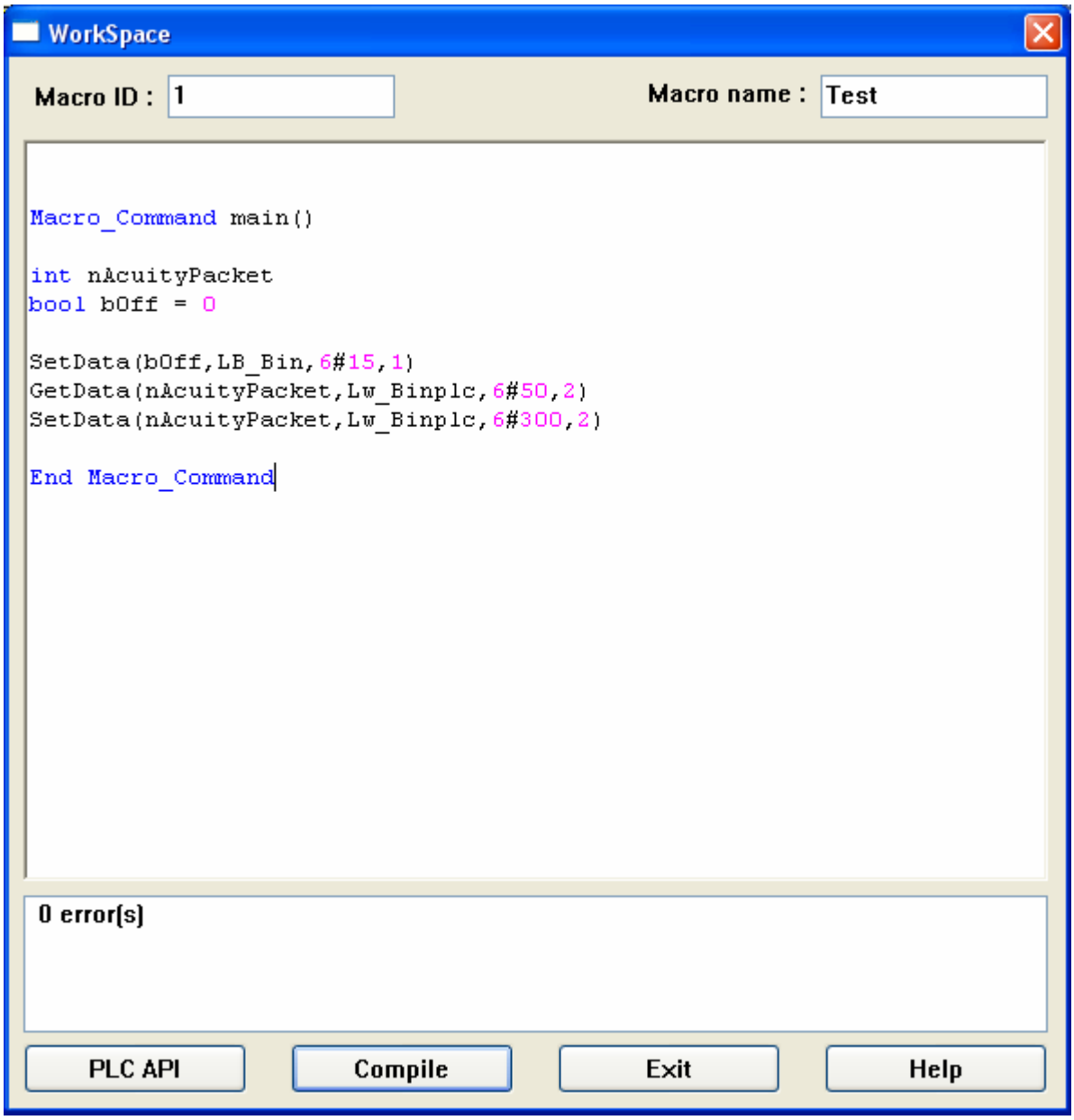
Step 1: Open the Macro dialog (MacroControlDlg) from Tools->Macro in the main screen of EB500.



Step 2: Each Macro can be copied, deleted or edited in MacroControlDlg dialog. The source code of Macro can be edited by opening MacroWorkSpaceDlg dialog.



