2	X	Center X coordinates
3	Y	Center Y coordinate
4	r	Circle radius
5~7	(keep)	Now is zero

The thickness of the :

Word	Mnemonics	Describe
0	type	Thickness code 8
1	(keep)	Now is zero
2	The thickness of the	Thickness of stroke.Unit: pixel
3~7	(keep)	Now is zero

The usual color value is :

color	The numerical
black	0x0000
blue	0x0010
green	0x0400
cyan	0x0410
red	0x8000
magenta	0x8010
brown	0x8400
gray	0xc618
Dark grey	0x8410
Light blue	0x001f
Light green	0x07e0
Pale blue	0x07ff
pink	0xf800
Pale magenta	0xf81f
yellow	Oxffe0
white	0xffff

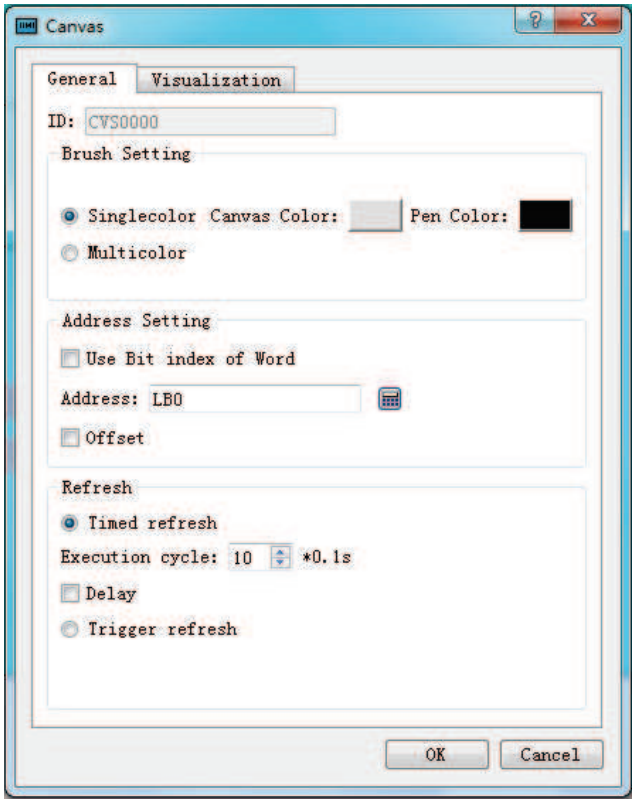
Address description of the reception area of the selected report

<table>
<tr><td>Word number</td><td colspan="2">instructions</td></tr>
<tr><td>0</td><td colspan="2">Select the X coordinate of the point</td></tr>
<tr><td>1</td><td colspan="2">Select the Y coordinate of the point</td></tr>
<tr><td>2</td><td colspan="2">Click the element, where the sequence number of the selected point</td></tr>
<tr><td rowspan="3">3</td><td>value</td><td>describe</td></tr>
<tr><td>0</td><td>A point of a point, the center of a circle or the beginning of an arc</td></tr>
<tr><td>1</td><td>The end point of a line</td></tr>
</table>

		or arc

6.6.15 Canvas

The function of the control is to draw the data based on the set of data in the specified control range, and the source of the data is described in detail.



- Display Settings: specific reference "Bit switch" display Settings。
- Brush setting
  - Monochrome brush

    Canvas color: setting the background color of the canvas control

    Brush color: set the brush control to draw the color of the graph

    A monochrome brush USES a bit-state state to determine whether to draw, for on, to draw the element, not to draw the element for off.
  - multicolor

    The background color is transparent, the drawing color is determined by the address value, the color is used in the ARGB32 color format, and the appropriate ARGB32 color format is required, and the corresponding color is drawn in the canvas area.
- Address setting
  - Index of use words

When using a monochrome brush, select this option and set the state of a word register to draw conditions.

- Address

  Set up the location of the area data source

- Address migration

  Address migration

  For example：

  The address is set to LB0, the address migration is set to LW0, drawing the actual address of the area address data =LB（0+LW0）

➢ Refresh

- Timing refresh

  Execution cycle: set the canvas refresh time and refresh the canvas periodically

- Delay

  Set the delay corresponding cycle to refresh the current canvas

- Trigger refresh

  Trigger address: set the trigger judgment address

  Trigger mode: set the trigger

  ON--->OFF

  ON<---OFF

  ON<-->OFF

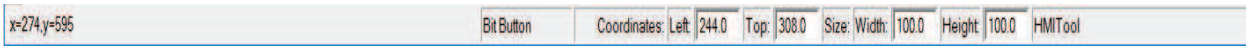
  Automatic reset: select this option to restore the address state to the previous state after setting the conditions.

➢ Canvas data：

- The canvas control is a 100*100 pixel size control with 10,000 addresses to control each element color in its canvas area.
- Monochrome brush: when using a monochrome brush address, the LB9999 address bit state is controlled by the LB9999 address bit state to control the corresponding elements, and the local site is drawn when the local site is on
- Multicolor brush: when using multicolor brush address for LW0, the corresponding element is drawn by the LW0~ LW9999 address value, using the ARGB32 color format, setting one address data for the corresponding color value in ARGB32, which can be painted in the canvas.

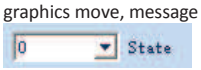
6.7 HMITOOL Control Instructions

HMITOOL configuration software provides full-featured basic controls. Users can double-click or right-click the control in the view area to set the properties. The position and size of controls can be modified manually in the status bar below or directly drag the mouse.



Address input of controls: It is available to input an address directly through a keyboard when it needs to operate on the PLC address. The address name is case-insensitive.

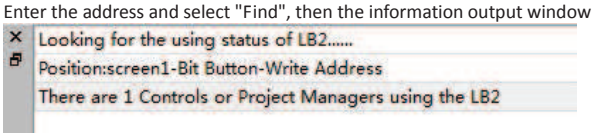
Status combo box: In the toolbar, there is a combo box where to display and change the state of bit button, bit indicator, multi-state button, multi-state indicator, graphics move, message display and picture display.



When a control of the above type is selected, its current state is displayed in the status combo box that users can pull down to change the current state.

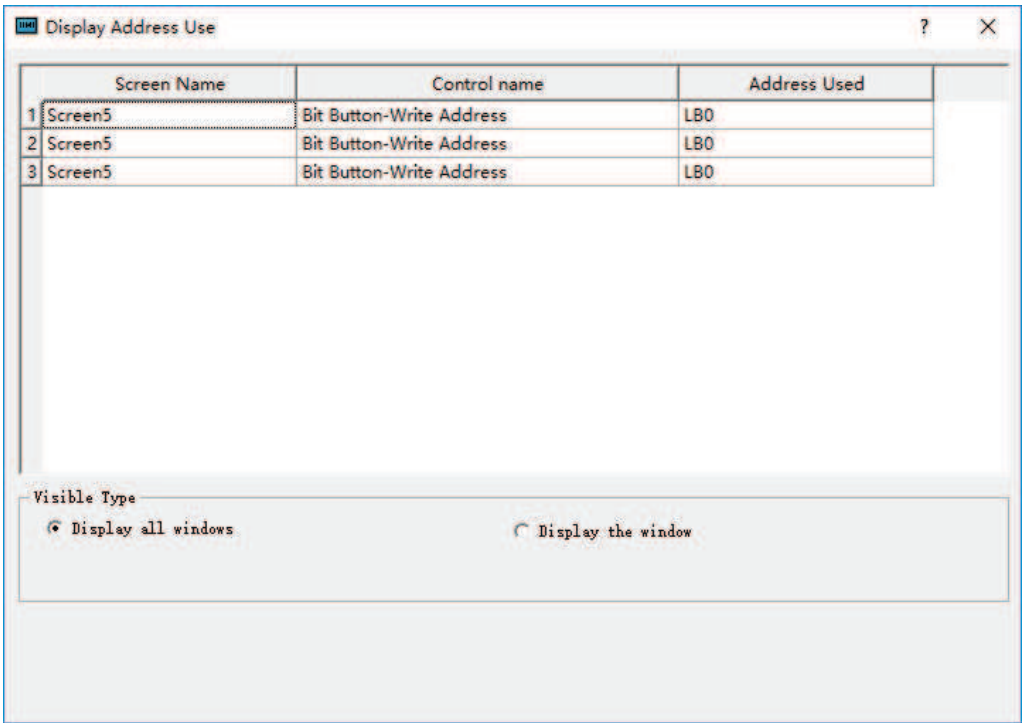
Address Search: Query the use of an address and list the information of control occupying this address; double-click the information item to select the control.

Enter the address and select "Find", then the information output window will list the search results:



Choose the search result and double-click to select the control occupying the address currently.

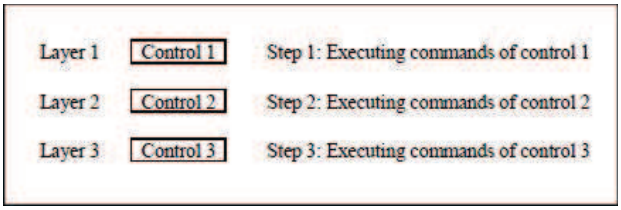
Address Display: Click the "Address Display" to show the list of addresses used by the configuration project; double-click a message, then it will pop up the property page of control using this address so as to set the property settings easily and quickly.



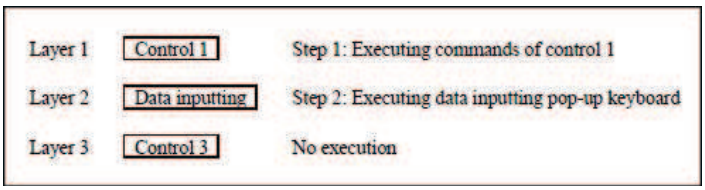
In the Display Mode, you can specify the selected address information or display all the screens.

Overlay of control: You can execute multiple controls by one operation through stacking controls. HMITOOL supports stacking up to 32 controls. When multiple controls are overlaid and touched, the corresponding operations are performed in the order in which they are superimposed, as shown in Figure below:

Superimposing the control 1, 2 and 3, after the touch, execute in turn the command 1, 2 and 3.



If there is a screen switch button in the overlaid control, it will switch the screen when it comes to the turn of this command. The superimposed button commands that are after the screen button will not be executed. Similarly, if there is a pop-up input keyboard, it will pop up a keyboard while without executing the after commands.



7. Macro

Macro is an advanced control method for touch screen, which provides more strong functions for touch screen. Through programming for macro command, the touch screen is given the same functions as logic and arithmetic operation with PLC. Using the macro flexibly is capable to achieve many strong functions that are unavailable for conventional components and to perfect the human-computer interface more.

HMITOOL provides new full macros that are different from the script language mode of other human-computer interfaces, and these macros are compatible with standard C Language (ANSI C89). As there are many literatures about C Language and this information are available easily, this chapter will not introduce syntax and basic knowledge in details to review the relevant basis of different macros but emphasize the establishment and usage of macros by instances.

This chapter will explain basic C Language briefly, relevant usage of macroinstruction and considerations.

Contents：

7.1 Introduction to C language

7.1.1 Data type of C language

1. Integer

Integer includes integer constant and integer variable. The integral constant is the integer constant. In C language, there are three kinds of integer constant: octal, hexadecimal and decimal.

2. Integer constant

1. Octal integer constant

The octal integer constant must begin with 0. That is to say, 0 is the prefix of octal integer constant. Its value is 0-7.Octal constant is usually unsigned. The followings are the legal octal:
015 (decimal:13) 0101 (decimal:65) 0177777 (decimal:65535)
The followings are the illegal octal: 256 (without prefix 0) 03A2 (contained non-octal codes) -0127(with negative sign)

2. Hexadecimal integer constant

The hexadecimal integer constant is prefixed with 0X or ox. Its value is 0-9, A-F or a-f.
The followings are the legal hexadecimal:

0X2A (decimal:42) 0XA0 (decimal:160) 0XFFFF (decimal: 65535) The followings are the illegal hexadecimal:
5A (without prefix 0X) 0X3H (contained non-hexadecimal codes)

3. Decimal integer constant

The octal integer constant has no prefix. Its value is 0-9. The followings are the legal decimal:
237 -568 65535 1627

The followings are the illegal decimal:

023 (prefix 0 is forbidden) 23D (contained non-decimal codes)

In the program, these notations are distinguished by prefix. Therefore, do not mistake the prefix in writing to avoid incorrect result.

4. When the suffix of integer constant is on the 16-bit computer, its basic integer is 16-bit. Therefore, the indicated figure value is limited. The decimal unsigned constant is within 0 ~ 65535, and the signed range is -32768 ~+32767.The unsigned octal number is ranged within 0 ~ 0177777. The unsigned hexadecimal number is within 0X0 ~ 0XFFFF or 0x0 ~ 0xFFFF.If the figure is beyond the above range, it must indicate with long integer. The long integer is suffixed with "L" or "l". For example:

Decimal long integer constant 158L (decimal: 158) 358000L (decimal: -358000)

Octal long integer constant 012L (decimal: 10) 077L (decimal: 63) 0200000L (decimal: 65536)
Hexadecimal long integer constant 0X15L (decimal: 21) 0XA5L (decimal: 165) 0X10000L (decimal: 65536). There is no difference between long int 158L and basic int constant 158. As 158L is the long integer, C compiling

system will assign 4-bit space for storage. As 158 is the long integer, C compiling system will assign 2-bit space for storage. Therefore, pay attention to operation and output format to avoid mistakes. The unsigned number can be indicated with suffix. The unsigned number of integer constant is suffixed with "U" or "u". For example: 358u,
0x38Au, 235Lu are the unsigned number. Use prefix and suffix together to indicate different figures. For example,

0XA5Lu indicates the hexadecimal unsigned long int A5 and the corresponding decimal is 165.

3. Integer variable

The integer variable can be classified as following:

1. Int

Its type specifier is int, which occupies 2 bytes in the memory. Its value is always basic integer.

2. Short int

Its type specifier is short int or short'C110F1. The occupied bytes and value range is same to basic int.

3. Long int

Its type specifier is long int or long, which occupies 4 bytes in the memory. Its value is always long integer.

4. Unsigned int

Its type specifier is unsigned.

The unsigned int can be integrated with the above three types:

(1) The type specifier of unsigned int is unsigned int or unsigned. (2) The type specifier of unsigned short int is unsigned short.
(3) The type specifier of unsigned long int is unsigned long.

All unsigned int occupies the same memory bytes with the signed int. As the sign bit is omitted, it can not indicate the negative number. The following table lists the assigned memory bytes and number range of various integers in ARM.

Type Specifier	Number Range	Assigned Bytes
Int	-2147483648~2147483647	■■■■
Short int	-2147483648~2147483647	■■■■
Signed int	-2147483648~2147483647	■■■■
Unsigned int	0~4294967295	■■■■
Long int	-922337203685477808~922337203685477807	■■■■■■■■
Unsigned long	0~18446744073709551615	■■■■■■■■

4. Integer variable declaration

The general form of variable declaration: type specifier, variable name identifier, …. Take examples as following:

int a,b,c; (a,b,c is integer variable)

long x,y; (x,y is long integer variable)

unsigned p,q; (p,q is unsigned integer variable)

Announcements in writing variables should be paid attention as following:

1. Several same type of variables can be allowed to indicate after the same type specifier. Space the variable names with comma. There must be a space at least between the type specifier and variable name.

2. The last variable name must be ended with ";".

3. The variable declaration must be in front of variable usage. It is always at the head of function body.

[Practice] //1int a,b;

short int c; short d=100; a=d-20; b=a+d;

c=a+b+d;

d=d-a+c-b.

5. Float constant

Real constant is also called float constant. Real constant is also called float constant. In the C language, the float is indicated with decimal only. It has two forms as following: Decimal form and exponential form

1. Decimal form

It is composed with figure 0-9 and decimal point. For example: 0.0,.25,5.789,0.13,5.0,300.,-267.8230 are the legal float number.

2. Exponential form

It is composed of decimal digit, exponent symbol "e" or "E" and exponent (be integer only, symbol is possible). The basic form is an E n (a is decimal, n is decimal integer) and the value is a*10,n. For example, 2.1E5 (equal to 2.1*10,5), 3.7E-2 (equal to 3.7*10,)-2*) 0.5E7 (equal to 0.5*10,7), -2.8E-2 (equal to -2.8*10,)-2*). The following are the illegal float number: 345 (without decimal point) E7 (without figure before exponent symbol) -5 (without exponent symbol) 53.-E3 (negative sign is incorrect) 2.7E (without exponent)

The standard floating number in C language has suffix. The figure with suffix "f" or "F" is the floating number. For example, 356f and 356 is equivalent.

6. Float variable

The float variable includes single and double. Their type specifiers are float and double. In the Turbo C, the single occupies four bytes (32-bit) memory, and it is ranged between 3.4E-38 ~ 3.4E+38, which provides seven effective figures only. The double occupies 8 bytes (64-bit) memory, and the value is ranged between 1.7E-308 ~ 1.7E+308, which

provides sixteen effective figures.

The form and written rules for float variable declaration is same with that of integer. For example: float x,y; (x,y is single float)

double a,b,c; (a,b,c is double float)

The float constant is not classified into single and double. All float constants are processed as double. a ■■■■

b ■■■■■■■■

a<---33333.33333

b<---33333.33333333333;

[Practice] //float int a=32; float b;

double d; b=12345678; d=b*100;

d=d+a;

d=d+58.123456;

7. Character

Characters include character constant and character variable. Character constant

Character constant is a character within single quote. For example, 'a'. 'b', '=', '+', , '?' are the legal character constants. In the C language, the character constants are always characterized as following:

1. Character constant must be included in single quote rather than double quotes or other brackets.

2. Character constant must be single character rather than character string.

3. Character may be any character in the character set. However, the figure cannot be involved in numerical operation after been defined as character. For example, '5' and 5 is different. '5' is character constant that are not involved in operation.

8. Character variable

The character variable value is character constant, i.e. single character. Its type specifier is char. The form and written rules of character variable declaration is same with that of integer variable.

For example:

char a,b;

As each character variable is assigned to a byte memory, one character is saved only. The character value is kept in the memory unit with ASCII code. For example, the decimal ASCII code for x is 120, and the decimal ASCII code for y is
121. Give 'x' and 'y' to character variable a and b: a='x', b='y'. Actually, it is to store 120 and 121 BC in a and b unit:

a 0 1 1 1 1 0 0 0 b 0 1 1 1 1 0
0 1
Therefore, they can be regarded as integer. C language allows to give character value to integer variable and give integer to character variable as well. It can output character variable as integer and output integer as character. The integer is
2-byte, and character is single byte. When the integer is processed as character, the low eight bytes are involved only. [Practice] //charint a=49;
char b; char d; b=a+10; d=a+b;
[Practice] //char c1,c2;

c1='a';c2='b';

c1=c1-32;c2=c2-32;

9. Character string constant

Character string constant is a character string included in double quotes. For example: "CHINA". "C program", "$12.5", they are legal character string constants. The character string constant is different from character constant. Their differences are described as following:

1. Character constant is included in single quotes while character string constant is included in double quotes.

2. Character constant is single character only while character string constant contains one or several characters.

3. A character can be given to a character variable, but a character string constant can not. In the C language, there is no corresponding character string variable.

4. The character constant occupies one byte in the memory. The bytes of character string constant are equal to the bytes of character string plus 1. Save character "\0" (ASCII code:0) in the increased byte. This is the ending symbol of character string. For example, the byte of "C program" in memory is C program\0. Although the character constant 'a' and character string constant ''a'' has one character both, their memory occupation is different.

   'a' occupies one byte in the memory, which is indicated as a

   ''a'' occupies two bytes in the memory, which is indicated as a\0 symbol constant.

10. Symbol constant

In the C language, a constant can be expressed with an identifier, which is called symbol constant. It must be defined before usage. Its general form is:

#define symbol constant

Wherein, #define is a preprocessor directive, which is called macro definition directive. It is used to define the identifier to the constant value. Upon definition, all of this identifier in the future program will be replaced by the constant value. Usually, the identifier of symbol constant is expressed with capital letter and the variable identifier is expressed with lowercase letter for distraction.

```
#define PI       3.14159 void main(){
float s,r; r=5; s=PI*r*r;
printf("s=%f\n",s);

}
```

It is defined by macro definition directive. P1 is defined to be 3.14159, and s,r is defined to be float. 5->r PI*r*r->s

Display program result float s,r. wherein, r=5, s=PI*r*r. This program is defined by macro definition directive before main function. P1 is 3.14159, which substitutes for P1 in the program. s=PI*r*r is equivalent to s=3.14159*r*r. Pay attention: symbol constant is not variable. Its value can not be changed in the overall action scope. That is to say, assignment statement is forbidden to re-assign in the program.

7.1.2 Initial value of variable and type conversion

1. Initial value assignment for variable

In the program, it is usually to assign initial value for the variable. There are many methods for initial value assignment in the language program, which are called initialization. In the variable declaration, the general form of initial value assignment is:

Type specifier variable 1 = 1, variable 2 = 2, ……; for example:

int a=b=c=5;

float x=3.2,y=3f,z=0.75;

char ch1='K',ch2='P';

Note: no continuous assignment is forbidden in the declaration, for instance a=b=c=5 is illegal.

2. Type conversion of variable

The variable type is convertible. There are two methods for conversion. One is automatic conversion, the other is forced conversion.

Automatic conversion

When the different types of data are involved in hybrid operation, the automatic conversion is completed by compilation system. The auto conversion should conform to the following rules:

1. If the data types involved in operation are different, first convert them into one type, then make operation.

2. Conversion is made as the data length so as to ensure the high precision. For example, when int and long is in operation, convert int into long, then make operation.