Step 3: Editing the source code of the Macro. Make sure the name and number of the Macro are correct. Compile the Macro and fix the error message.

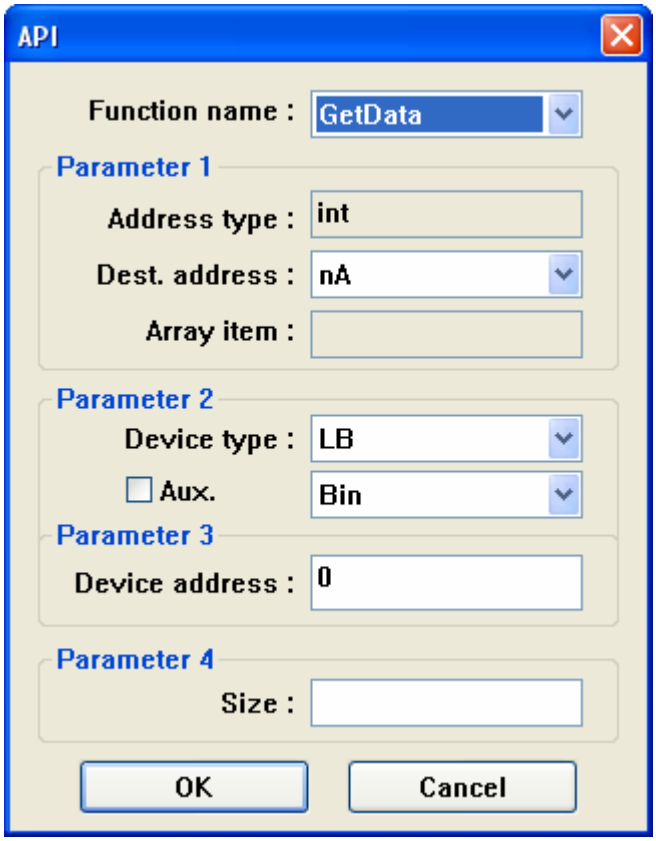


2. Editing the communication source code of Macro

A. Input

(Step 1:) Enter the keyword “Insert” in the proper position.

{ Or by moving the cursor to the proper position and push PLC button }



(Step 2:) Select functions and parameters of the library in Library Editing Dialog. Push button “OK” to enter this sub-function; push button “Cancel” to abort this sub-function.