3. All float operations are double. Even the expression contains float only, it must be converted to double for operation.

4. Char and short must be converted into int for operation.

5. In the assignment operation, when the data types on both sides of assignment sign are different, the right data type will be converted to the left type. If the right data is longer than left one, it will loss a part of data. In this case, the precision will be reduces. The lost data will be rounded off.

3. Forced type conversion

The forced type conversion is achieved by type conversion operation. Its general form is (type specifier) (expression). It is used to convert the operation results into the specified type of type specifier by force. Take an example of (float) a. convert a to float (int)(x+y), convert the result of x+y into integer. In forced conversion, there are some points to be noted:

1. Type specifier and expression must be included in the bracket (the single variable may not be bracketed). If (int)(x+y)

   is written to be (int)x+y, it means to convert x into int and plus y.

2. Either forced conversion or auto conversion is just the temporary conversion of data length for convenience of the operation. It will not change the variable type in the data declaration.

7.1.3 One-dimensional array

In the program design, the array organizes several variables with same category in ordered form for convenience. The set containing data elements with the same category in order is called array. In C language, the array belongs to construction data. One array can be split into several array elements. These array elements are either basic data or construction data. Therefore, the array can be classified into numerical array, character array, pointer array, structure array according to the category of array element.

This section will introduce the numerical array and character array, others will be described in the successive sections. If the array type declaration uses array in C language, it must be made type declaration first. The general form of array declaration is: type specifier array name [constant expression]. Wherein, type specifier refers to any basic data or construction data. Array name refers to the array identifier defined by users. The constant expression in square bracket indicates the quantity of data elements, which is also called array length.

For example:

int a[10]; int array a contains 10 elements.

float b[10],c[20]; float array b contains 10 elements; float array c contains 20 elements. char ch[20]; character array ch contains 20 elements.

For the array type declaration, there are several points as following:

1. The array type actually refers to the value type of array element. For the same array, the data type of all elements is the same.

2. The writing rules for array name should conform to that of identifier.

3. The array name should not be same with other variable name. For example: void main ()

{

int a;

float a[10];

……

} is incorrect

4. The constant expression in the square bracket refers to the element quantity. For example, a [5] indicates that array a contains 5 elements. However, its subscript is started from 0. Therefore, the five elements are a[0],a[1],a[2],a[3],a[4].

5. The square bracket can not include the element quantity of variables but symbol constant or constant expression is available. For instance:

```
#define FD      5
void main ()
{
int a[3+2],b[7+FD];
……
```

} is legal. However, the following expression form is incorrect. void main()

```
{
int n=5;
int a[n];
……
}
```

6. It is allowed that the same type declaration can describe several arrays and several variables.

For example: int a,b,c,d,k1[10],k2[20];

1. Representation of array element

Array elements are the basic unit of array. It is also a variable, which is identified with array name and a subscript. The subscript indicates the sequence number of element in the array. The general form of array element is: array name [subscript]. Wherein, subscript is integer constant or integer expression only. If it has decimal, this value will be integer automatically by C programming. For example, a[5],a[i+j],a[i++] are the legal array elements. Array element is usually called subscript variable. The subscript variable could not be used unless array is defined. In C language, the subscript variable is used one by one rather than the whole array.

The general form of initialization assignment is: static type specifier array name [constant expression]=[value……]. Wherein, static refers to the static storage type. It is specified that only static storage array and external storage array can be initialized assignment (the relevant static storage and external storage concepts will be introduced in Chapter 5) in C language. The data in { } are the initial value of each element, and the elements are spaced with comma such as static int a[10]={ 0,1,2,3,4,5,6,7,8,9 }, which is equivalent to a[0]=0;a[1]=1...a[9]=9;

2. There are several provisions for initial assignment of array in C language:

1. It is allowed to assign initial value for partial elements. When the elements in { } is less than element quantity, the initial value is assigned for the front only. For example: static int a[10]={0,1,2,3,4}, it indicates that the initial value will be assigned for the first 5 elements a[0] ~ a[4], and the last 5 elements will be assigned 0 automatically.

2. Assign initial value for element one by one, and the overall assignment for array is unavailable. For example, if assigning 1 for 10 elements, it can write to be static int a[10]={1,1,1,1,1,1,1,1,1,1} rather than static int a[10]=1.

3. If the initial value assignment is unavailable for all arrays with initialization, all elements will be 0.

4. If assigning all elements, the array element quantity may not be given in the array declaration. For example: static int a[5]={1,2,3,4,5} can be written as static int a[]={1,2,3,4,5}. The dynamic assignment can be made during program execution. In this case, it can use do statement and scanf function to assign

the array elements one by one.

3. **Character array**

The array for storing characters is called character array. The form of type declaration for character array is same to that of numerical array as previous introduction. For example: char c[10]. As the character and integer is similar, it can be defined as int c[10], but each array element occupies 2 bytes in memory. The character array may be two-dimensional array. For instance, char c[5][10] is a two-dimensional character array. The character array is allowed to made initialization assignment in type declaration. Take static char c[10]={`c`,` `,`p`,`r`,o`,g`,r`,`a`,`m`} as an example. After assignment, the element value is c[0]c[1]c[2]c[3]c[4]c [5]c[6]c[7]c[8]c[9] for array C. Wherein, c[9] is not assigned and assigned to 0 by system automatically. When assigning initial value for all elements, the length declaration can be omitted such as static char c[]={`c`,` `,`p`,`r`,`o`,`g`,`r`,`a`,`m`}, in which the length of C array is set to 9.

C language allows for initialization assignment for array in character string. For example, static char c[]={'c', ' ','p','r','o','g','r','a','m'} can be written to static char c[]={"C program"} or static char c[]="C program" without {}. The assignment in character string occupies one more byte than assignment one by one. It is used to store the ending sign of character string '\0'. The actual storage of array c in memory is C program\0. Wherein, `\0' is added by C programming system automatically. As '\0' sign is adopted, the array length is not normally defined in initialization assignment of character string but processed by system automatically. If in the mode of character string, the input and output of character array becomes simple and convenient. Except the initial value assignment with character string, it can input and output the character string of one character array with printf function and scanf function at one time not requiring input/output each character with do statement one by one.

7.1.4 Basic operator and expression

Category, priority and associativity of operator

There are many operators and expressions in C language, which is seldom in the high-level language. It is the rich operator and expression that complete the C language. This is one of main features of C language.

The operators of C language have different priorities. In addition, they have individual associativity. In the expression, the data for operation should not only conform to the priority of operators but also subjected to the associativity so as to confirm the operation direction from left to right or from right to left. This associativity is unavailable for other high-level language, which increases the complexity of C language.

Operator Category

The operators of C language can be classified as following:

1. **Arithmetic operator**

   It is used for different data operations, including addition (+), subtraction (-), multiplication (*) and division(/) (or modular arithmetic, %), increment (++) and decrement (--).

2. **Relational operator**

   It is used for comparison operation, including greater than (>), less than (<), equal to (==), be equal or greater than (>=), be equal or less than (<=) and unequal to (! =).

3. **Logical operator**

   It is used for logical operation, including And (&&), Or (||) and Not (!).

4. **Bit operation operator**

   The data for operation is taken as binary bit, including bit and (&), bit or (|), bit not (~), bit or (^), left shift (<<) and right shift (>>).

5. **Assignment operator**

   It is used for assignment operation, including simple assignment (=), composite arithmetic assignment (+=,-=,*=, /=, %=)
   and composite bit operation assignment (&=,|=, ^=,>>=,<<=).

6. **Conditional operator**

   This is a ternary operator for conditional evaluation (?:).

7. **Comma operator**

   It is used for combining several expressions to one expression (,).

8. **Pointer operator**

   It is used for two operations as content-of (*) and address-of (&).

9. Size-of operator

It is used for size-of operation of data.

10. Special operator

It includes bracket (), subscript [] etc.

7.1.5 Section summary

1. C data type

Basic type, construction type, pointer type and void type

2. Classification and characteristics of basic type

Type Specifier	Byte	Number Range
char	1	C character set
int	4	-214783648 ~ 214783647
short int	4	-214783648 ~ 214783647
long int	8	-922337203685477808 ~ 922337203685477807
unsigned	4	0 ~ 4294967295
unsigned long	8	0 ~ 1844744073709551615
float	4	3/4E-38 ~ 3/4E+38
double	8	1/7E-308 ~ 1/7E+308

3. Constant suffix

L or l for long int

U or u for unsigned
F or f for float

4. Constant type

Int, long int, unsigned, float, char, char string, symbol constant, and escape character

5. Data type conversion

Auto conversion

The system realizes auto conversion for the hybrid operation of different types of data, which converts from small byte data to big byte data. For the mutual assignment of different data, the system also converts automatically, which converts the right data type into left one.

Forced conversion

It is converted by forced conversion operator.

6. Priority and associativity of operator

Generally speaking, the unary operator has higher priority and the assignment operator has lower priority. The arithmetic operator has higher priority, and the relational and logical has lower priority. Most operators have left associativity, unary operator, ternary operator and assignment.

7. Expression

Expression is the formula composed with connection constant, variable and function of operator. Each expression has one value and type. The evaluation of expression is made according to the sequence specified by priority and associativity of operator.

8. Array

1. Array is the commonest data structure in program design. The array contains numerical array (int array, float array), character array and pointer array, structure array to be described later.

2. Array may be one-dimensional, two-dimensional or multi-dimensional.

3. The type declaration of array consists of type specifier, array name and array length (elements quantity of array). The array element is also called subscript variable. The array type refers to the value type of subscript variable.

4. Make array assignment with three methods: initialization assignment, dynamic assignment by inputting function and assignment statement. The numerical array can not be overall assigned, input or output with assignment statement but assigned for array element one by one with do statement.

7.2 C language programming preliminary

7.2.1 Statement of C program

The execution part of C program is constituted by statements, and the program function is also realized by execution statement. C statement is classified into five categories:

1. Expression statement
2. Control statement
3. Compound statement
4. Null statement

1. Expression statement

Expression statement consists of expression and semicolon. Its general form is expression;. Execution of expression statement is to compute the expression value. For example, x=y+z; assignment statement y+z; operate statement with addition, but the result is not kept. It has no actual significance i++. Increment 1 statement, i value increases 1.

2. Control statement

Control statement is to control the program process so as to realize various structures of program.

It is composed of special statement delimiter. There are nine control statements in C language, which can be classified into three kinds:

(1) Conditional judgment statement if statement, switch statement
(2) Looping execution statement

do while statement, while statement, for statement

(3) Go to statement

break statement, go to statement, continue statement, return statement

3. Null statement

The statement with semicolon only is called null statement. Null statement executes nothing. In the program, null statement can be the null loop body. Take an example of while (getchar()!='\n'). For this statement, if the character input from keyboard is not Enter, it requires re-input. Here, the loop body is null statement.

4. Assignment statement

Assignment statement consists of assignment expression and semicolon. Its general form is variable = expression. Its functions and features are same to that of assignment expression. It is one of the most popular statements in the program. There are some points to be noted in the usage of assignment statement:

1. As the expression on the right of assignment sign "=" can be an assignment expression, the following form Variable=(variable=expression); is established, then the nestification is formed. Its expanded expression is Variable=Variable=…=Expression;

   For example:

   a=b=c=d=e=5; according to the right associativity of assignment operator, it is equivalent to: e=5 actually;

   d=e; c=d; b=c; a=b;
2. Pay attention to the difference between assigning initial value and statement for variable in the variable declaration.

   Assigning initial value to variable is a part of variable declaration. The variable with initial value assignment should be spaced with comma to other similar variable, but the assignment statement must be ended with semicolon.

3. In the variable declaration, it is forbidden to assign initial value for several variables successively. For example, the following declaration is incorrect. Int

a=b=c=5 must be written to int a=5,b=5,c=5. However, the assignment statement must be assigned continuously.

4. Note the difference between assignment expression and assignment statement. Assignment expression is a kind of expression, which can be used in any allowable place. But the assignment statement can not.

   The following statement is legal: if((x=y+5)>0) z=x; the function of statement: if expression x=y+5 is greater than 0, then z=0.

   The following statement is illegal: if((x=y+5;)>0) z=x; as x=y+5; is a statement, it can not be used in expression.

7.2.2 Branch structure program

Relational operator and expression

In the program, it usually compares the size of two data so as to confirm the next process. The operator for comparing data size is called the relational operator. There are such relational operators in C language as following:

< less than

<= less than or equal to

> greater than

>= greater than or equal to

== equal to

!= unequal to

The relational operator is binary operator, which is left associative. Its priority is lower than that of arithmetic operator and higher than that of assignment operator. In the six relational operators, <,<=,>,>= has the same priority, which is higher than == and !=. While == and != has the same priority.

Relational expression

The general form of relational expression is Expression Relational operator Expression. For example, a+b>c-d,x>3/2,'a'+1<c,-i-5*j==k+1 are legal relational expression. As the expression is relational expression concurrently, the nestification may occur such as a>(b>c),a!=(c==d) etc. The value of relational expression is "true" and "false", which is expressed with "1" and "0".

1. Logical operator and expression

In C language, the logical operators include AND operator &&, OR operator || and NOT operator !. AND operator && and OR operator || are binary operators with left associativity. NOT operator ! is unary operator with right associativity. The priority relation between logical operator and other operators can be expressed as following:

The following can be derived depending on the priority of operator:
a>b && c>d is equivalent to (a>b) && (c>d)

!b==c||d<a is equivalent to ((!b)==c)||(d<a)

a+b>c && x+y<b is equivalent to ((a+b)>c) && ((x+y)<b) Evaluation of logical operation

The evaluation of logical operation may be true or false expressed with 1 and 0 individually. Its evaluation rules are as following:

1. When the two values of AND operation && are true, the results are true; otherwise they are false. For example, 5>0 && 4>2. As 5>0 is true and 4>2 is true, the corresponding result is true.
2. When one of two values involved in OR operation || is true, the result is true. When two values are false, the result is false. Take an example of 5>0||5>8. As 5>0 is true, the corresponding result is true.
3. When the NOT operation! involved in operation is true, the result is false; when the involved operation is false, the result is true.

For example, the result of !(5>0) is false.

In the logical operation value of C programming, it represents “true” with “1” and represents “false” with “0”. Vice versa, when judging a value is true or false, 0 represents false and the non-zero data represents true. For example, as 5 and 3 are non zero, the value of 5&&3 is “true” (i.e. 1).

Another example: the value of 5||0 is “true” (i.e. 1).

The general form of logical expression is Expression – Logical operator – Expression. Wherein, the expression can be logical expression as well, which forms nestification? Take the (a&&b) &&c as an instance. The above expression can be written to a&&b&&c according to the left associativity of logical operator. The value of logical expression is the final value of various logical operation, which represents “true” and “false” with “1” and “0” respectively.

2. if statement

The branch structure can be constituted with if statement. It makes judgment according to the given conditions so as to confirm what branch program period is to be executed. If statement of C language has three basic forms.

1. The first form is: basic form. if (expression) statement

It's semanteme: if the expression value is true, the following statement will be executed, otherwise not.

2. The second form is if-else. if(expression)

statement 1; else statement 2;

Semanteme: if the expression value is true, it will execute statement 1; otherwise statement 2.
Input two integers, and output the bigger one. Judge a and b size with if-else statement. If a is bigger, it outputs a; otherwise b.

3. The third form is if-else-if form.

In the first two forms, if statement is normally used for two branches. When there are several branches for selection, if-else-if statement is adopted. Its basic form is:

if(expression 1)

statement 1;

else if(expression 2)

statement 2;

else if(expression 3)

statement 3;

…

else if(expression m)

statement m; else statement n;

Semanteme: judge the expression value in sequence. When a value is true, it executes the corresponding statement. Then it executes program out of if statement. If all expressions are false, it will execute statement n. Then continue to execute the subsequent program.

There are some points to be noted in if statement:

(1) In the three forms of if statement, the one behind if is expression. This expression is usually the logical expression or relational expression. But it may be other expressions such as assignment expression even a variable. For example, if(a=5) statement and if(b) statement are allowable. As long as the expression value is not 0, it is true. If the expression value in if(a=5)…; expression is always not 0, the subsequent statement will be executed. This kind of situation may not take place in the program, but the syntax is legal.

Another example, program segment: if(a=b)

printf("%d",a); else printf("a=0");

Semanteme of the statement: assign b to a. if it is not 0, this value is output; otherwise it outputs "a=0" character string. This kind of application usually occurs in the program.

(2) In if statement, the conditional judgment expression must be included in bracket, and ended with semicolon.
(3) In the three forms of if statement, all statements should be single statement. If a group (several) statements are required execution with conditions, this group of statements must be bracketed with {} to form a compound statement. Pay attention that no semicolon is allowed behind }.

For example:

if(a>b){ a++; b++;

}

else{ a=0;

b=10;

}

4. Conditional operator and conditional expression

If the single assignment statement is executed only in the conditional statement, it is usually realized by conditional expression, which not only simplifies the process but also improves the operation efficiency.

Conditional operator ? and : is a ternary operator, which means three values are involved in operation. The general form of conditional expression composed by conditional operators is:

Expression 1 ? Expression 2: Expression 3

Its evaluation rule: if the expression 1 is true, its value of expression 2 will be the value of conditional expression; otherwise, the value of expression 2 will be the value of whole conditional expression. Conditional expression is normally applied in assignment statement. For example:

if(a>b) max=a;

else max=b;

max=(a>b)?a:b; is expressed with conditional expression. Its semanteme is: if a>b is true, assign a to max; otherwise assign b to max.

In the application of conditional expression, there are some points to be noted as following:

1. The operation priority of conditional operator is lower than that of relational operator and arithmetic operator but higher than assignment operator. Therefore, max=(a>b)?a:b can be removed the bracket to be max=a>b?a:b.
2. Conditional operator ? and : is a pair of operator, which can not be separated in application.
3. The associative direction of conditional operator is from right to left.

7.2.3 switch statement

C language provides another switch statement for selection of multiple branches. Its general form is:

```
switch(expression){
case constant expression 1: statement 1;
case constant expression 2: statement 2;
…
case constant expression n: statement n;
default: statement n+1;
}
```

Semanteme: calculate the expression value, and compare with the subsequent constant expression value one by one. When the expression value is equal to a constant expression value, the subsequent statement is executed. Then judgment is not made. Continue the statement behind all case. If the expression value is different from the constant expression behind case, it will execute the statement behind default.

There are several points to be noted in switch statement:

1. All constant expression values behind case must be different, otherwise there will be mistake.
2. Several statements are allowed behind case, and they can not be bracketed with {}.
3. The sequence of case and default clauses may be changed and will not affect the program execution.
4. Default clause may be omitted

7.2.4 Loop structure program

The loop structure is an important structure of program. When the given condition is satisfied, one program segment is executed repeatedly until the condition is unsatisfied. The given condition is called loop condition, and the program segment executed repeatedly is called loop body. C language provides many loop statements, which may compose different loop structures.

1. While statement

The general form of while statement: While (expression) statement;

wherein, the expression is loop condition, and the statement is loop body.

Semanteme of while statement: compute the expression value. When the value is true (not 0), the loop body statement is executed.

There are some points to be noted in while statement:

1. The expression of while statement is usually the relational expression or logical expression. As long as the expression value is true (not 0), it can continue loop.
2. If the loop body contains one or more statements, it must be bracketed with {} to form the compound statement.

3. Note the loop conditions to avoid endless loop.

2. Do-while statement

General form of do-while statement:

do statement;

while (expression);

Wherein, the statement is loop body, and the expression is the loop condition. Semanteme of do-while statement:

First execute the loop body statement for one time, then judge the expression value. If the value is true (not 0), the loop is continuous; otherwise the loop ends.

The difference between do-while statement and while statement is that do-while executes first and judges late. Therefore, do-while will execute the loop body for one time at least. But while statement judges first and executes late. If the condition is unsatisfied, the loop body statement is not executed for one time.

while statement and do-while statement is usually mutual re-write.

In this example, the loop condition is rewritten to be –n. Otherwise, one more loop will be executed. There are some points to be noted in do-while

statement:

1. In the if statement and while statement, no semicolon is added behind the expression; while the expression of do-while statement must be ended with semicolon.
2. do-while statement may be composed to the nested loop and nested with while statement mutually.
3. The loop body between do and while is made up of several statements, and bracketed with {} to form a compound statement.
4. When converting do-while and while statement mutually, pay attention to modify the loop control conditions.

7.2.5 for statement

For statement is a kind of loop statement with stronger function and wider application provided by C language. Its general form is:

For (Expression 1;Expression 2;Expression 3)

statement;

Expression1: it is usually to assign initial value to loop variable, and it is assignment expression. It also allows to assign initial value to loop variable except for statement. In this case, the expression may be omitted.

Expression 2: it is usually the loop condition, and it is relational expression or logical expression. Expression 3: it is usually for modifying the

value of loop variable, and it is assignment statement.

These three expressions may be comma expression. That is to say, each expression can be composed with several expressions. Three expressions are options and can be omitted.

The “statement” in general form is loop body statement. Semanteme of for statement is:

1. First, calculate the value of expression 1.
2. Then, compute the value of expression 2. If the value is true (not 0), loop body is executed once more; otherwise exit the loop.
3. Calculate the value of expression 3 and return to execute step 2 again. During the for process, expression 1 is calculated for one time, and expression 2 and 3 may repeat for several times. The loop body may be executed for many times or not executed.

There are several points to be noted in for statement:

1. Each expression in for statement can be omitted, but the semicolon must exist. For example:

   <1>for(expression; expression) expression is omitted

   <2>for(expression; expression;)expression is omitted

   <3>for(;expression; expression) all expression is omitted

2. When the loop variable has assigned initial value, Expression 1 may be omitted as shown in Example 3.27. If Expression 2 or 3 is omitted, the endless loop may be caused. In this case, the loop should be ended in loop body.

3. The loop body may be void statement.

   #include" stdio.h" void main(){

```
int n=0;
printf("input a string:\n"); for(;getchar()!='\n';n++); printf("%d",n);
}
```

In this example, the expression 1 in for statement is omitted, and the expression 3 is not for modifying loop variable but for inputting the characters counting. Thus, the counting that should be completed in loop body has completed in the expression. Therefore, the loop body is void statement. Pay attention, the semicolon behind void statement is essential. If this semicolon is missed, the following printf statement will be executed as loop body. On the other hand, if the loop body is not void statement, it is forbidden to add semicolon behind the bracket of expression. In this case, the loop body will be regarded as void statement and not executed repeatedly. All of these are the common mistakes in programming, which must be attached great importance.

7.2.6 break statement

Break statement is used in switch statement or loop statement only. It is for exiting switch statement or local loop and directing to the subsequent program. As the transferring direction of break statement is specific, the statement marks are not required. The general form of break statement is break. As shown in the above example, the break statement is used in switch statement and for statement for skip. Break statement provides several exits for loop statement, which makes programming more flexible and convenient in some circumstances.

7.2.7 continue statement

Continue statement is used in loop body only. Its general form is continue;

Semanteme: finish this loop and not execute the other statement behind continue statement in the loop body any more, turn to judge and execute the next loop condition. Pay attention: this statement only ends the loop on this layer and will not exit the loop.

7.2.8 Section summary

1. From the execution process, the program is basically classified into three basic structures: sequence structure, branch structure and loop structure.
2. The most basic unit in program execution is statement. There are five kinds of statements in C language:
   (1) Expression statement any expression and semicolon forms the expression statement. The general expression statement is assignment statement.
   (2) Function call statement the function call and semicolon constitutes the function call statement.
   (3) Control statement it is used for control program process, and composed of special statement delimiter and required expression. It mainly includes conditional judgment execution statement, loop execution statement, go to statement etc.
   (4) Compound statement it is composed by several statements included in {}. Compound statement is regarded as single statement. It can be used in any place allowing statement such as loop body.
   (5) Void statement it is composed by semicolon only without actual function.
3. Relational expression and logical expression are two important expressions, which are mainly used for judgment of conditional execution and loop execution.
4. C language provides many forms of conditional statement to form the branch structure. (1) if statement is mainly for one-way selection.
   (2) if-else statement is mainly for two-way selection.
   (3) if-else-if statement and switch statement are for multiway selection. These forms of conditional statement are normally mutual substituted.
5. C language provides three loop statements.
   (1) For statement is mainly used to assign initial value for loop variable, step increment and loop structure of loop times. (2) The loop times and control condition can be confirmed during loop process, and the confirmed loop may use while or do-while statement.
   (3) Three loop statements can be nested mutually to form the nested loop. The loops may be in parallel but not crossed. (4) Branch statement can transfer the process out of loop body, but it can not transfer process from outside to loop body.

(5) Avoid endless loop in the loop program. That is to say, the loop variable must be ensured to be modified during operation. Change the loop condition to be false gradually, thus finish the loop.

6. Statement summary in C language

Name	General Form
Simple statement	Expression statement; Void statement ;
Conditional statement	if(expression)statement;
	if(expression)statement 1; else statement 2;
	if(expression 1)statement 1; else if(expression 2) statement 2…else statement n; switch statement
	switch(expression){ case constant statement: statement…default: statement; } loop statement while statement
	while(expression)statement;
	for statement for(expression 1; expression 2; expression 3) statement; break statement break;
	continue statement continue;
	return statement return(expression);

7.3 Macro function introduction

- Use macro function to program
- Function button introduction in macro editor
- New create macro
- Insert library function in the program
- Use internal storage area in macro

7.3.1 Use macro function to program

This section will introduce the basic functions of macro and describe the relevant control functions and application method briefly.

There are ways to access macro:

Method 1: select “Setting (S) -> Macro" from the menu as shown in Figure 5-1 (macro menu), and pop up interface as shown in Figure 5-1(macro editor).

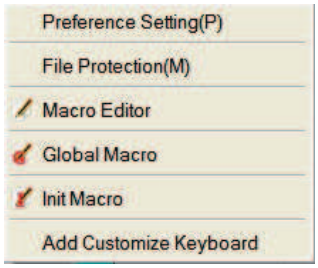


Figure 5-1 (Macro Menu)

Method 2: click shortcut button “Macro” as shown in Figure 5-2 (Macro button), and pop up interface as shown in Figure 5-3 (Macro Editor).

Figure 5-2 (Macro Button)

7.3.2 Function button introduction in macro editor

The functional controls are arranged in macro editor window as shown in Figure 5-3 (Macro Editor).

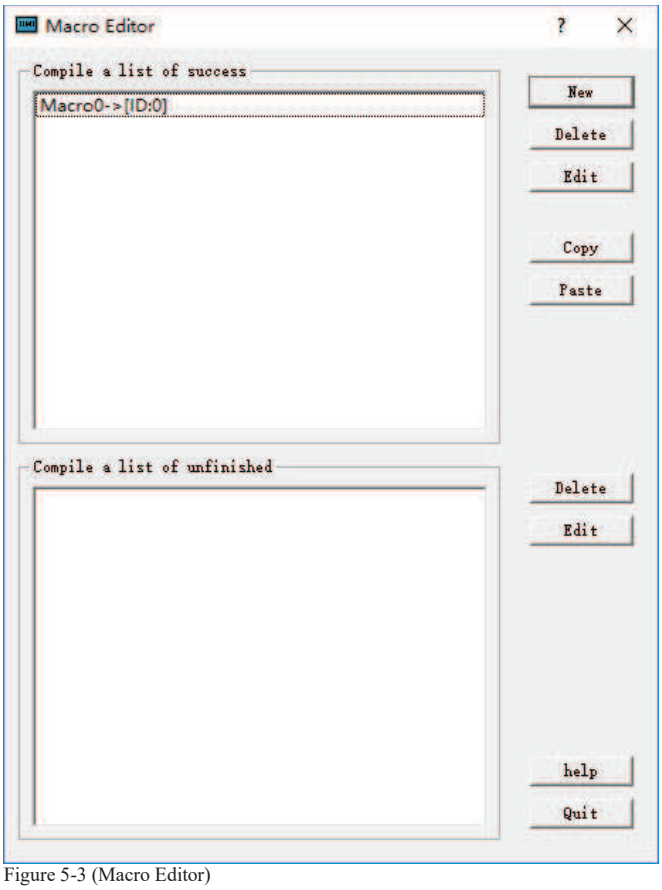


Figure 5-3 (Macro Editor)

<1> Button function description:

1. **New:** new create a macro.
2. **Delete:** delete the selected macro.

3. **Edit:** edit the selected macro (same with double click macro)
4. **Copy:** select one macro and click “copy” to copy the selected macro.
5. **Paste:** paste the copied contents, and the suffix of macro name adds one automatically.
6. **Help:** pop up the macro description.
7. **Quit:** close or exit the current dialog box.

<2> “Compile success” window: for recording all macro names with successful compilation in project as shown in (5-3). Select one and double click to enter into macro edition.

<3>“Unfinished compile” window: for recording all unfinished compile macro name in project as shown in (5-3). Select one and double click to enter into macro edition.

7.3.3 New create macro

Click “new add” button and pop up the dialog box of macro compiler as shown in Figure 5-4 (Macro Compiler)

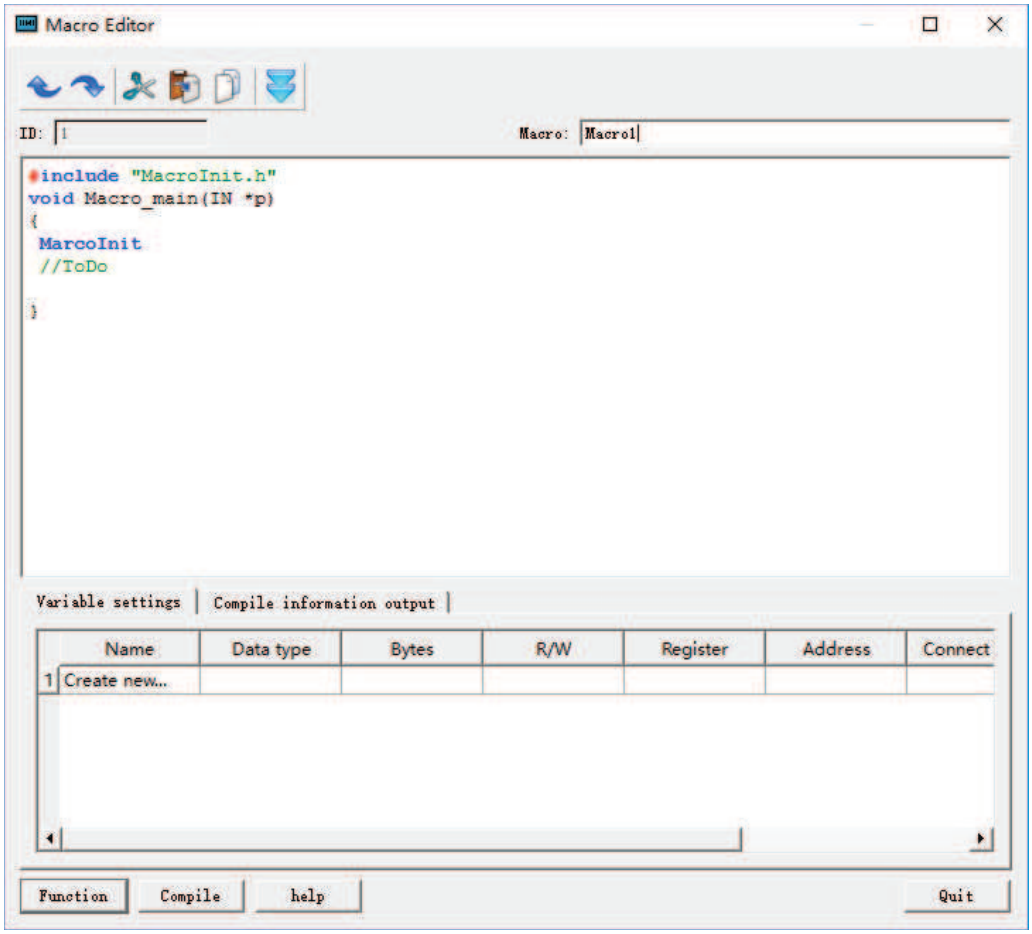


Figure 5-4 (Macro Compiler)

Button Function Description

Compile: Compile the current macro source code, and check the syntax error.

Function: Insert the selected library function at the cursor location in the edition window of source code.

Close: Close the current dialog box.

Description: pop up the help instruction of macro usage.

Instructions:

Instruction 1: convention introduction of program compilation

Create the source program of C language conforming to ANSI C standard in the edition window.

Instruction 2: information output window

The window outputs the information of compilation and connection state of program. Users may make debugging and modification according to the information prompt.

Instruction 3: close/open window

In the variable setting, users can right click the edition window of source code, and it will pop up the shortcut as Figure 5-5 (window close/open). Click “close/display variable setting window” and “close/display information output window” to close/open the corresponding window. Users can adjust the edition box size as required.

Instruction 4: variable setting

Variable name: input the variable name required in the program.

Data class: select the corresponding data class of variable name so as to distribute the corresponding memory size. Word length: set the occupied memory of corresponding variable automatically depending on the data class. Read/write: set the read-write property of variable in the memory cell of touch screen.

Address: the defined physical memory address of variable in the touch screen.

Variable setting instruction: when the external requires interaction with touch screen such as data exchange and data acquisition, it provides the interface to change the behavior characteristics of touch screen dynamically.

Special tips: the variables required in the program should be put in the variable definition part (the variable definition starts from code).

7.3.4 Insert library function in the program

Click "function” button and pop up the dialog box as Figure 5-5. Select the corresponding library function in the function name list box, click “confirm” button and the function will be inserted to the cursor location of edition window automatically.

The functions, parameter types and other detailed instruction of library function refer to Appendix 1.

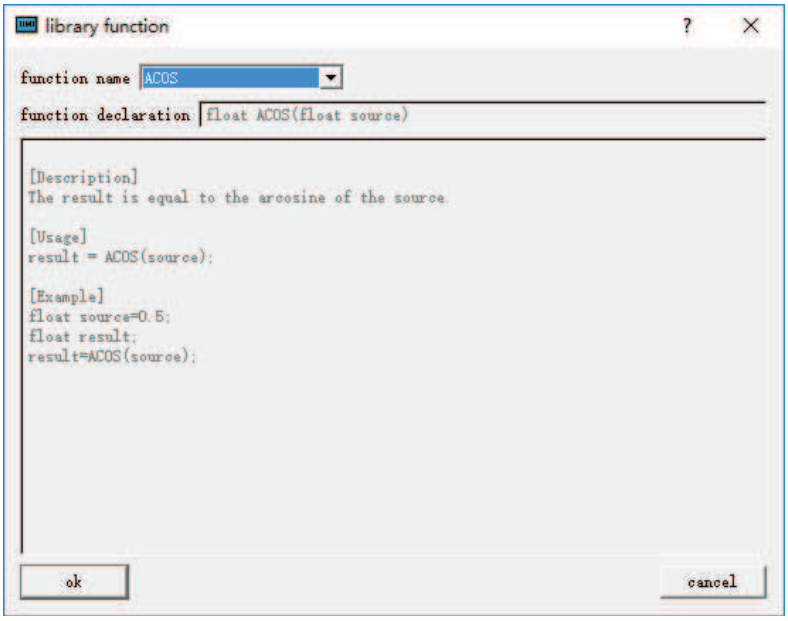


Figure 5-5 (Function Selection)

7.3.5 Use internal storage area in macro

HMITOOL configuration software provides the operation interface for HMI internal storage area. Users can make direct operations for the internal storage area in the macro. Detailed methods have two kinds:

1. Access storage area with keywords

LocalBit: reference of internal storage area LB; LocalWord: reference of internal storage area LW;

RWIWord: reference of internal storage area RWI.

The keyword can be used directly in macro. For example:

```
if(LocalBit[5])
{
LocalWord[1]=0;
}
else
{
LocalWord[1]=1;
}
```

2. Establish a linkage between variable and internal storage area by setting of variable. Detailed usage refers to macro instance.

7.4 String handling functions

Function name	function
StringCat	Joint two character strings
StringCompare	Compare the values of two character strings; case sensitive
StringCompareNoCase	Compare the values of two character strings; case insensitive
StringCopy	String copy
StringFind	Return to the position of the first occurrence of target string in the source string; if there is no found in the source string, return to -1.
StringFindOneOf	Return to the position of the first character in the source string that matches any character in target string; if there is no found in the source string, return to -1.
StringIncluding	Retrieve a substring of the source string that contains characters in the set string.
StringInsert	Insert a string into a specific position within the destination string content.
StringLength	Obtain the length of a string.
StringMid	Retrieve some characters sequence from the specified offset of the source string.
StringSplit	Split out the string.
StringToLower	Convert the characters of a string to lowercase.
StringToUpper	Convert the characters of a string to uppercase.
StringTrim	Split the source2 out from source1 to form a new one with the rest.
StringTrimLeft	Split out any character of source2 in source1; Return the result to the dest.

1、StringCat

[Description]

Append source string to destination string.

[Usage]

result=StringCat(dest[start],source);

[Example]

char a[8]="abcd";

char *b="efg";

char *c;

c=StringCat(a,b); // c="abcdefg"，it is better if a is a character array.

2、StringCompare

[Description]

Do a case-sensitive comparison of two strings.

[Usage]

result = StringCompare(source1,source2);

[Example]

int result;

char *a="abcd";

char *b="efg";

result = StringCompare(a,b); //result=-1(if a>b, result=1;if a=b, result=0; if a<b then result=-1)。

3、StringCompareNoCase //

[Description]

Do a case-insensitive comparison of two strings.

[Usage]

result = StringCompareNoCase(source1,source2)

[Example]

int result;

char *a="abcd";

char *b="EFG";

result=StringCompare(a,b);//result=1;

result=StringCompareNoCase(a,b);//result=-1;

4、StringCopy

[Description]

Copy ons string to another.

[Usage]

result = StringCopy(dest,source);

[Example]

char a[4]="abcd";

char* b="efg";

char* c;

c=StringCopy(a,b);//c="abcdefg"; it is better if a is a character array.

5、StringFind

[Description]

Return the position of the first occurrence of target string in the source string.

[Usage]

index=StringFind(source1,source2);

[Example]

char a[8]="ab1c123d";

char* b="123";

char* c="ef";

int index;

index=StringFind(a,b); //index=4;

index=StringFind(a,c); //index=-1;

6、StringFindOneOf

[Description]

Return the position of the first character in the source string that matches any character in target string.

[Usage]

index=StringFindOneOf(source1,source2);

[Example]

char a[8]="ab1c123d";

cahr* b="1b";

in index;

index=StringFindOneOf(a,b); //index=2;

7、StringIncluding

[Description]

Retrieve a substring of the source string that contains characters in the set string.

[Usage]

StringIncluding(source1,source2,dest);

[Example]

char a[8]="aB1Eree";

char* b="aBe";

char dest[8];

StringIncluding(a,b,dest);//dest="aBee"; it is better if dest is a character array.

8、StringInsert

[Description]

Insert a string in a specific within the destination string content. Notice it that the destination string has sufficient buffer.

[Usage]

StringInsert(pos,source,dest);

[Example]

char a[8] = "aB1e";

char *b ="kdr";

int pos=2;

StringInsert(pos,b,a); //a="aBkdr1e".

9、StringLength

[Description]

Obtain the length of a string.

[Usage]

result = StringLength(source);

[Example]

char a[8]="ABerer";

int length;

length = StringLength(a); //length=6;

10、StringMid

[Description]

Retrieve some characters sequence from the specified offset of the source string and store it in the destination buffer.

[Usage]

StringMid(source,count,dest);

[Example]

char a[8]="aB1e";

char b[3];

StringMid(a,2,b); //b="B1" it is better if b is a character array and it has enough memory to save the fetched character.

11、StringSplit

[Description]

Split out the string.

[Usage]

StringSplit(dest1,dest2,source,pos);

[Example]

char a[8]="aB1edge";

char b[5];

char c[5];

StringSplit(b,c,a,3); // it is better if b and c are character arrays.

12、StringToLower

[Description]

Convert the characters of a string to lowercase.

[Usage]

StringToLower(source,dest); // it is better if dest is a character array.

[Example]

char a[8]="ABeRe";

char b[8];

StringToLower(a,b);//b="abere".

13、StringToUpper

[Description]

Convert the characters of a string to uppercase.Notice if there is sufficient buffer.

[Usage]

StringToUpper(source,dest);

[Example]

char a[8]="ab1ere";

char b[8]; // it is better if b is a character array.

StringToUpper(a,b); //b="AB1ERE" ;

14、StringTrim

[Description]

split the source2 out from source1,return the result to dest.

[Usage]

StringTrim(source1.source2,dest);

[Example]

char a[12]="a1erd1esw";

char *b="1e";

char c[10];

StringTrim(a,b,c) //c="ardsw" . It is better if c is a character array.

15、StringTrimLeft

[Description]

split out any character of source2 in source1,Return the result to the dest.

[Usage]

StringTrimLeft(source1,source2,dest);

[Example]

char a[8]="aB1edge";

char* b="be";

char c[8];

StringTrimLeft(a,b,c);//c="aB1dg", It is better if c is a character array.

7.5 Data operation function

Function name	function
GETBIT	Get bit value.
HIByte	Retrieve the high byte from the low word of a specified value.
HIWord	Retrieve the high word from the specified value.
INVBIT	Set specific bit to be inversed (ON->OFF, OFF->ON).
LOByte	Retrieve the low byte from the specified value.
LOWord	Retrieve the low word from the specified value.
SWAPB	Swap the low byte and high byte of the specified value.
SWAPW	Swap the low word and high word of the specified value.
SETBIT	Set specific bit to be ON or OFF.

1、GETBIT

[Description]

Get bit value.

[Usage]

result = GETBIT(source, bit_pos) ;

[Example]

short source = 0x5, bit_pos = 0 ,result ;

result = GETBIT(source,bit_pos) ; // result == 1

2、HIByte

[Description]

Retrieve the high byte from the low word of a specified value.

[Usage]

result=HIByte(source) ;

[Example]

short source = 0x1234,result ;

result=HIByte(source) ; // result = 0x12

3、 HIWord

[Description]

Retrieve the high word from the specified value.

[Usage]

result=HIWord(source) ;

[Example]

int source = 0x45232568,result ;

result=HIWord(source) ; // result = 0x4523

4、INVBIT

[Description]

Set specific bit to be inversed (ON->OFF, OFF->ON).

[Usage]

result = INVBIT(source, bit_pos) ;

[Example]

short source = 0x6, bit_pos =1 ,result ;

result=INVBIT(source, bit_pos) ; // result = 4

5、LOByte

[**Description]**

Retrieve the low byte from the specified value.

[Usage]

result=LOByte(source) ;

[Example]

short source = 0x1234,result ;

result= LOByte(source) ; // result = 0x34

6、 LOWord

[Description]

Retrieve the low word from the specified value.

[Usage]

result=LOWord(source) ;

[Example]

int source = 0x12345678,result ;

result=LOWord(source) ; // result == 0x5678

7、SWAPB

[Description]

Swap the low byte and high byte of the specified value.