B. Edit

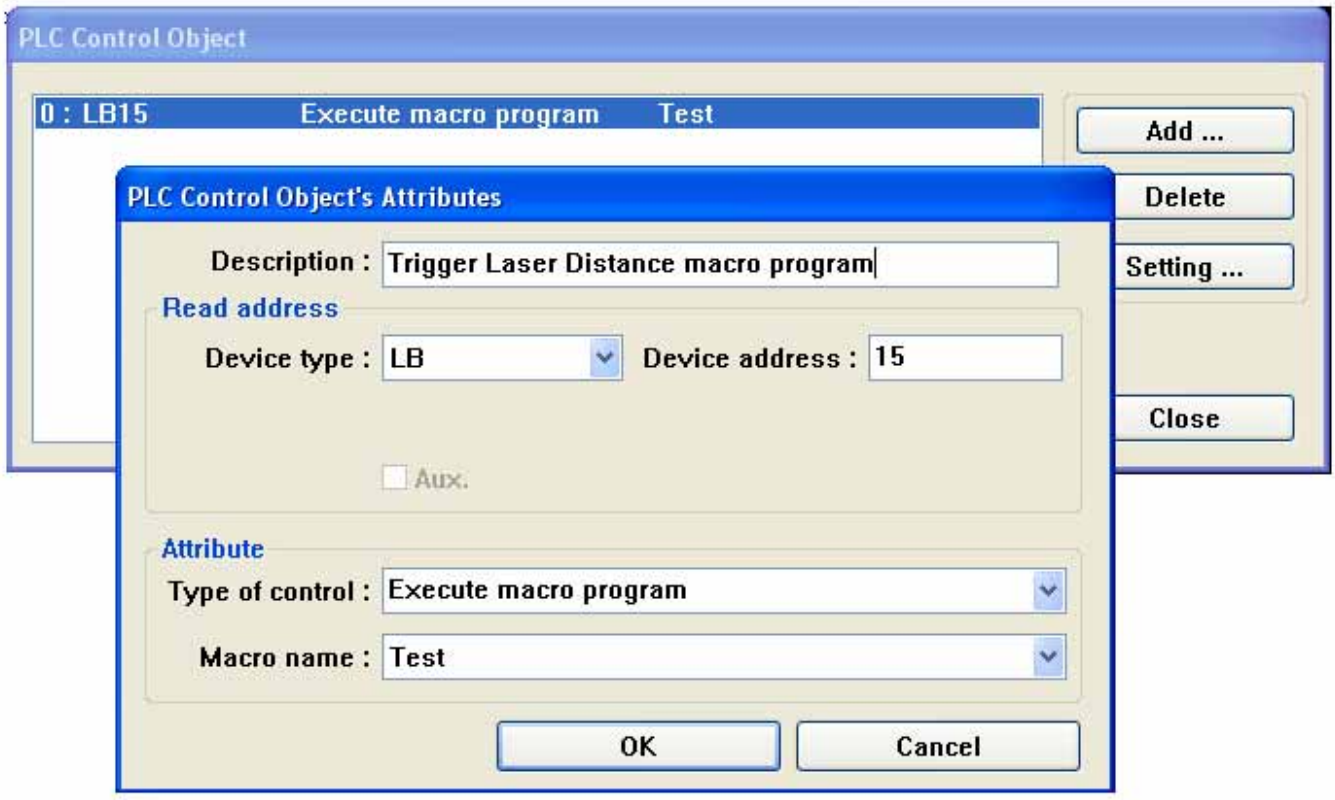
Move the cursor onto the modifying position to modify it. Follow the detail Step1 and Step of (1:) list above.

C. Delete

Highlight the selected function and push the button “Delete” to delete it.

3. Trigger condition of Macro

(Step 1:) Select control type to “Execute Macro Program” in the object property dialog of PlcControl.



(Step 2:) Select a MacroID and define a trigger condition in the object property dialog PlcControl.

13.5 Some notes about using Macro

1. Limitation of storage space of Macro

The size of a Macro in a eob file is limited by the storage capacity. The maximum storage space of local variables in a Macro is 4K bytes.

2. Limitation of maximum lines of Macro to execute

There are at most 256 Macros in a eob file.

3. Macro may possibly cause deadlock of the machine.

When there is a infinite loop in a Macro without communicating with PLC

When the size of array exceeds the storage space in a Macro.

4. The Limitation of communication speed of Macro

The execution of Macro may be a little slow down when communicating with PLC. This is caused by the data transferring time.

13.6 Compile error message

1. Error message format

(: Error_message_number) Error_Message
ex：(:37) undeclared identifier : i
When there are compile errors, the error description can be referenced by the compile error message number.

2. Error description

(1：) "Syntax error:" “'identifier'
There are many possibilities to cause compiler error.

for example :

```
Macro_Command main()
char i, xyz //this is an unsupported variable name ,”Error message: “Syntax error: xyz”
End Macro_Command
```

(2：) ‘identifier’ used without having been initialized
Must define the size of an array during declaration.

For example：

```
Macro_Command main()
char i
int g[i]//    i used without having been initialized
End Macro_Command
```

(3：) redefinition error : ‘identifier’
The name of variable and function within its scope must be unique.

For example：

```
Macro_Command main()
int g[10],   g//    error
End Macro_Command
```

(4：) function name error : ‘identifier’
reserved keywords and constant can not be the name of a function

For example：

Macro_Command If()// error

End Macro_Command

(5：) parentheses have not come in pairs
Statement missing “(“ or “)”

For example：

Macro_Command main)// missing (
End Macro_Command

(6：) illegal expression without matching ‘If’

(7：) illegal expression (no ‘Then’) without matching ‘If’

(8：) illegal expression (no ‘EndIf’)

(9：) illegal ‘EndIf’ without matching ‘If’

(10：) illegal ‘Else’
The format of “If” statement is:
If [logic expression]Then

[Else [If [logic expression] Then]]

EndIf
Any format other than this format will cause compile error.