[Usage]

result=SWAPB(source) ;

[Example]

short source = 0x1234,result ;

result=SWAPB(source) ; // result = 0x3412

8、SWAPW

[Description]

Swap the low word and high word of the specified value.

[Usage]

result=SWAPW(source) ;

[Example]

int source = 0x12345678, ;

int result ;

result=SWAPW(source) ; // result = 0x56781234

9、SETBIT

[Description]

Set specific bit to be ON or OFF.

[Usage]

result = SETBIT(source,bit_pos,1) ;

[Example]

short source = 0x4 ;

short bit_pos = 1, result ;

result = SETBIT(source,bit_pos,1) ; // result = 0x6

7.6 Data switch function

Function name	function
ASCII2DEC	Convert a string to a decimal value.
ASCII2FLOAT	Convert a string to a floating value.
ASCII2HEX	Convert a string to a hexadecimal value.
BCD2BIN	Convert a BCD value to a BIN value.
BIN2BCD	Convert a binary value to a BCD value.
DEC2ASCII	Convert a decimal value to a string.
FLOAT2ASCII	Convert a floating value to a string.
HEX2ASCII	Convert a hexadecimal value to a string.

1、ASCII2DEC

[Description]

Convert a string to a decimal value.

[Usage]

result=ASCII2DEC(source , sizeof(source)) ;

[Example]

char source[4] = {'2', '3', '4', '5'} ;

short result ;

result=ASCII2DEC(source , 4) ; // result=2345

2、ASCII2FLOAT

[Description]

Convert a string to a floating value.

[Usage]

result=ASCII2FLOAT(source,sizeof(source)) ;

[Example]

char source[4] = {'5', '.', '7', '8'} ;

float result ;

result=ASCII2FLOAT(source,4) ; // result = 5.78

3、ASCII2HEX

[Description]

Convert a string to a hexadecimal value.

[Usage]

result=ASCII2HEX(source,sizeof(source)) ;

[Example]

char source[5] = {'1','6','3','4'} ;

short result ;

result=ASCII2HEX(source,4) ; // result = 0x1634

4、BCD2BIN

[Description]

Convert a BCD value to a BIN value.

[Usage]

result=BCD2BIN(source) ;

[Example]

short source = 0x2648 ;

short result ;

result=BCD2BIN(source) ; // result = 2648

5、BIN2BCD

[Description]

Convert a binary value to a BCD value.

[Usage]

result=BIN2BCD(source) ;

[Example]

short source = 1234 ;

short result ;

result=BIN2BCD(source) ; // result = 0x1234

6、DEC2ASCII

[Description]

Convert a decimal value to a string.

[Usage]

result= DEC2ASCII(source,sizeof(source)) ;

[Example]

short source = 5678 ;

```
char *result ;

result= DEC2ASCII(source,4) ;

//   result[0] == '5', result[1] == '6', result[2] == '7', result[3] == '8'
```

7、FLOAT2ASCII

[Description]

Convert a floating value to a string.

[Usage]

```
result=FLOAT2ASCII(source);
```

[Example]

```
float source = 56.8;

char result[4];

result=FLOAT2ASCII(source);

//result[0] = '5',result[1] = '6',result[2] = '.',result[3] = '8'
```

8、HEX2ASCII

[Description]

Convert a hexadecimal value to a string.

[Usage]

```
HEX2ASCII(source, result[start]) ;
```

[Example]

```
short source = 0x5678 ;

char *result ;

result = HEX2ASCII(source) ;

//result[0] = '5', result[1] = '6', result[2] = '7', result[3] = '8'
```

7.7 Math arithmetic function

Function name	function
ACOS	The result is equal to the arcosine of the source.
ADDSUM	Use addition to calculate checksum.
ASIN	The result is equal to the arcsine of the source.
ATAN	The result is equal to the arctangent of the source.
COT	The result is equal to the cotangent of the source.
COS	The result is equal to the cosine of the source.
CRC	Get 16-bit CRC.
CSC	The result is equal to the cosecant of the source.
LOG	Calculate the natural logarithm of a number.
LOG10	Calculate the base-10 logarithm of a number.
POW	Calculates x raised to the power of y.
RAND	Produces a pseudorandom number (range:0~65535)
SEC	The result is equal to the secant of the source.
SIN	The result is equal to the sine of the source.
SQRT	The result is equal to the square of the source.
TAN	The result is equal to the tangent of the source.
XORSUM	Use XOR to calculate checksum.

1、ACOS

[Description]

The result is equal to the arcosine of the source.

[Usage]

result = ACOS(source);

[Example]

float source=0.5;

float result;

result=ACOS(source);

2、ADDSUM

[Description]

Use addition to calculate checksum.

[Usage]

checksum=ADDSUM(data, sizeof(data));

[Example]

char data[5] = {0x1, 0x2, 0x3, 0x4, 0x20};

int checksum;

checksum=ADDSUM(data, 5); // checksum=0x2a;

3、ASIN

[Description]

The result is equal to the arcsine of the source.

[Usage]

result = ASIN(source);

[Example]

float source=0.5;

float result;

result=ASIN(source);

4、ATAN

[Description]

The result is equal to the arctangent of the source.

[Usage]

result = ATAN(source);

[Example]

float source=1;

float result;

result=ATAN(source);

5、COT

[Description]

The result is equal to the cotangent of the source.

[Usage]

result = COT(source);

[Example]

float source=45(度);

float result;

result=COT(source); //result = 1

6、COS

[Description]

The result is equal to the cosine of the source.

[Usage]

result = COS(source);

[Example]

float source=60(度);

float result;

result=COS(source); //result = 0.5

7、CRC

[Description]

Get 16-bit CRC.

[Usage]

bit_CRC=CRC(source,sizeof(source)) ;

[Example]

char source[5] = {0x1, 0x2, 0x3, 0x4, 0x5} ;

short bit_CRC ;

bit_CRC=CRC(source,5) ;

8、CSC

[Description]

The result is equal to the cosecant of the source.

[Usage]

result = CSC(source);

[Example]

float source=30(度);

float result;

result=CSC(source); //result = 2

9、LOG

[Description]

Calculate the natural logarithm of a number.

[Usage]

result = LOG(source);

[Example]

float source =100,result;

result = LOG(source); //result =4.61

10、LOG10

[Description]

Calculate the base-10 logarithm of a number.

[Usage]

result = lOG10(source)

[Example]

float source=100,result;

result = LOG10(source); //result = 2.00

11、POW

[Description]

Calculates x raised to the power of y.

[Usage]

result = POW(x,y); note:x must be constant

[Example]

float result,y;

y=0.5;

result=POW(25,y); //result = 5

12、RAND

[Description]

Produces a pseudorandom number (range:0~65535)

[Usage]

Random= RAND()

[Example]

unsigned short random;

random =RAND(); //random = 363

13、SEC

[Description]

The result is equal to the secant of the source.

[Usage]

result = SEC(source);

[Example]

float source=60(度);

float result;

result=SEC(source); //result = 2

14、SIN

[Description]

The result is equal to the sine of the source.

[Usage]

result = SIN(source);

[Example]

float source=30(度);

float result;

result=SIN(source); //result = 0.5

15、SQRT

[Description]

The result is equal to the square of the source.

[Usage]

result = SQRT(source);

[Example]

float source=16;

float result;

result=SQRT(source); //result = 4

16、TAN

[Description]

The result is equal to the tangent of the source.

[Usage]

result = TAN(source);

[Example]

float source=45(度);

float result;

result=TAN(source); //result = 1

17、XORSUM

[Description]

Use XOR to calculate checksum.

[Usage]

checksum = XORSUM(source, sizeof(source));

[Example]

char source[5] = {0x1, 0x20, 0x3, 0x48, 0x5};

short checksum;

checksum = XORSUM(source, 5); // checksum =0x6f;

7.8 Communication function

Function name	function
CLEARBUFFER	Clear buffer of communication port.
DELAY	Set a delayed time
FillLW_8	Give 8-bit dates to LW register
FillLW_16	Give 16-bit dates to LW register.
FillLW_32	Give 32-bit dates to LW register.
FillLW_Float	Give float dates to LW register.
GETCHARS	Get data from communication port.
GETBUFFERLENGTH	Get buffer length from communication port.
PUTCHARS	Send data to communication port.
InitEthernet	TCP/IP client mode initial connect.
readEthernet	TCP/IP client mode read receive data.
writeEthernet	TCP/IP client mode write send data.
CountsEthernet	TCP/IP client mode counts in receive buffer;
ClearEthernet	TCP/IP client mode delete all data in receive buffer.

1、CLEARBUFFER

[Description]

Clear buffer of communication port.

[Usage]

int CLEARBUFFER(PortID);

[Example]

int Clearresult,PortID;

PortID=0;

Clearresult= CLEARBUFFER(PortID);

2、DELAY

[Description]

Set a delayed time ;

[Usage]

void DELAY(int dwMilliseconds);

[Example]

int dwMilliseconds=1000;

DELAY(dwMilliseconds);

3、FillLW_8

[Description]

Give 8-bit dates to LW register;

[Usage]

FillLW_8(&LocalWord[i],a,NULL,count);

FillLW_8(&LocalWord[i],NULL,a,count);

[Example-1]

char a[6]=" arsdw" ;

FillLW_8(&LocalWord[6],a,NULL,5);//LW6='a',LW7='r',LW8='s',LW9='d'LW10='w'。

[Example-2]

char a[6]={1,2,-1,3,4};

FillLW_8(&LocalWord[6],NULL,a,5);//LW6=1,LW7=2,LW8=-1,LW9=3,LW10=4。

4、FillLW_16

[Description]

Give 16-bit dates to LW register.

[Usage]

FillLW_16(&LocalWord[i],a,NULL,count);

FillLW_16(&LocalWord[i],NULL,a,count);

[Example]

short a[6]={12,1,2,-3,4};

unsigned short b[6]={12,5,4,7,8};

FillLW_16(&LocalWord[0],b,NULL,5);//LW0=12,LW1=5,LW2=4,LW3=7,LW4=8。

FillLW_16(&LocalWord[6],NULL,a,5);//LW6=12,LW7=1,LW8=2,LW9=-3,LW10=4。

5、FillLW_32

[Description]

Give 32-bit dates to LW register.

[Usage]

FillLW_32(&LocalWord[i],a,NULL,count);

FillLW_32(&LocalWord[i],NULl,a,count);

[Example]

int a[6]={12,1,2,-3,4};

unsigned int b[4]={0x123465,0x541245,0x4444444};

FillLW_32(&LocalWord[0],b,NULL,3);//LW0=1193061,LW2=5509701,LW4=71582788。

FillLW_32(&LocalWord[6],NULL,a,5);//LW6=12,LW8=1,LW10=2,LW12=-3,LW14=4。

6、FillLW_Float

[Description]

Give float dates to LW register.

[Usage]

FillLW_Float(&LocalWord[i],source,count)

[Example]

float a[3]={1.2,3.6,4.5};

FillLW_Float(&LocalWord[8],a,3);//LW8=1.2,LW10=3.6,LW12=4.5。

7、GETCHARS

[Description]

Get data from communication port.

[Usage]

GETCHARS(PortID, Data)；

[Example]

char Data[20]；

int Length,PortID,Result；

Length=5；

PortID=0；

Result=GETCHARS(PortID,Data)；

8、GETBUFFERLENGTH

[Description]

Get buffer length from communication port.

[Usage]

int GETBUFFERLENGTH(PortID);

[Example]

int bufferlen,PortID;

PortID=0;

bufferlen=GETBUFFERLENGTH(PortID).

9、PUTCHARS

[Description]

Send data to communication port.

[Usage]

PUTCHARS(PortID, Data, Length) ;

[Example]

char Data[5] = {0x02, 0x30, 0x31, 0x4d, 0x5e} ;

int Length,PortID ;

Length=5 ;

PortID=0 ;

PUTCHARS(PortID, Data, Length) ;

10、InitEthernet

[Description]

TCP/IP client mode initial connect.

IPAdd--server IP address;

networkPort--connect port number;

Success return 1, fail return 0.

[Usage]

int InitEthernet(char *IPAdd, int networkPort);

[Example]

char *IPAdd= “192.168.1.100”;

int networkPort=5 ;

int result ;

result= InitEthernet(IPAdd, networkPort) ;

11、readEthernet

[Description]

TCP/IP client mode read receive data.

cBuffer--data buffer;

Success return receive counts, fail return -1.

[Usage]

int readEthernet(char *cBuffer);

[Example]

char cBuffer[]={22,33};

int result;

result=readEthernet(cBuffer);

12、writeEthernet

[Description]

TCP/IP client mode write send data.

cBuffer--data need to send;

iSize--counts need to send;

Success return 1, fail return 0.

[Usage]

int writeEthernet(char *cBuffer,int isize);

[Example]

char cBuffer[]={22,33};

int isize=8;

int result;

result=writeEthernet(cBuffer,isize);

13、CountsEthernet

[Description]

TCP/IP client mode counts in receive buffer;

Success return counts in receive buffer, fail return 0

[Usage]

int CountsEthernet();

[Example]

int result;

result= CountsEthernet();

14、ClearEthernet

[Description]

TCP/IP client mode delete all data in receive buffer.

Success return 1, fail return 0.

[Usage]

int ClearEthernet();

[Example]

int result;

```
result= ClearEthernet();
```

8. Simulation

This chapter mainly introduces usage and steps for offline simulation and online simulation.

Tips:

“Run HMITOOL as Administrator” is necessary to execute functions of Offline simulation and Online simulation.

8.1 Offline Simulation

You can check the correctness of the configuration project with the off-line simulation provided by HMITOOL before transferring it to the HMI and connecting the HMI to connected devices.

Operation process:

Save the current project, select the offline menu to execute the simulation command, then it appear a pop-up offline simulation window where you can control the project to realize part of HMI functions by clicking the mouse instead of touching the HMI, as shown in Figure 6-1:

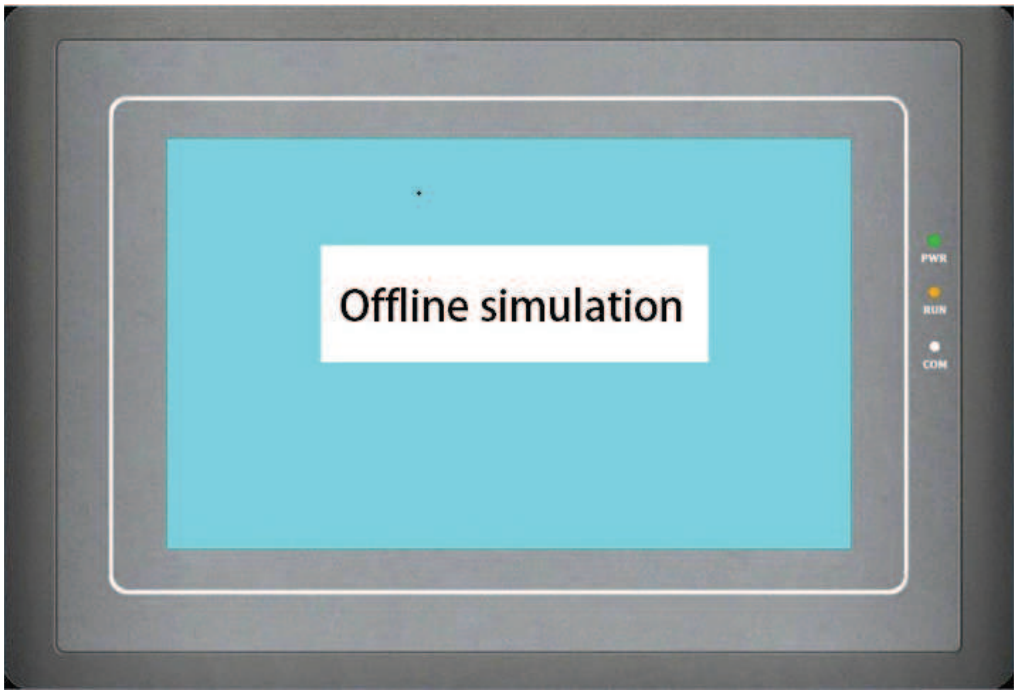
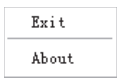


Figure 6-1 Offline simulation

In the simulation window, right click to bring up the following menu:



- Exit: Close the offline simulation window; press the keyboard ESC can also exit the offline simulation.
- About: About the dialog box.

Tips:

You can close offline simulation through taskbar “Right Key”-“Close”.

Offline simulation example

As shown in Figure 6-2, the edited configuration project can be off-line simulated so as to check and find errors in engineering configuration screens, such as

whether the monitor address is correct and so on. Just in case.

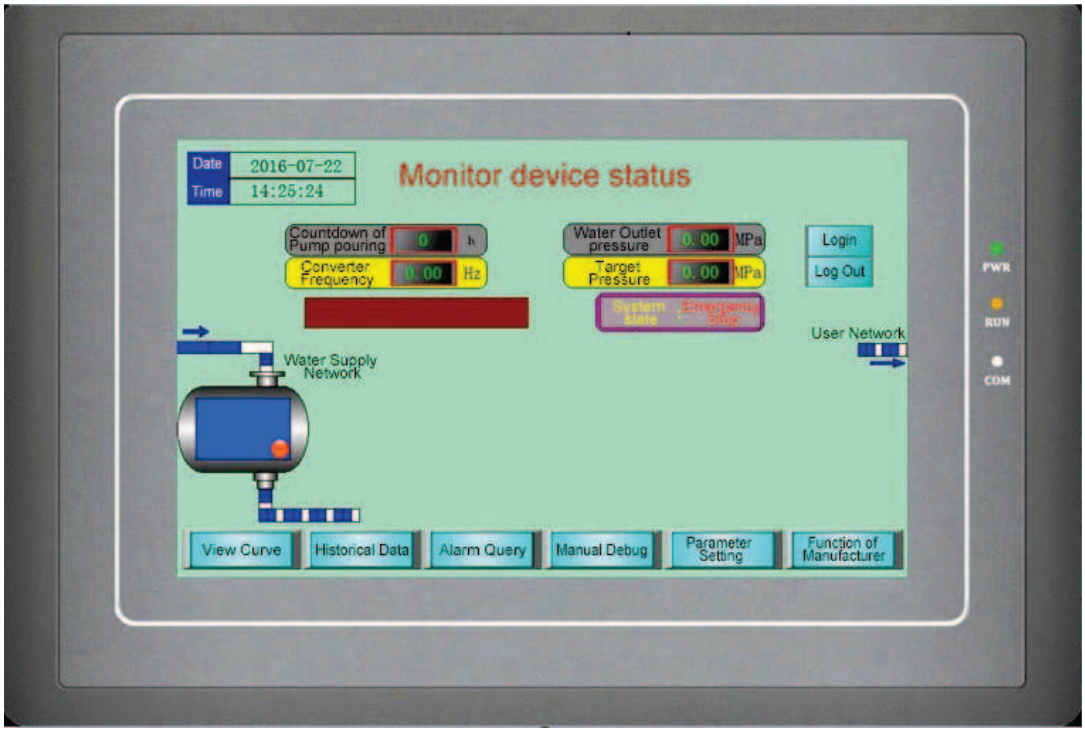


Figure 6-2 Example of offline simulation

Steps to execute offline simulation:

Install HMITOOL Software first;

Run HMITOOL as Administrator.

Tips:

Note: In the offline simulation, only these functions of Function button are available: “Previous recipe", "Next recipe", "Save current recipe", "Change User’s Level", "Logoff" and "Touch Sound ON / OFF ". The Historical Alarm, Historical Data Display, Historical Trend Graph, Timer, and Macro on the Toolbar cannot be simulated.

The other controls are the same operation. After setting properties of each control, click "Save" and then select Offline simulation, as shown in Figure 6-3:

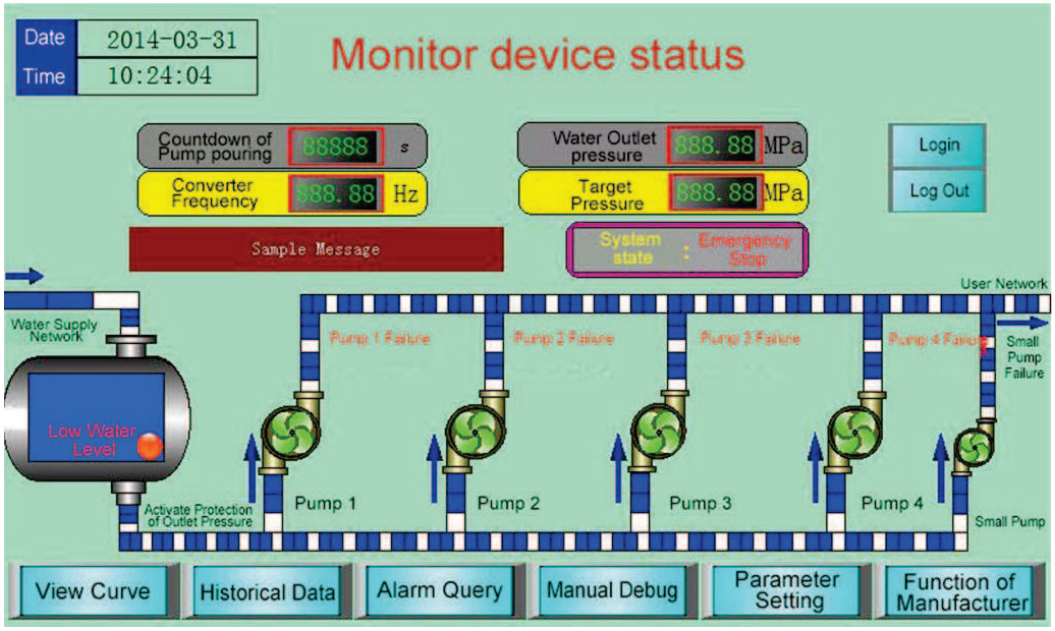


Figure 6-3Offline simulation

8.2 Online Simulation

Online simulation allows communicating between PC and PLC or other related devices without HMI. It is employed to debug the configuration project and conduct testing work when the HMI fails through simulating its operation status. Refer to offline simulation for configuration editing.

Online simulation requires the connection to PLC, so it needs to ensure that the cable connecting PLC and PC functions properly. The online simulation runs in 30 minutes, after that, it will quit automatically.

Tips:

Note: It is the download cable that is needed for common PLC, but not that connecting PLC to HMI. Please contact technical support staff in case of communication failure.

9. System Settings

This chapter introduces setting functions of HMITOOL system. It is necessary to set them up in order to ensure normal running of HMI, including that of PLC.

Contents：

9.1 Communication Port Property

Communication port property is applied to set the communication parameters between HMI and connected devices. HMIs like VS-070F，VS-070H，VS-102H，VS-121F,VS-150AS are equipped with two communication ports, COM1 and COM2, supporting simultaneous communication with two different PLCs; and parameters of each port must be set. HMIs like VS-035F，VS-043F，VS-043H have one communication port, COM1, and it communicate only with one connected device.

Double click "Link" and "Link 1", as shown in Figure 7-1:

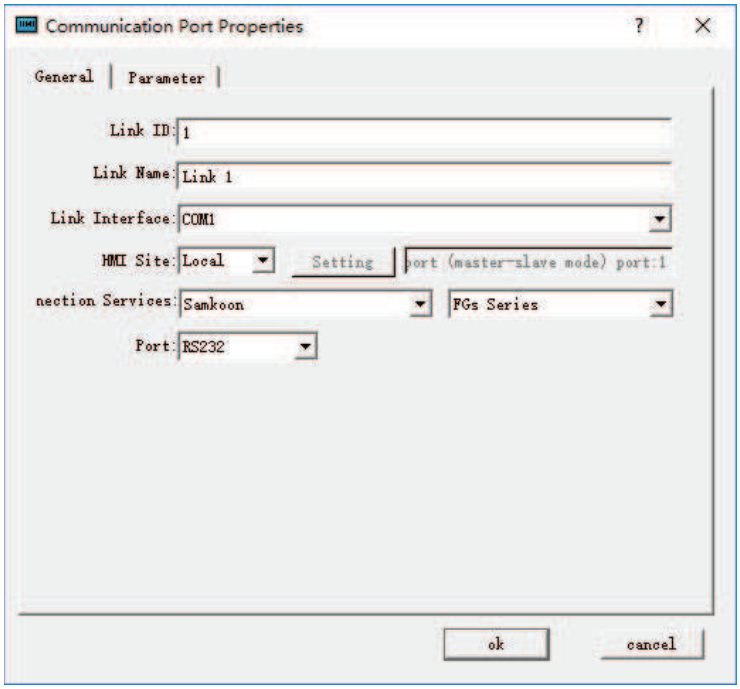


Figure 7-1 COM1 COM port property dialog box

In the "Link" page, various brands and models of PLC are available. In addition, the "Link name" and "PLC consecutive address interval"

Click the Parameter tab of the dialog box, as shown in Figure 7-2:

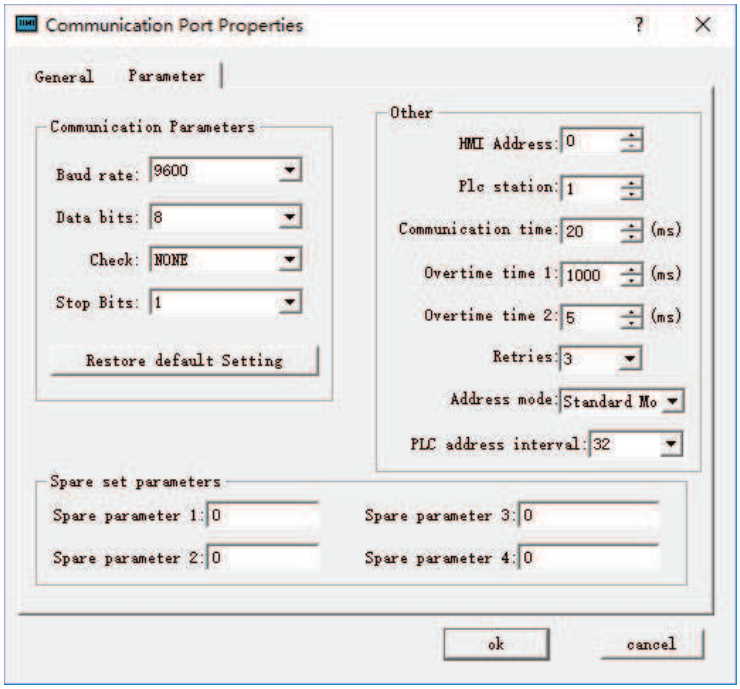


Figure 7-2 COM1 Communication port parameter setting

Parameter settings:

Equipment services: manufacturer, PLC brand and CPU model.

Port: RS232 / 485/ 422

Baud rate: 1200/2400/4800/9600/19200/38400/57600/115200 / 187.5k

Check: Odd / Even / None

Data bits: 7/8

Stop bits: 1/2

Model: PLC model

HMI station number: Set the station number of the HMI

Communication time: HMI sends data to the PLC and receives the data from PLC, and it cannot send data again until after the set communication time.

Overtime time 1 and Overtime time 2: Calculate first the value of Timeout 1 divided by that of Timeout 2, as shown in the figure above: the timeout period is 200ms. There is a situation called timeout if the HMI has not received the data 200ms after sending data.

Number of retries: The number of times that the data is retransmitted after a timeout

Address mode: ① Standard mode: One serial port connects one PLC; ② Extended mode: One serial port connects multiple PLCs of the same type

PLC Continuous Address Interval: The maximum number of words that PLC can read at one time

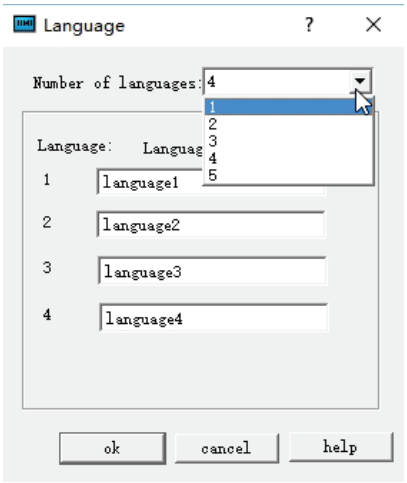
9.2 Setting

Let’s come to some settings of HMI. It is also important for project configuration. We can change HMI system settings in a narrative way to achieve the expected effect of users.

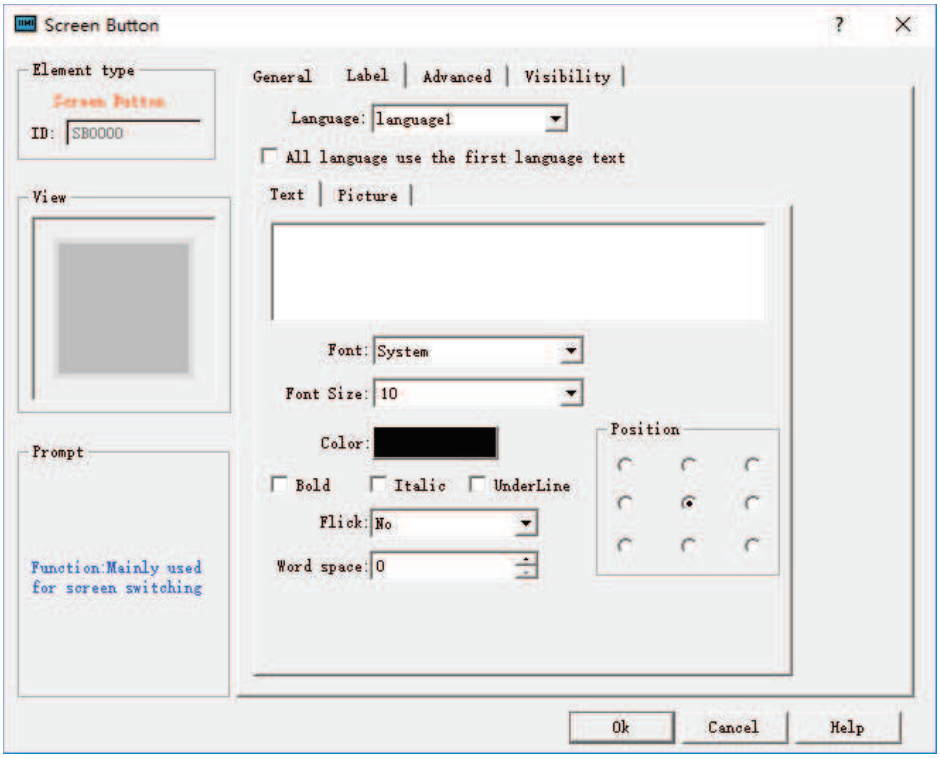
Contents:

9.2.1 Language

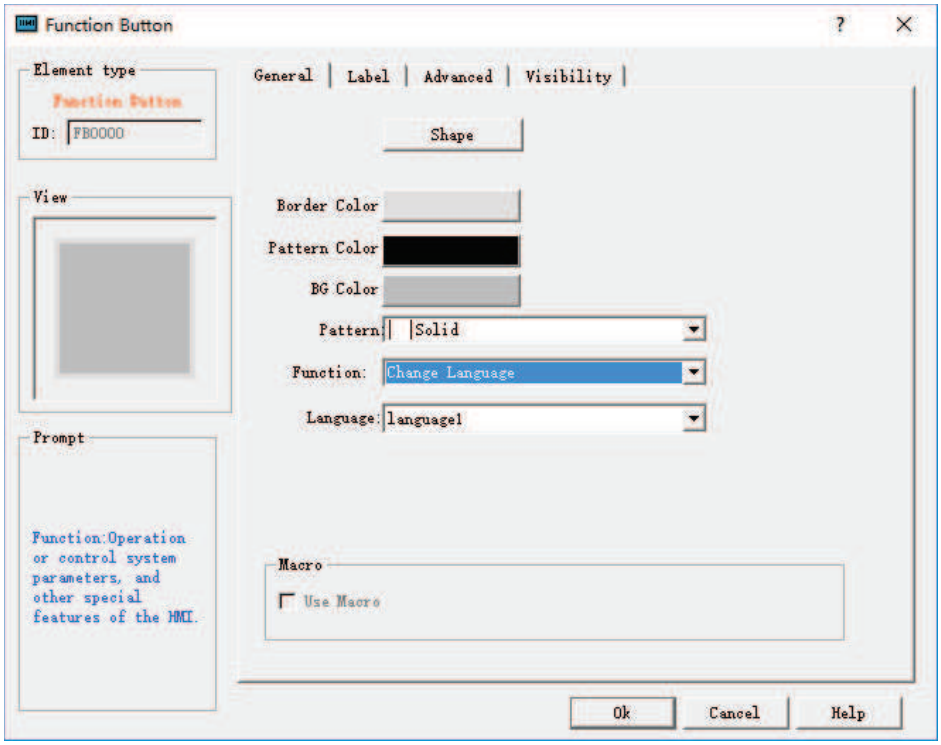
Language: This function realize multiple languages; it is necessary to preserve languages employed in configuration in character library of PC.
Double click the Language option or right click to open the dialog box of Language Setting.

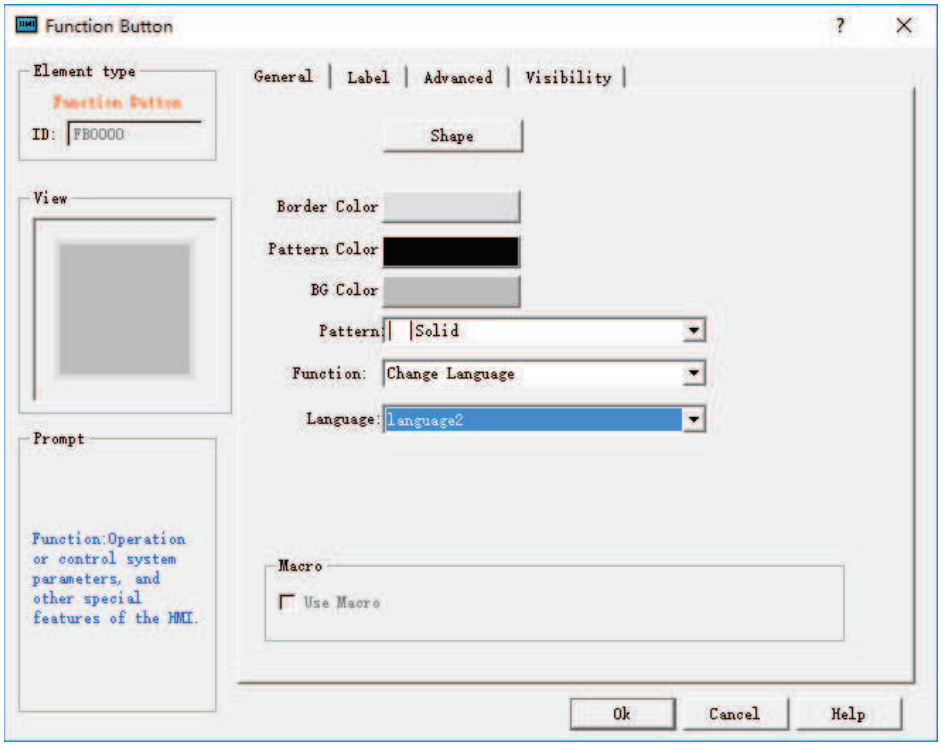


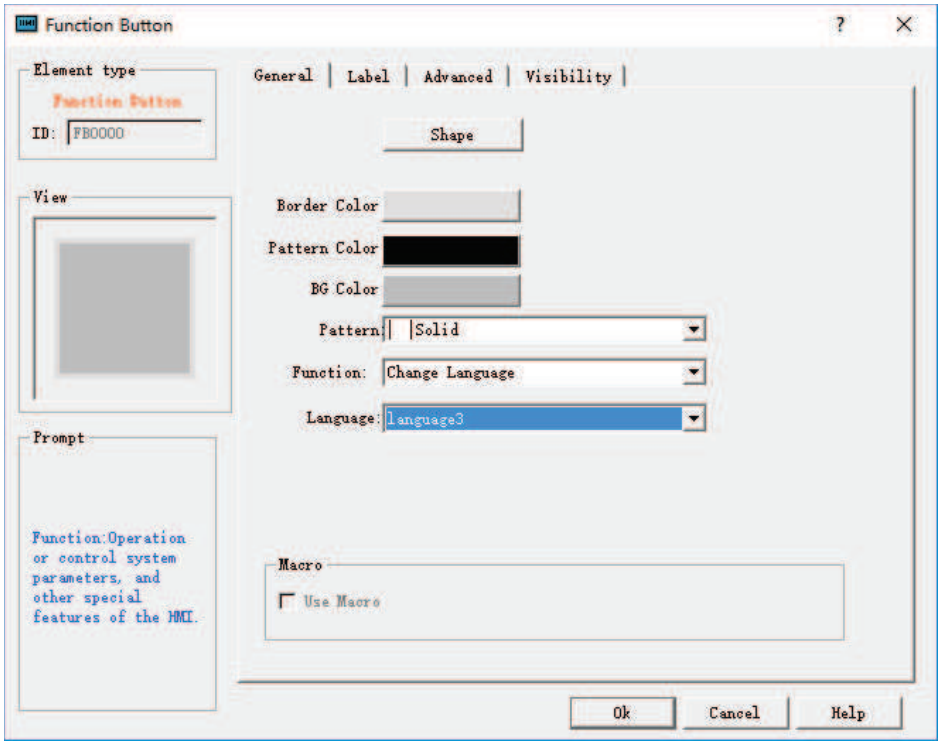
Language total: Set total number of the system language, at most five languages.
Language Name: Set a name for each language.
Example:
1. As shown above, set the total number of languages to 3, and the language name is by default; create a new text input control, such as Screen button, select Language 1 in the language drop-down box, enter in the text box " lang1", and then set the parameters of font size and color, etc.; select Language 2 in the language drop-down box, then enter "lang2"; select Language 3 in the language drop-down box, then enter in the text box "lang3", as shown below:



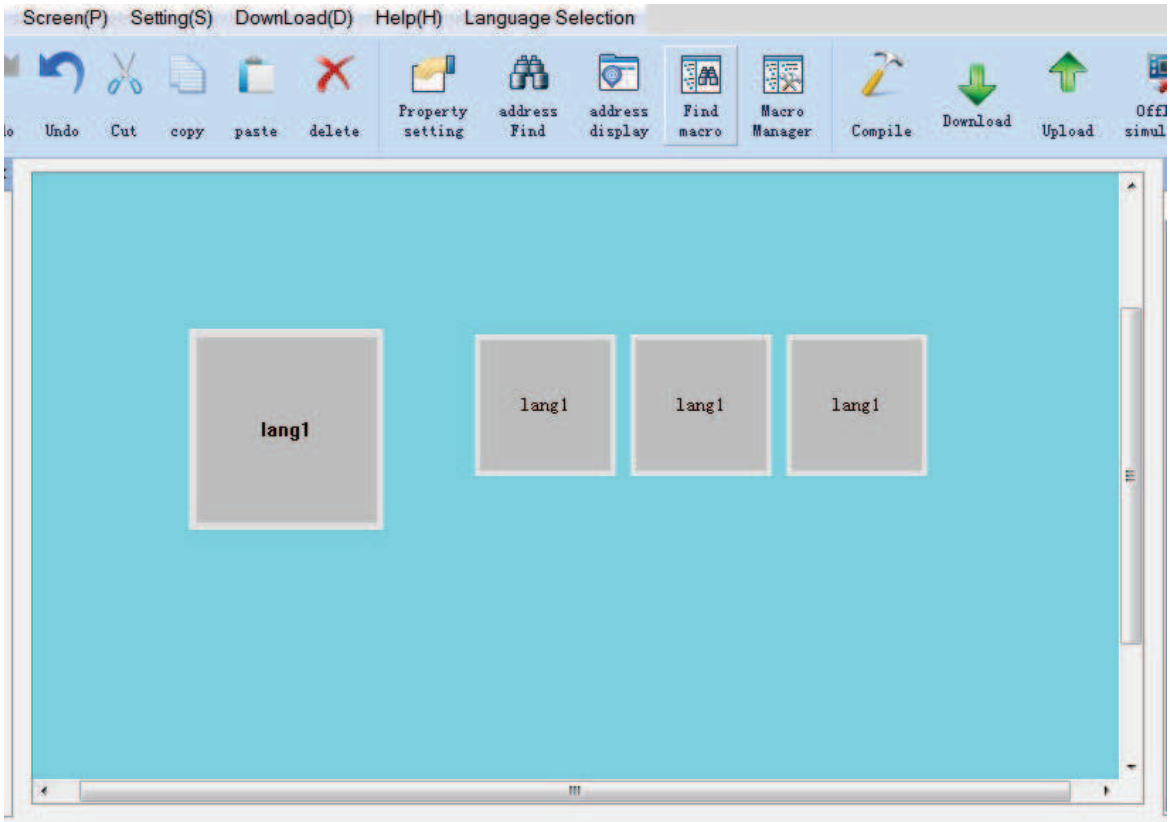
2. Click "OK" after the settings, and then left click on the screen to add this Screen control button when the mouse cursor shows a cross shape; adjust its size. After then create three function buttons, input texts on the "Label" page of each function button. It should note that "Function" options below "General" page of each button are the same: "Switch language"; while options of "Language" are different: Language1, Language2 and Language3, respectively.