(11：)‘Case’ expression not constant
"There should be constant behind “Case” "
(12：) ‘Select’ statement contains no ‘Case’
"Missing “Case” behind “Select” "

(13：) illegal expression without matching ‘Select Case’
"Missing “expression” behind “Select Case” "

(14：) ‘Select’ statement contains no ‘End Select’
"Missing “End Select” statement

(15：) illegal ‘Case’
Illegal “Case” statement"

(16：) illegal expression (no 'Select') without matching ‘End Select’
"Unfinished “Select” statement before “End Select” "

The format of “Select Case” statement is:
Select Case [expression]
Case [constant]

Case [constant]

Case [constant]

Case Else

End Select
Any format other than this format will cause compile error.

(17：) illegal expression (no 'For') without matching ‘Next’
" “For” statement error: missing “For” before “Next” "

(18：) illegal variable type (not interger or char)
"Should be integer of char variable"

(19：) variable type error
"Missing assign statement"

(20：) must be key word ‘To’ or ‘Down’
"Missing keyword “To” "

(21：) illegal expression (no 'Next')
"Missing “Next” statement"

The format of “For” statement is:

For [variable] = [initial value] To [end value] [Step]

Next [variable]
Any format other than this format will cause compile error.

(22：) ‘Wend’ statement contains no ‘While’
" “While” statement error: missing “While” before “Wend” "

(23：) illegal expression without matching ‘Wend’
Missing “Wend” statement"

The format of “While” statement is:
While [logic expression]

Wend
Any format other than this format will cause compile error.

(23：) syntax error : ‘Break’
“Break” statement can only be used in “For”, “While”, or “Select Case” statement
“Break” statement takes one line of Macro.

(25：) syntax error : ‘Continue’
"Illegal “Continue” statement"
“Continue” statement can only be used in “For” statement, or “While” statement
“Continue” statement takes one line of Macro.

(26：) syntax error
"Expression error"

(27：) syntax error
"Illegal operation object"
The mismatch of operation object in expression cause compile error.

For example：

```
Macro_Command main( )

int a, b

For a = 0 To 2
    b = 4 + xyz //illegal operation object
```

Next a

End Macro_Command

(28：) must be ‘Macro_Command’
"Missgin “Macro_Command” "

(29：) must be key word ‘Sub’
"Missing “Sub” "
The format of function declaration is:
Sub(Macro_Command) [data type] function_name(…)

End Sub(Macro_Command)
Any format other than this format will cause compile error.

(30：) number of parameters is incorrect
"Mismatch of the number of parameters "

(31：) parameter type is incorrect
"Mismatch of data type of parameter"
The parameters of a function must be equivalent to the arguments passing to a function to avoid compile error.

(32：) variable is incorrect

(33：) function name : undeclared function
"Undefined function"

(34：) expected constant expression
Illegal member of array

(35：) invalid array declaration
Illegal definition of array

(36：) array index error
Illegal index of array

(37：) undeclared identifier : + ‘identifier’
"Undefined symbol"
Any variable or function should be declared before use.

(38：) PLC encoding method is not supported
The parameter of GetData(…) , SetData(…) should be legal PLC address.

(39：) ‘idenifier’ must be integer, char or constant
Should be integer, character or constant
The format of array is:
Declaration: array_name[constant] (constant is the size of the array)
Usage: array_name[integer, character or constant]
Any format other than this format will cause compile error.

(40：) execution syntax should not exist before variable declaration or constant definition
"Illegal Macro statement before declaration statement "
For example：

```
Macro_Command main( )
int a, b
For a = 0 To 2
b = 4 + a
int h, k     // declaration statement position error
Next a
End Macro_Command
```

(41：) float variables cannot be contained in shift calculation
"Floating point can not bitwise shift"

(42：) function must return a value
"Missing function return value "

(43：) function should not return a value
"Function can not return a value"

(44：) float variables cannot be contained in calculation
"Illegal Float data type in expression"

(45：) PLC address error
"Error PLC address"