Function Button
Element type
Function Button
ID: FB0000
View
Prompt
Function:Operation or control system parameters, and other special features of the HMI.
General
Label
Advanced
Visibility
Shape
Border Color
Pattern Color
BG Color
Pattern: Solid
Function: Change Language
Language: language1
Macro
Use Macro
Ok
Cancel
Help


Function Button
Element type
Function Button
ID: FB0000
View
Prompt
Function:Operation or control system parameters, and other special features of the HMI.
General
Label
Advanced
Visibility
Shape
Border Color
Pattern Color
BG Color
Pattern: Solid
Function: Change Language
Language: language2
Macro
Use Macro
Ok
Cancel
Help



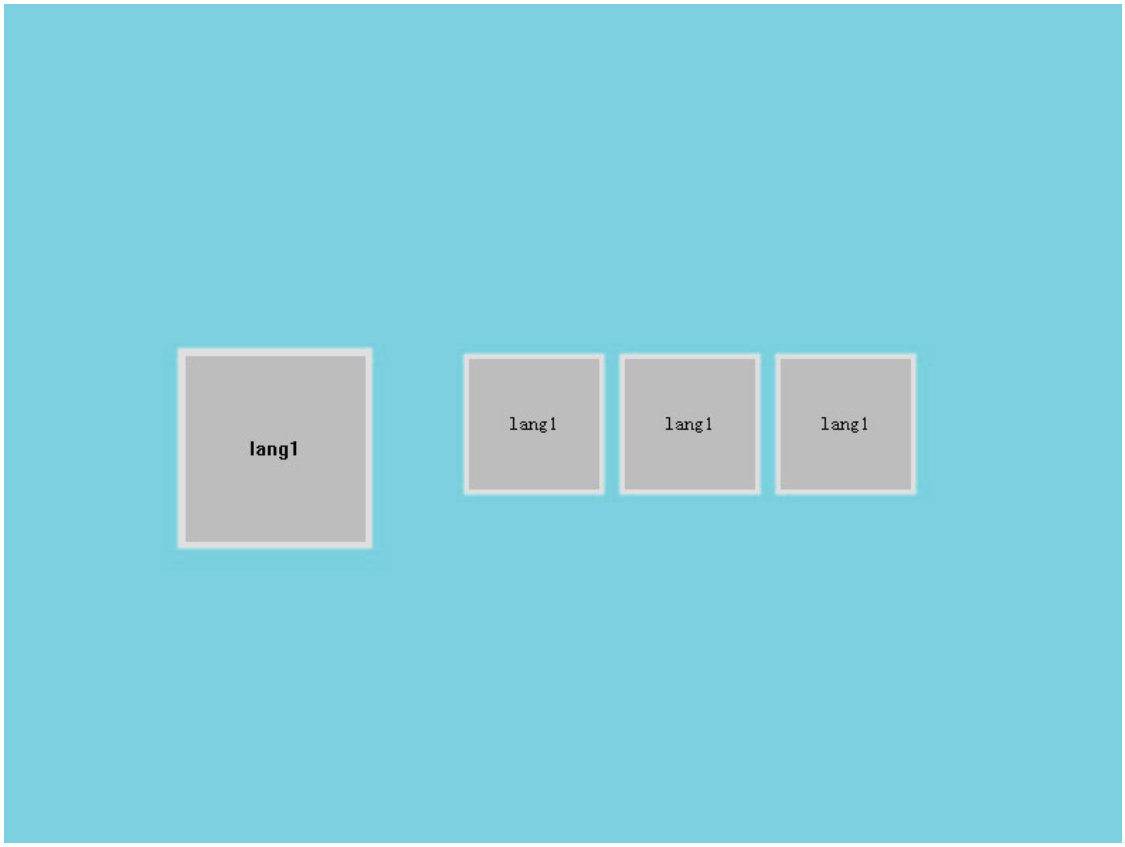
3. Save the project after the settings, as shown below:



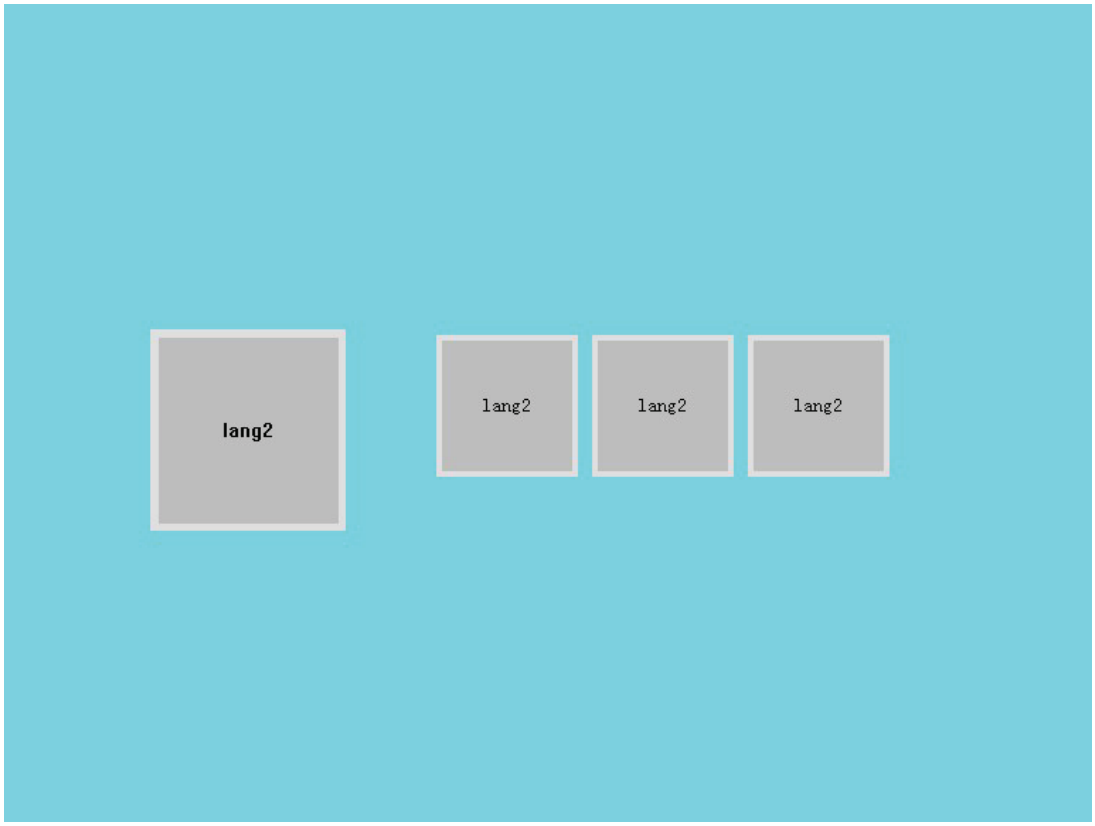
4. Simulate (or download to HMI)

All controls with text will be presented in the according language when click Function button 1, 2 or 3.

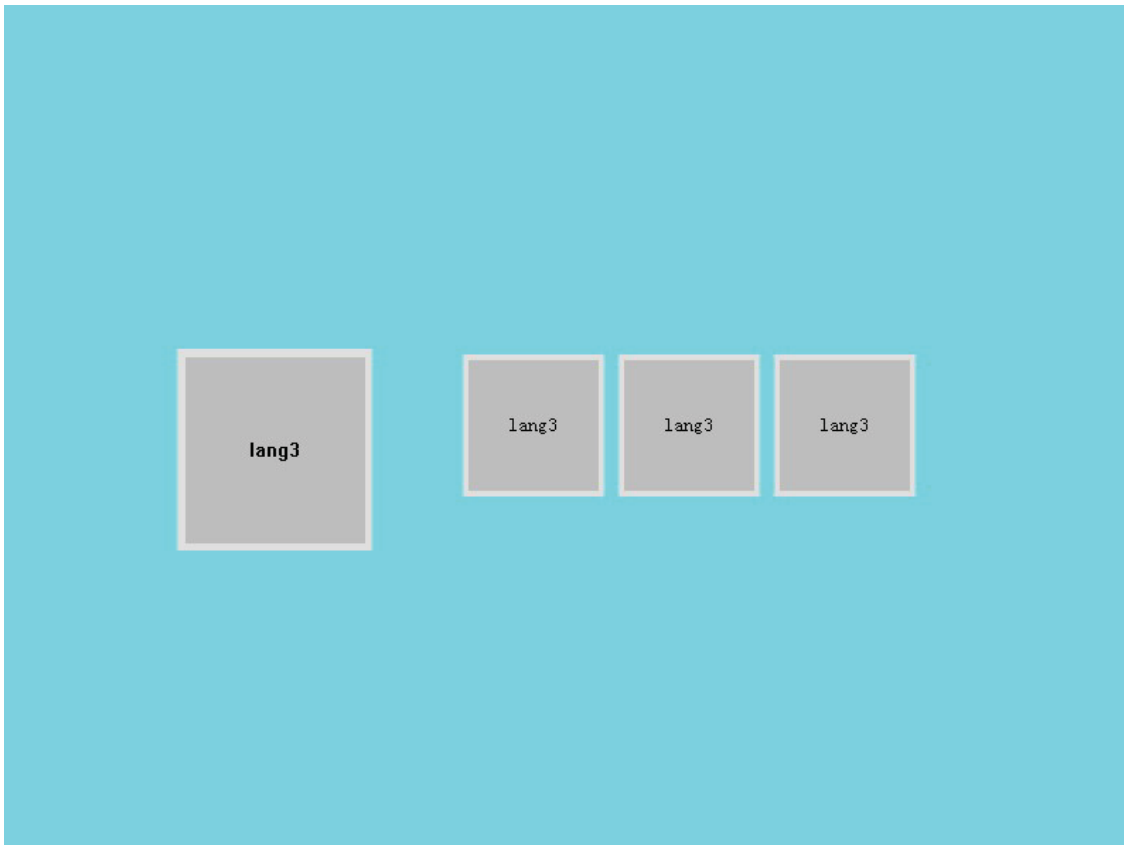
As shown below:



The figure above shows the state after clicking Function button 1 to switch the language to Language 1.



The figure above shows the state after clicking Function button 2 to switch the language to Language 2.



The figure above shows the state after clicking Function button 3 to switch the language to Language 3.

To achieve the multi-language function in the entire project, it is necessary to input different texts for each control or object with text (same operation with the first point). This enables the entire project to be switched to the specified language when execute the command of "Switch Language".

9.2.2 HMI parameter settings

HMI model can be assigned when create a project or be modified during the configuration through HMI parameter settings in the Project Manager.
Double click “HMI Parameter Setting: in the Project Manager, then a dialog box pops up, as shown in Figure 7-3:

Parameter Setting:

1. You can modify the parameter settings through Preference Setting in the drop-down menu Settings, or double-click the HMI Parameter Settings in Setting, Project Manager. Then a pop-up dialog box is shown as Figure 7-3 below:

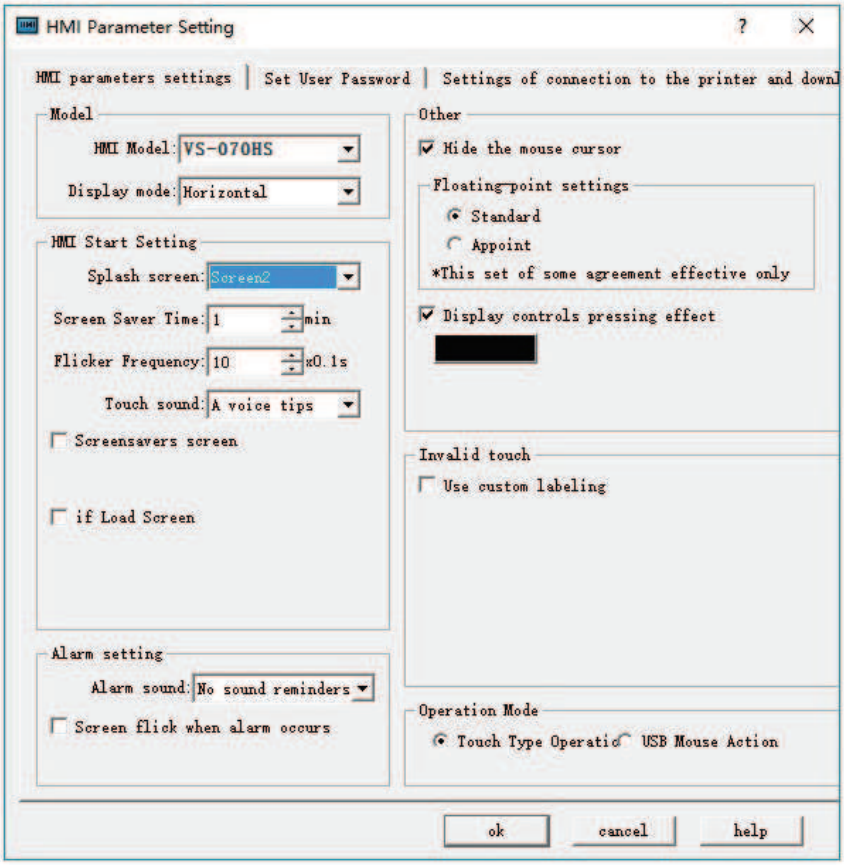


Figure7-3 HMI Parameter Setting dialog box

Display mode: Adjust horizontal or vertical display screen.
HMI Start Setting
Splash screen: The start screen of HMI when powered on
Screen Saver Time: Set screensaver time
Flicker Frequency: Set the flicker frequency of objects or controls which can blink; you can change the speed of flicker by inputting directly a value and clicking the spin button.
Alarm sound: When the alarm occurs, the touch screen can remind users through sound; the right drop-down menu contains two cases, with or without sound reminder.
Touch sound: Options in the right drop-down menu can decide Sound reminder or No sound reminder.
Screensavers screen: Select a screen among the existing screens as screensaver screen; just touch the screen to return the former screen.
If Load Screen: If choose this option, the HMI will display the selected loading screen after power-on for a certain while, and then switch to the splash screen; thus users can give some tips or notes.
Screen flicker in case of alarm: switch background colors between that of Alarm control or Alarm Bar and screen background color; flicker frequency is in accordance with the above. This function doesn’t work when the background is a picture or the background pattern is transparent.

- Other settings
- Hide mouse cursor: Whether to display the mouse
- Floating-point settings:
- [Standard]: High byte floating point ahead while the low one behind;
- [Appoint]: users determine the order.
- Invalid touch: Select the "Use custom labeling", which means touchable controls cannot be touched, then the picture marking the control chosen by user is untouchable.

- Operation Mode: Choose one mode between traditional touch operation and USB mouse operation.
- Display controls pressing effect: HMITOOL is newly added pressing effect display. When the pressing effect is applied, rectangular pressing trace will appear if users touch the control. The pressing trace color depends on user's preference. Currently Bit switch, Word switch, Screen button, Function button, Numeric input and ASCII input controls support this effect. The default is No effect.

Tips:

Note: Pressing effect of elliptical controls is still a rectangle.

2、Set user password:

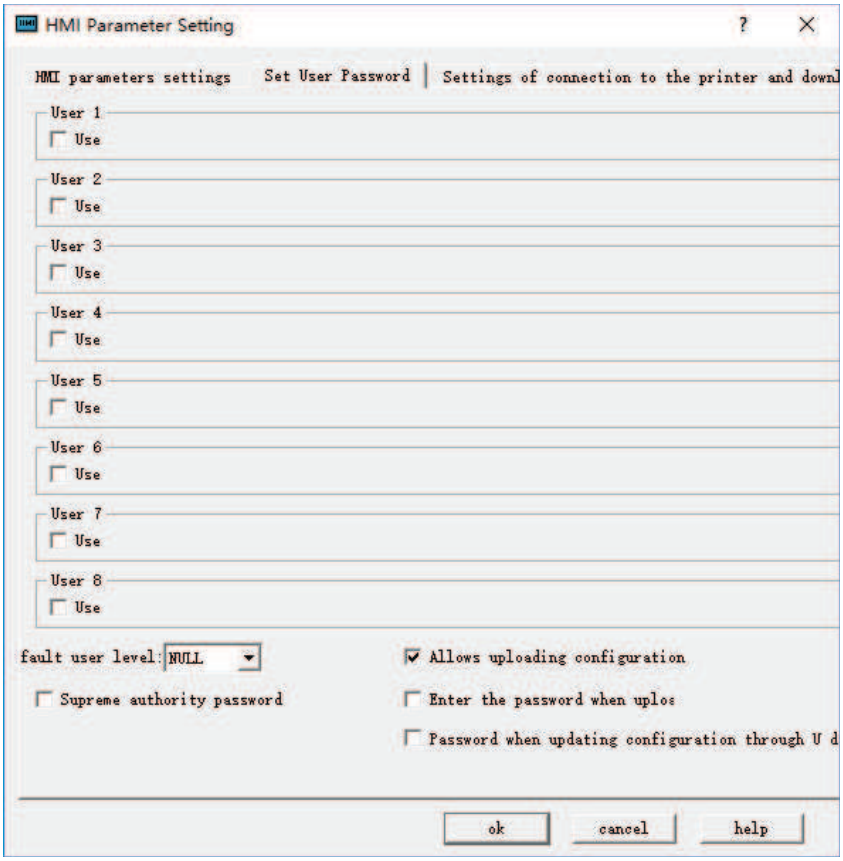


Figure 7-4 Set User Password

- This function supports 8 sets of user password, each of which has eight levels of password for users to choose.
- The default user level: The current default user class is NULL.
- Allow uploading configuration: If select this option, you can upload the project from HMI to PC; otherwise it cannot be uploaded.
- Enter the password when uploading: Whether uploading projects requires a password.
- Password when updating configuration through U disk: The password set up, the password is necessary when you update the HMI configuration projects through U disk.
- Supreme authority password: This password can operate all controls limited by grades.
- Upload: Whether this password enables upload configuration projects from HMI to PC.
- Permissions: Whether this password can perform all limiting operations.

3、**Settings of connection to the printer and downloading**

Click "Settings of connection between HMI and printer" button, as shown in Figure 7-5:

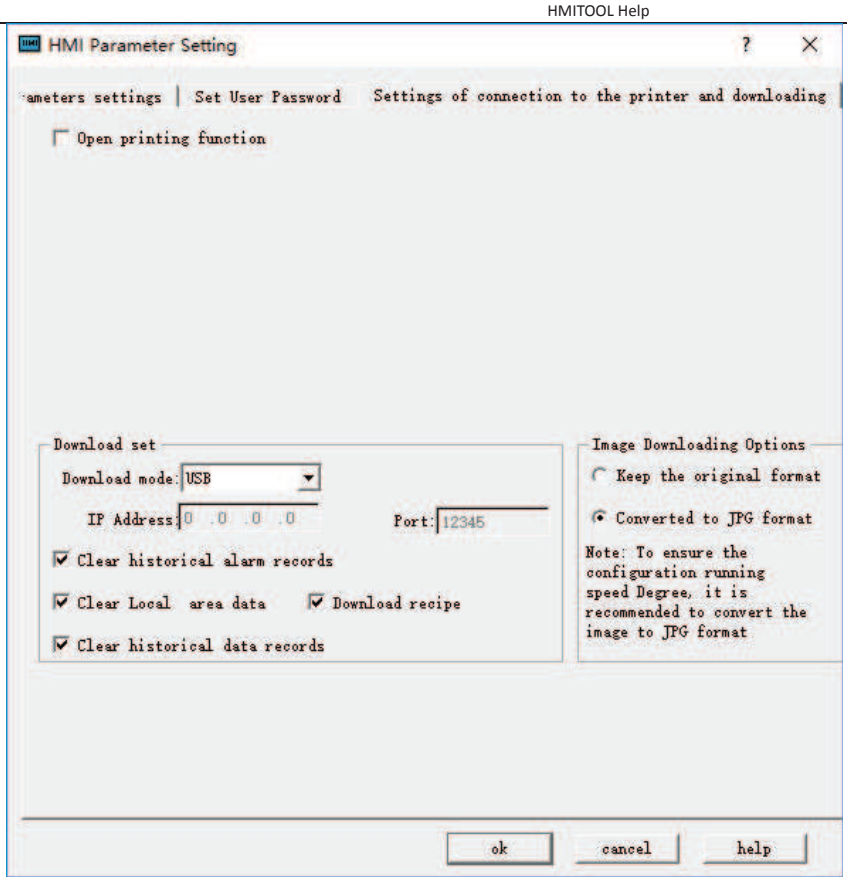


Figure 7-5 parameter settings of connection between the printer and the HMI

This page contains parameter settings of connection between the printer and the HMI; users can select corresponding parameters according to printer models.

Download settings:

Download mode: Users need to select USB or Ethernet connection to download. If choose the Ethernet, it's necessary to set the IP address and port number (IP address and port number are those of the HMI).

Clear historical alarm records: Whether to clear the history alarm information before downloading.

Clear historical data records: Whether to clear the history data information before downloading.

Download Formula: Whether to download formulas in the current project to the HMI.

Image Downloading Options:

Keep the original format: Selecting this option means retain the image format unchanged when downloading.

Convert to JPG format: Convert all images to JPG format before downloading them into HMI.

(Note: Due to the larger space occupied by BMP format images, it's suggested to select this option.)

4. Storage location and alarm system, as shown below

Figure 7-6 Storage location and Alarm system

- Storage location: Location for historical data, historical alarm and screen shots. Provide "Local (FLASH)", "U disk" and "SD card" three options for users to choose according to their need. The default is Local (FLASH).

- Alarm system
- Use the alarm system: Whether to use the system alarm. When an alarm occurs, the alarm system displays the current alarm information at the top or bottom of each screen according to the options set by users.
- Alarm background color, Text color, Font and Font size, these are attributes options of the alarm display system. Users can set these options as needed.
- Way of display: "Always display" refers to the alarm displayed all the time; "Cyclic display" means that alarm display or disappear within a certain period to achieve the flicker effect.
- Position of alarm bar: Choose the position where the alarm bar appears.

Tips:

Note: Change of the screen saver time is not valid until "Screensaver screen" is chosen; just touch the screen of screensaver to return to the former screen.

9.2.3 HMI State

HMI state setting is employed to write the current HMI state information to the data register of PLC.

Double click the "HMI State" item in the Project Manager, a dialog box will pop up as shown in Figure 7-7:

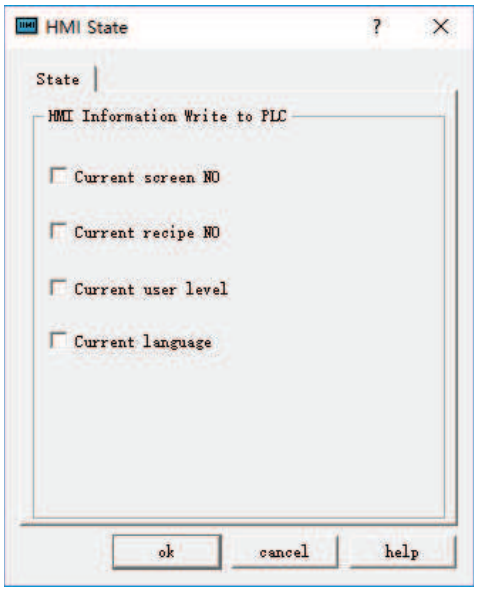


Figure 7-6 HMI Status Setting

- Current screen NO: Write the serial number of the current HMI screen to the PLC.
- Current formula NO: When the serial number of the current formula to the PLC.
- Current user level: Write the current user level to the PLC.
- Current language: Write the serial number of the currently applied language to the PLC

Click "OK" HMI to complete the settings.

9.2.4 PLC Control

PLC control means realizing operations of screen switching, change user level, change formula and write formula through PLC. Double click "PLC Control" in the "Project Manager", the dialog box shown in Figure 7-8 will pop up:

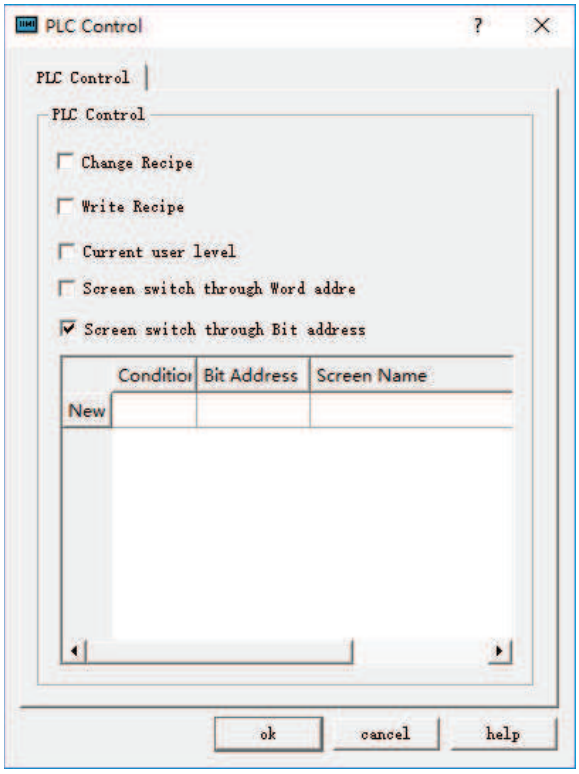


Figure 7-7 PLC control dialog box

- Change formula: HMI changes the formula according to its address value.
- Write formula: HMI controls the write of the formula based on its address value.
- Current user level: HMI controls the current user level according to this address value.
- Screen switch through Word address: Control screen switch through the word address. HMI switches the screen according to this word address value; if the address value is n, it jumps to the nth screen.
- Screen switch through Bit address: Double click the "New" line; a pop-up dialog box will appear as shown in Figure 7-9:

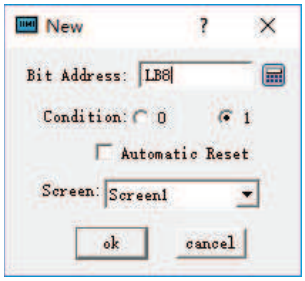


Figure 7-8 Create a new bit control screen switching dialog box

Input the "Bit Address", choose the "Condition" and the screen to jump to; click OK. Then a bit control is completed. As shown in Figure 7-10:

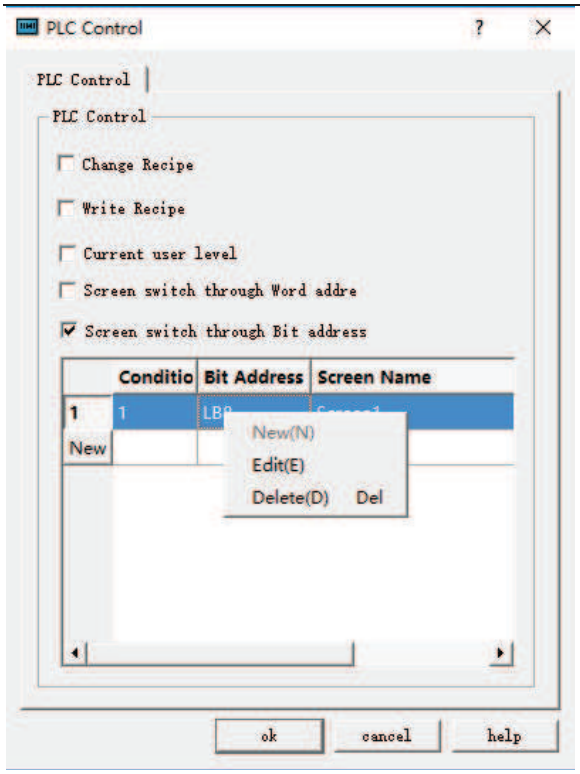


Figure 7-9 Add a bit control screen switch

Similarly, countless screen bit controls can be created; the order of screen switching implementation accords with the serial number. Right click on any information of bit control, a menu pops up then where to perform operations of "New", "Edit" and "Delete".

Figure 7-10: Its function is to jump to the scree 1 when the value of the bit address LB8 is equal to 1.

Tips:

Note: When switch screens through PLC, the value read for the first time does not affect the screen switching. It functions only when the value of PLC register is different from the first read value.

9.2.5 Clock

Clock setting includes three functions: downloading system time to HMI, writing time to PLC and synchronization with PLC.

Double click “Clock” in the Project Manager to open the attribute setting box of Clock.

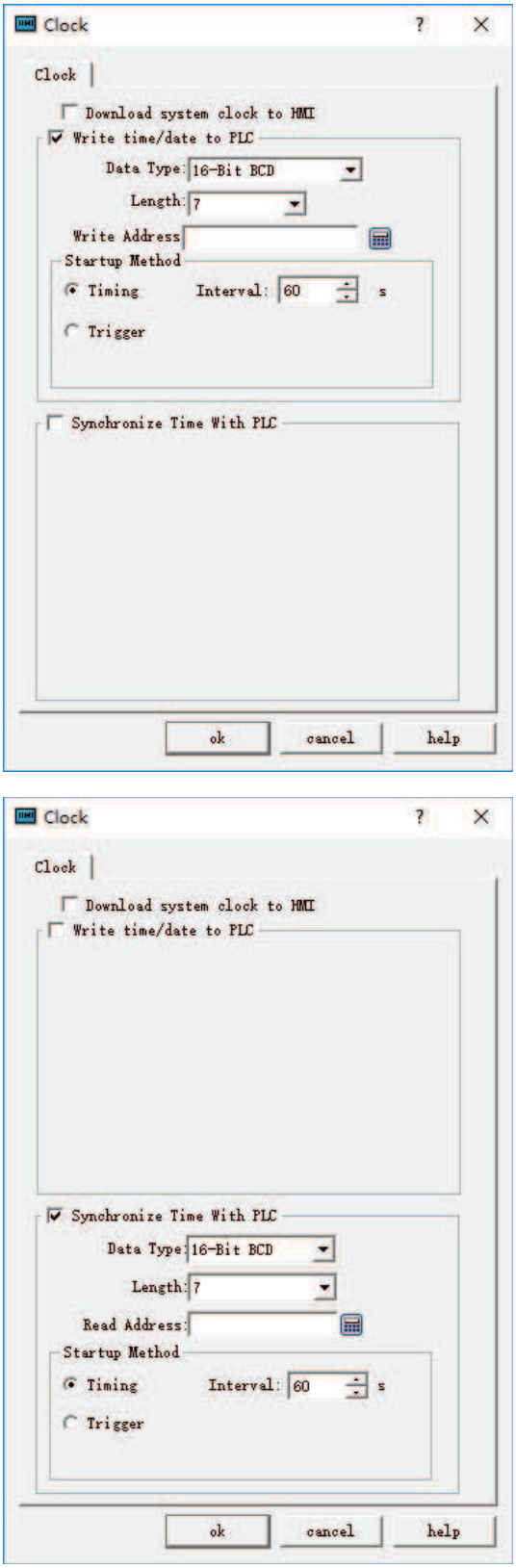


Figure 7-10 Clock Setting dialog box

Download system clock to HMI: Whether to download the system clock information to HMI.

- Write time/date to PLC:
  - The Data type and Length are default values.
  - Time: Year-Month-Day-Hour-Minute-Second-Week. Refer to LW 60000~LW 60006 in Register Address for detailed information. It cannot modify the Week in the system time.

    Write address: The system time information will be written into this PLC address.
  - Startup Method:

Timing: Write the system time information into the specified PLC address periodically at specified time internal.
Trigger: The system time information will be written into the specified PLC address when the trigger address is 1.
Auto Reset: Reset automatically the trigger address to 0 when it is 1.

Synchronization with PLC time:

The Data type and Length are default values.

Time: Year-Month-Day-Hour-Minute-Second-Week. Refer to LW 60000~LW 60006 in Register Address for detailed information. It cannot modify the Week in the system time.

- Read address: The information in the PLC address will be read.
- Startup Method:

Timing: Write the system time information into the specified PLC address periodically at specified time internal.
Trigger: The system time information will be synchronized with the PLC when the trigger address is 1.
Auto Reset: Reset automatically the trigger address to 0 when it is 1.

9.2.6 File Encryption

Whether a password is necessary to open the project file.

Double click the "File Encryption" option in the “Project Manager", as shown in Figure 7-12 pop-up dialog box:

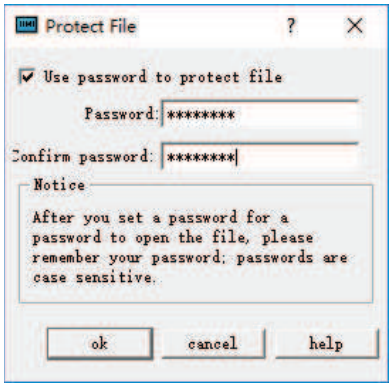


Figure 7-11 File Encryption dialog box

Select the "Use password to protect file" in the above dialog box; enter in the Password and Confirm Password; click "OK" to complete.

In this case, if user reopens the project, a password input box will be displayed as shown in Figure 7-13:

Figure 7-12 Password input box

Users need to input secret to open project.

9.2.7 HMI Protection

HMI protection is mainly employed to set a certain period of time during which HMI can functions normally. If the time is beyond the set period, HMI will jump to a specify screen where a password is necessary to re-use the HMI.

Double click the "HMI protection" option in the "Project Manager" to pop up dialog box shown in Figure 7-13:

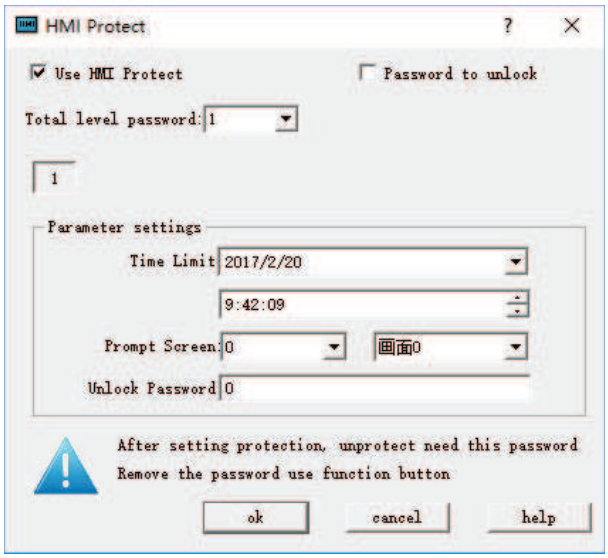


Figure 7-13 HMI Protection dialog box

Example to illustrate the application of this function:

➢Select "Use password protection": A password is required to enter the HMI protection the next time.

➢Select "Use HMI protection"

Set the "Total level password" to 3; select the 1 button icon, choose 2009-04-01 and 12-00-00 in the Time Limit column under the "Parameter settings"; decide the screen 1 as the "Prompt Screen"; Set "Password to unlock" to 1111. Then the level 1 protection is set up.

As in the above case, set the time limits and passwords for level 2 and level 3.

Assuming it is 8 o'clock, 2009-04-01now, since the time limit of level 1 HMI protection is 12-00-00, 2009-04-01, the HMI can still runs normally; while 4 hours, when it is 12 o'clock, the HMI will automatically jump to the set prompt screen (screen 1). Add a function button with the function of "Unlock HMI" to this screen. Then enter 1111 in the pop-up password input box of this button so as to remove the level 1 password protection and continue to operate.

Similarly, it needs to input the corresponding passwords when it reaches the time limit. When the protection of level 3 is removed, the HMI protection is hereby invalid.

This example illustrates the application of HMI protection; settings of the password level are similar to the above case.

9.2.8 Variable Table

Define a tag with an address. When the tag is employed, it corresponds to the corresponding address. To configure the variable settings, double click the "variable table" option in the "Project Manager", then a dialog box will pop up as shown in Figure 7-14:

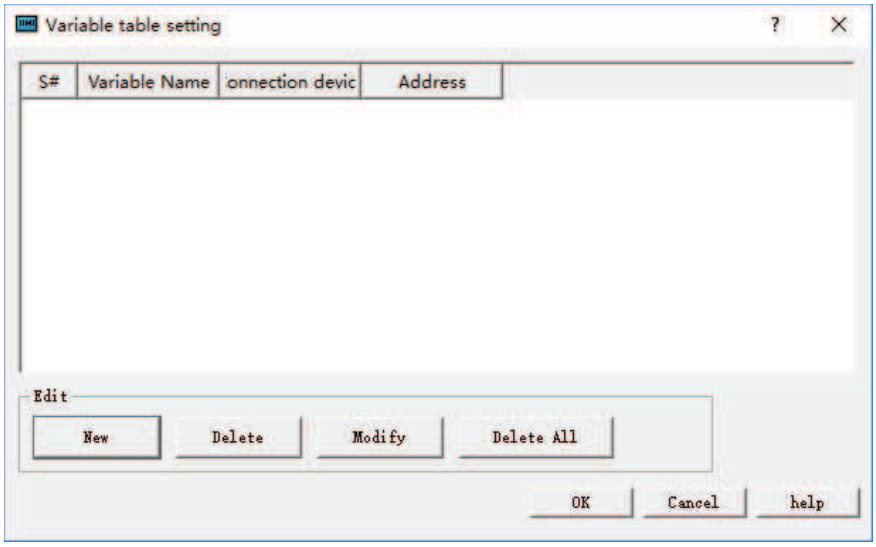


Figure 7-14 Variable Table Setting Dialog Box

- New: Add a new tag; the maximum total number is 1000
- Delete: Delete the selected tag.
- Modify: Modify the selected tag or double click the selected tag.
- Delete all: Delete all variable tags.

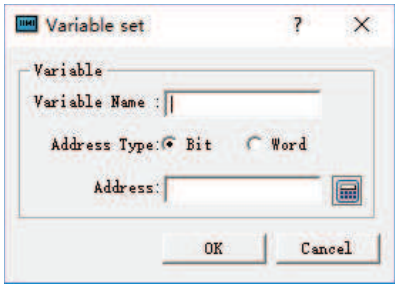


Figure 7-15 Create a screen switch through bit control dialog box

- Variable name: Set the name of the newly created variable.
- Address Type: Select the data type of the address.
- Address: Select the address of the connection.

9.3 Screen

Create a new screen as shown in Figure 7-16:

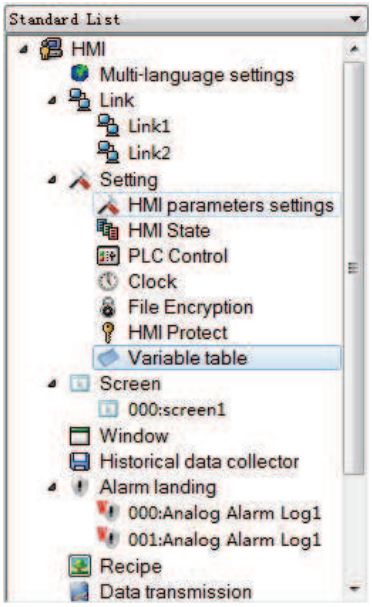


Figure 7-16 Right-click screen options

- Right click the "Screen" in the Project Manager, and then select "Add Screen" to create a new one. In the pop-up dialog box, set the properties such as "Screen Name"; click "OK".
- Two ways to open other screens: double click the screen name in the "Project Manager"; right click the screen name and select "Open" in the pop-up menu.
- Delete: Delete the screen by right clicking the screen name and selecting "Delete" in the pop-up menu.
- Property: Right click the screen name, and select "Property" in the pop-up menu; or, the select "Screen Properties" in the menu bar.
- Copy: Right click the screen name, and select "Copy" in the pop-up menu to copy; set the new screen name and click OK to complete.

9.4 Window

Way to add a new window is similar to that to create a new screen. Refer to screen operations for the open, delete, copy or properties of window. The Properties dialog box of the window is shown in Figure 7-17.

Open a window in screen, and a window will display in the screen.

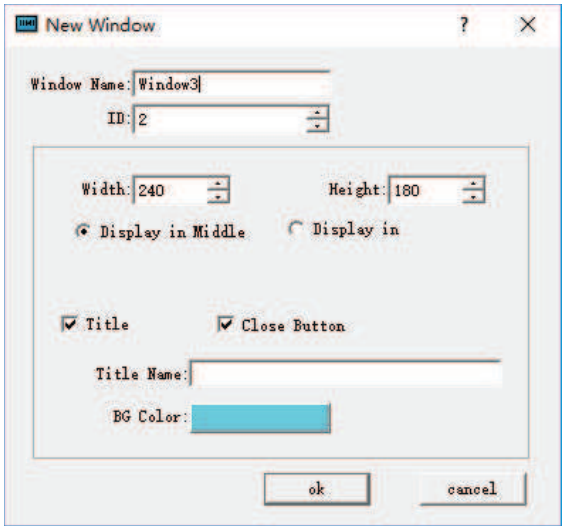


Figure 7-17 Window Properties dialog box

- Height, Width: Set the height and width of the window respectively.
- Display in Middle: When the window pops up, it is displayed in the center of the HMI.
- Display in: Set the X-coordinate and Y-coordinate point on which to display the window. The origin of coordinates is the top left corner of the HMI screen.
- Title: Enter the title name that will pop up as a title bar with the window.
- Close Button: Whether the window has a close button.
- Background Color: Set the background color of the window.

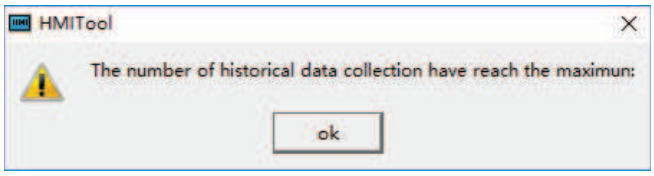
9.5 Historical Data Collector

The Historical Data Collector functions in conjunction with the Historical Data Display on the toolbar. The former is mainly applied for parameter settings while the later is mainly to display the historical data.

Create a historical data collector by right clicking "Historical Data Collector" and select "New Historical Data Collector" in the pop-up menu.

Note: HMITOOL newly supports multiple sets of historical data; up to four historical data collectors can be built.

Note that you cannot create more than four sets; otherwise an error dialog box will pop up as follows:



Double click "Historical Data Collector 0" in "Project Manager" to open a dialog box as shown in Figure 7-18:

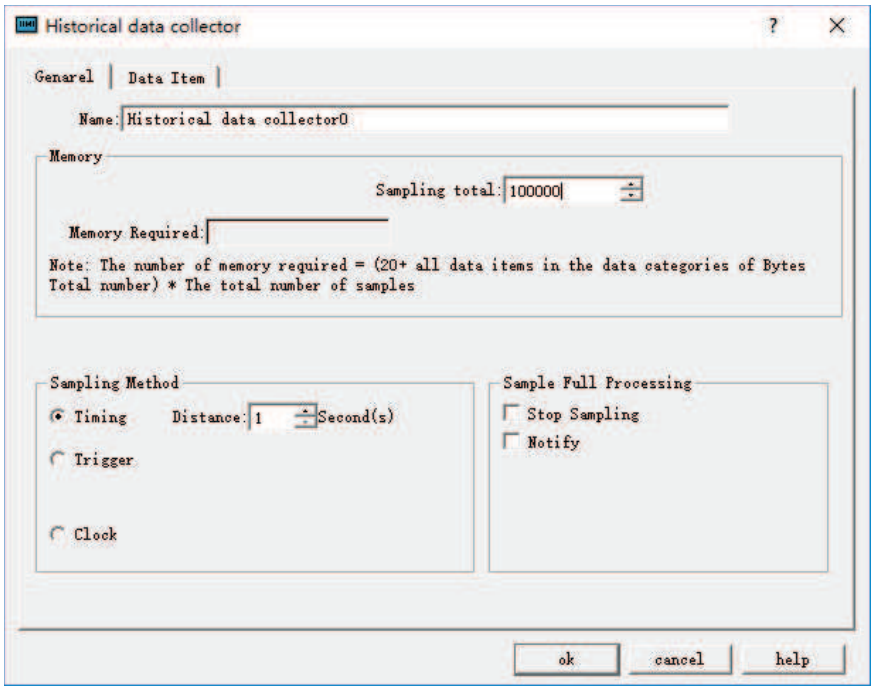


Figure 7-18 Historical Data Collector dialog box

- Name: Set the name displayed in the Project Manager.
- Sampling length: Number of data to be read each time from the memory.
- Total number of samples: Times of extracting data.

Tips:

Note: The memory required = (20 + total bytes of data types occupied by all data items) * total number of samples.

"20" refers to the bytes occupied by time and date;

“Total bytes of data types occupied by all data items” refers to the sum of the data types selected for each item on the Data Items page.

For example, as shown in Figure 7-20:

a, if the data type of each item is "16-bit positive integer", then "total bytes of data types occupied by all data items " = 2 +2 +2 +2 +2; that "all data items (20 + 10) * 2, which is equal to 60; the bytes number of "16-bit data " is 2 while "32-bit data occupies 4 bytes.

b. If the data type selected for "LH1" and "LH3" is "32-bit positive integer" (when the data type selected for "LH1" and "LH3" is "32-bit positive integer", the address of the "Data item" displays as LH0, LH1, LH3, LH5, LH6), then "the total number of bytes occupied by all data items" = 2 + 4 + 4 + 2 + 2= 14; then the required memory = (20 + 14) * 2= 68.