(46：) array size overflow (max. 4k)
"Stack can not exceed 4k bytes"

(47：) macro command entry function is not only one

"Only one main entrance in the Macro is allowed"

(48：) macro command entry function must be only one

"Too many main entrance: " 'identifier'

The only one main entrance of Macro is:

Macro_Command function_name( )

End Macro_Command

(49：) a extended addresse's station no. must be between 0 and 7

Ex:

SetData(bits[0] ,LB_Bin ,2#300,100)

2#300→max station:7

13.7 Example source code

1. “For” statement and other expressions (arithmetic, bitwise shift, logic and comparison)

```
Macro_Command main()
int a[10], b[10], i

b[0]= (400 + 400 << 2) / 401
b[1]=    22 *2 - 30 % 7
b[2]= 111 >> 2
b[3]= 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
b[4]= Not 8 + 1 And 2 + 1 Or 0 + 1 Xor 2
b[5]= 405 And 3 And Not 0
b[6]= 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
b[7]= 6 – (~4)
b[8]= 0x11
b[9]= 409

For i = 0 To 4 Step 1
      If(a[0] == 400) Then
            GetData(a[0],3x_Bin,0,9)
            SetData(b[0],3x_Bin,11,10)
End If
Next

End Macro_Command
```

2. while, if, break,

```
Macro_Command main()
int b[10], i
i = 5
While i == 5 - 20 % 3
            GetData(b[1], 3x_Bin, 11, 1)

            If b[1] == 100 Then
                  Break
            End If

Wend
```

```
End Macro_Command
```

3. Global variables and function call

```
char g

Sub int fun(int j, int k)
int y

SetData(j, LB_Bin, 14, 1)
GetData(y, LB_Bin, 15, 1)
g = y

Return y

End Sub

Macro_Command main()
int a, b, i

a = 2
b = 3
i = fun(a, b)
SetData(i, LB_Bin, 16, 1)

End Macro_Command
```

4. “If” statement

```
Macro_Command main()
int K[10], j

For j = 0 To 10
    k[j] = j
Next

If k[0] == 0 Then
    SetData(k[1], 3x_Bin, 0, 1)
End If

If k[0] == 0 Then
```

```
    SetData(k[1], 3x_Bin, 0, 1)
Else
        SetData(k[2], 3x_Bin, 0, 1)
End If

If k[0] == 0 Then
    SetData(k[1], 3x_Bin, 1, 1)
Else If k[2] == 1 Then
    SetData(k[3], 3x_Bin, 2, 1)
End If

If k[0] == 0 Then
    SetData(k[1], 3x_Bin, 3, 1)
Else If k[2] == 2 Then
    SetData(k[3], 3x_Bin, 4, 1)
Else
    SetData(k[4], 3x_Bin, 5, 1)
End If

End Macro_Command
```

5. “Select” statement

```
Macro_Command main()
int K[10], j

For j = 0 To 10
        k[j] = j
Next

Select Case k[1]
    Case 1
    j = 1
    Break
    Case 2
    j = 2
        Break
End Select

SetData(j, 3x_Bin, 0, 1)
End Macro_Command
```

6. “while” statement

```
Macro_Command main()
char i = 0
int a[13], b[14], c = 4848

b[0] = 13

While b[0]
    a[i] = 20 + i * 10

    If a[i] == 120 Then
        c =200
        Break
    End If

    i = i + 1
Wend

SetData(c, 3x_Bin, 2, 1)
End Macro_Command
```

7. “Break” and “Continue” statement

```
Macro_Command main()
char i = 0
int a[13], b[14], c = 4848

b[0] = 13

While b[0]
    a[i] = 20 + i * 10
    If a[i] == 120 Then
        c =200
        i = i + 1
        Continue
    End If

    i = i + 1
```

```
        If c == 200 Then
            SetData(c, 3x_Bin, 2, 1)
        Break
        End If
    Wend

    End Macro_Command
```

8. array

```
    Macro_Command main()
    int a[25], b[25], i

    b[0] = 13

    For i = 0 To b[0] Step 1

        a[i] = 20 + i * 10
    Next

    SetData(a[0], 3x_Bin, 0, 13)
    End Macro_Command
```