Read Address: Start address to read data (Refer to Data Item for detailed information)

Sampling method:

• Timing: Read data at set interval.

• Trigger: Read data when the value of the trigger address is 1.

• Clock: Start reading data at a user-set time interval.

Full sampling disposal manner: Disposal when the data sampling reaches the maximum total number.

Stop sampling: Stop data sampling in case of full sampling; if this option is not selected, the new read data will replace the oldest one.

Notify: Inform an address whose value is 1 when the sampling number reaches the maximum.

Click the "Data Item" tab, set the page properties, as shown in Figure 7-19:

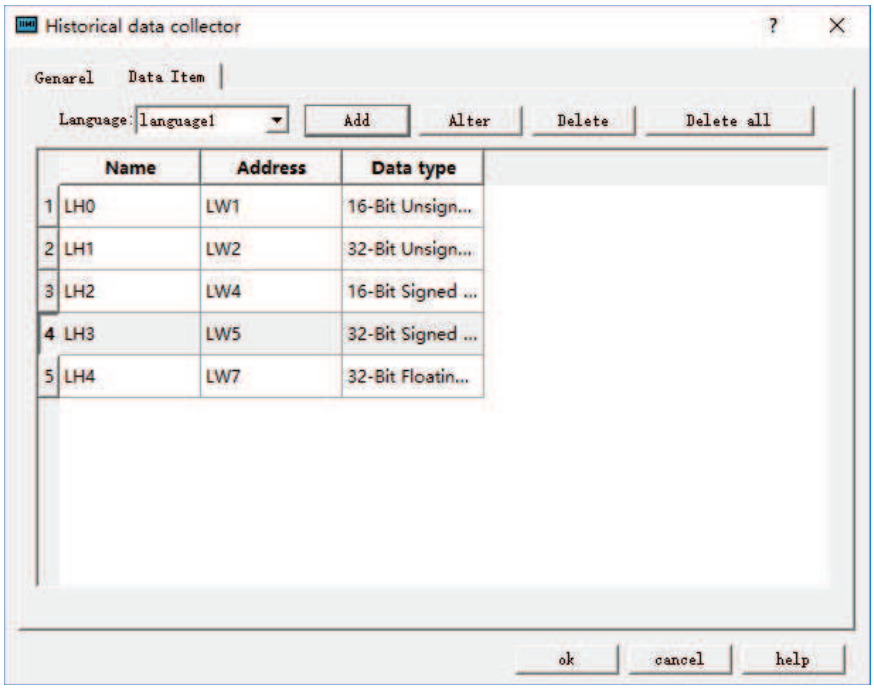


Figure 7-19Data Item of Historical Data Collector

If multi-language is selected, it is necessary to set names for each language when adding new data items; otherwise it cannot add new items.

In HMITOOL, the address of each group of historical data can be set discrete, according to the actual situation of LW register, for example: S1 is LW1, S2 is LW89, S3 is LW8905.

Click on any item in the list box to set the properties.

An example in detail:

Set the read address to LW1 on the General page, the register address is set consecutively and the sampling length is 5; click the Data Item tab and add 5 columns of data (LH0-LH4) in the list. Select each to set the properties such as name, address, data type, total digits decimal places and whether to scale.

Assuming that the names, data type and scaling of LH0-LH4 are set as shown in Figure 7-20; other attributes are default values:

LH0-16 bit Positive integer, Scaling, with a gain of 3 and an offset of 2

LH1-32 bit positive integer, Scaling, with a gain of 2 and an offset of 1

LH2-16-bit integer, Scaling is not selected

LH3-32-bit integers, Scaling is not selected

LH4-Floating point, Scaling, with a gain of 4 and an offset of 5

After the settings, since the sample length is 5 (set in General page), it reads out 5 address values consecutively starting with "Read Address". Addresses are read according to the data type of each item.

If select 16-bit data, the extracted address is continuous. If 32-bit data is selected, the fetched addresses are separated. If the scaling function is selected, the final read value is equal to the value of read address multiplied by the gain plus the offset.

Tips:

Note: The gain value in scaling must be greater than 1!

Set up each data type and whether to scale; the addresses to be read are as follows:

LH0-LW1 (16-bit data, read address + 1) Final value of LH0 = Address value of LW1 × 3 + 2

LH1-LW2 (32-bit data, read address +2) Final value of LH1 = Address value of LW2 × 2 + 1

LH2-LW4 (16-bit data, read address +1) Final value of LH2 = Address value of LW4

LH3-LW5 (32-bit data, read address +2) Final value of LH3 = Address value of LW5

LH4-LW7 (32-bit data, read address +2) Final value of LH4 = Address value of LW7 × 4 + 5

After settings, click "OK". Click "Historical data display" icon on the toolbar; set the row number to 5 in the pop-up dialog box, the other attribute values are by default.

Click "OK" to generate a table automatically in the screen, as shown in Figure 7-20:

Time	Date	LH0	LH1	LH2	LH3	LH4

Figure 7-20 Data Table of Historical Data Display

Application of the above table:

- Date: The date on which the address value is read
- Time: The time when to read the address value
- "LH0": Display the value of the LW1 at a certain time.
- "LH1": Display the value of the LW2 at a certain time.

• "LH2": Display the value of the LW4 at a certain time.

• "LH3": Display the value of the LW5 address at a certain time.

• "LH4": Display the value of the LW7 address at a certain time.

For example, the read address is LW1, the user-defined data sampling length is 3, the total number of samples is 10, the sampling method is Trigger, the time interval is 1 second, and the data type of each data is the default;

1). The option of Stop Sampling is not selected:

The LW1, LW2, LW3 address values are read every 1 second. Since the total number of samples is 10, 10 sets of data are read out at the tenth second. Because the "Stop sampling" option is not selected, the process continues. A fixed memory area depending on the sampling length, total number of samplings and data type is allocated to store the read data, so when the number of data groups has reached the "total number of samplings," the latest read data will replace the first one.

2). The option of Stop Sampling option is selected:

The only difference is that it stops sampling when achieve the total number of samplings.

Tips:

Employ “Historical Trends”or “Historical Data Display”to read the values more intuitively.

9.6 Alarm settings

HMITOOL alarm settings consist of digital alarm login and analog alarm login. Users can view the alarm type and the occurrence time through Alarm control or Alarm Bar after setting up the alarm login.

Contents：

9.6.1 Digital Alarm Login

Alarm Setting is employed to display alarm information, only with which the alarm control and alarm bar run normally (in fact, the alarm control and alarm bar displays the alarm message of "digital alarm login" and "analog alarm login").

Right click the "Alarm landing" in the Project Manager and select "New digital alarm log", as shown in Figure 7-21:

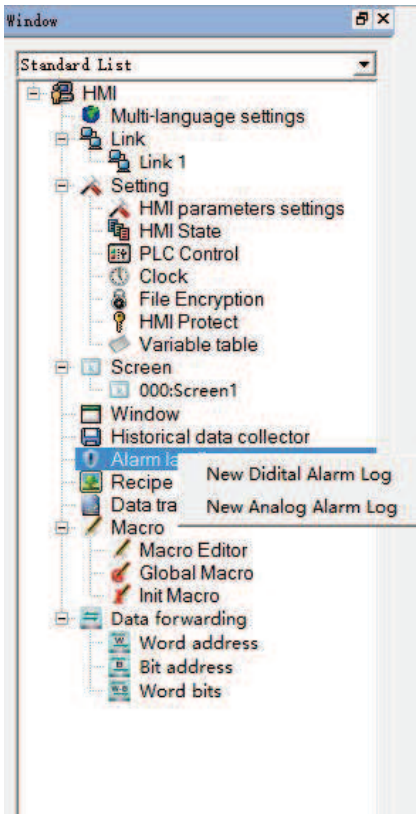


Figure 7-21 Select alarm log

Double click the "digital alarm login" option to open the digital alarm settings dialog box, as shown in Figure 7-22:

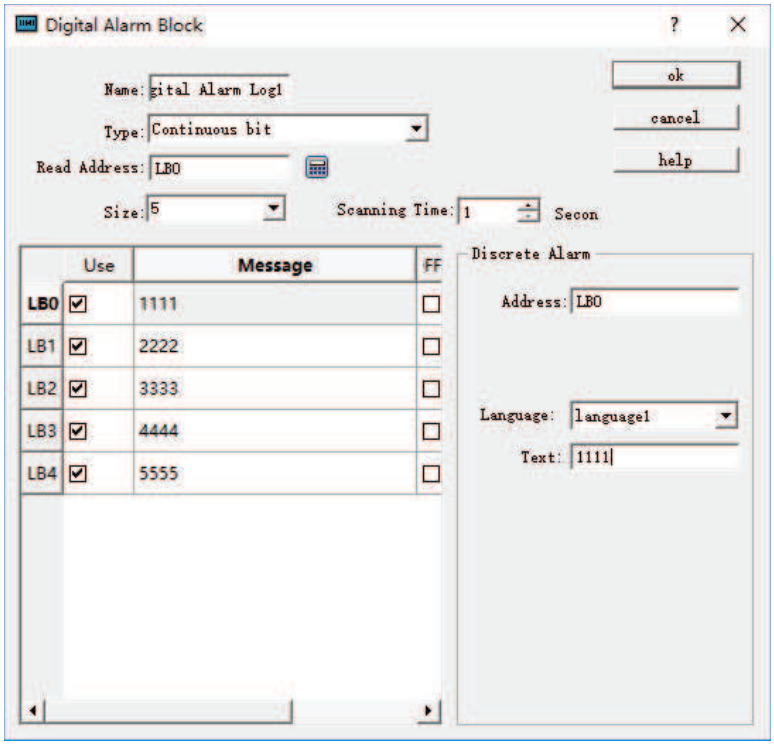


Figure 7-22 Select continuous digit

Steps to build a digital alarm log are as follows:

- Enter the read address first; it is assumed here that the internal address LB1 is set.
- In the "Size" column, select the total number of alarms, i.e. the total number of bit addresses, which is continuous. It is assumed here that the set value is six.
- Set the bit alarm “Scanning time", that is, the scanning frequency.
- Click an item in the list box, and then move the mouse to the right of the "Message" module; input text information in the “Text” edit box serving as the alarm content. Here assume that the alarm messages are as shown in Figure 7-23.
- If "Use" is selected, the alarm information will be displayed in "Alarm Control" or "Alarm Bar". Otherwise the alarm message will not appear even if the corresponding address is 1. If the address value of this item is 0, the alarm information does not show.
- OFF Alarm: When this option is selected, the alarm will be generated when the corresponding address value is 0 (OFF value). Otherwise, an alarm will be generated when the address value is 1 (ON value). By default, an alarm occurs when the address value is 1 (ON).

Here is an example: Suppose the choice of type is "Continuous bit".

- If the value of LB1 address is 0, the alarm information will not be displayed in "Alarm Control" and "Alarm Bar", regardless of whether the "Use" option is selected or not.
- If the value of the LB1 address is 1 and the "Use" option is not selected, the alarm information is not displayed in the "Alarm Control" and "Alarm Bar".
- If the value of LB1 is 1 and the "Use" option is selected, the alarm information will be displayed in "Alarm Control" and "Alarm Bar".

Click "OK" to complete digital alarm settings. Users can add 6 bit buttons with the function of Alternation whose address is LB1-LB6; place controls of "Alarm" and “Alarm Bar" on the screen; then execute offline simulation to check whether it displays alarm information.

The following is an example of offline simulation as shown in Figure 7-23:

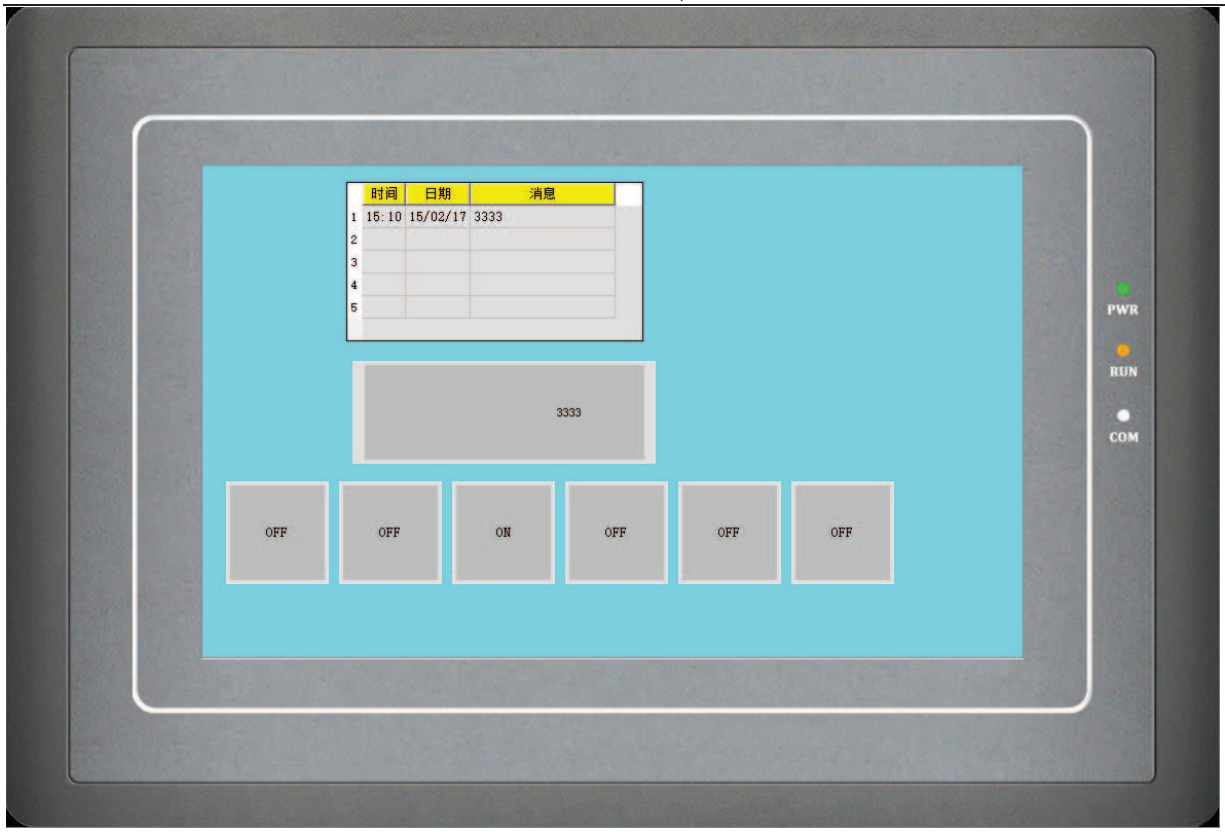


Figure 7-23 Offline simulation example of digit alarm

When "Continuous Bit of Word" is selected as the Type, as shown in Figure 7-24 below:

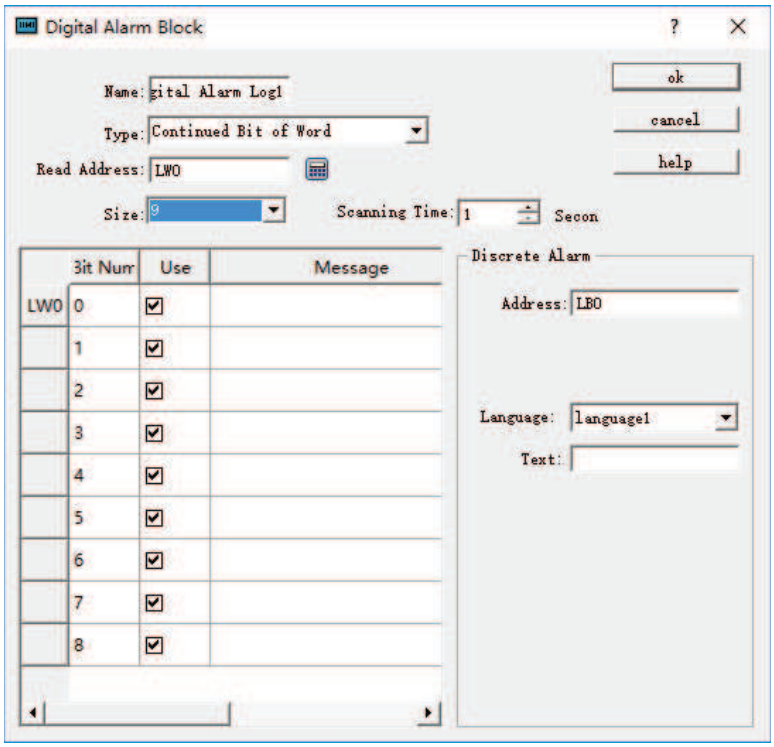


Figure 7-24 Select Continuous Bit of Word

If the “Use” of a certain bit number is selected, an alarm will be generated when the value of this bit value is 1.

Tips:

Note: When copying the contents of alarm messages from other documents to the alarm text of digital alarm log and analog alarm log, please make sure that there is no line break (invisible) in the message. It is recommended to input the message manually rather than copy and paste, in case of display error of alarm message.

If "Word value" is selected for the type, as shown in Figure 7-25 below:

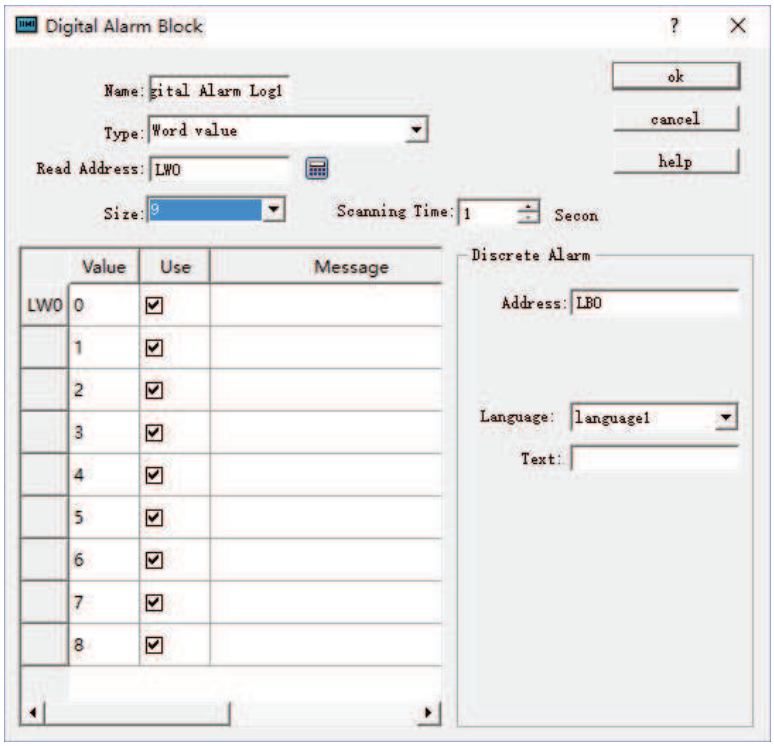


Figure 7-25 The "Word Value" digital alarm dialog box

When a value of the address is selected, the alarm is generated when the value of the address is equal to the value selected in the list.

Place an Alarm control with 7 lines as the example in Figure 7-24 and a Dynamic alarm bar whose attributes are set as shown in Figure 7- 26:

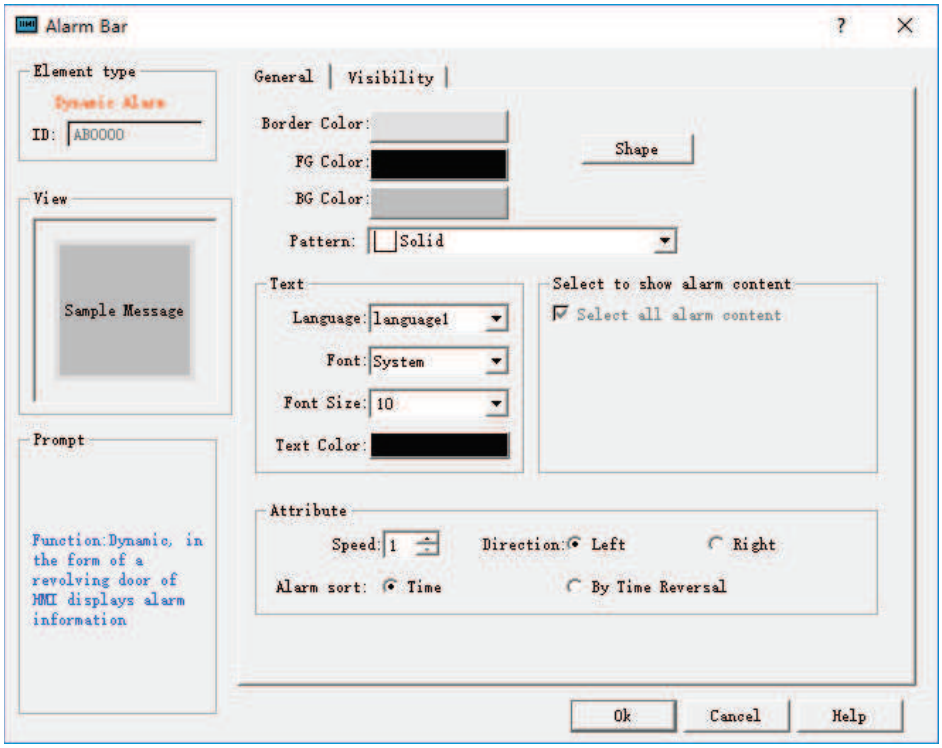


Figure 7-26 Dynamic Alarm Bar Properties dialog box

Add lastly six bit buttons whose write address is LB1-LB6 with the "Alternation", “Monitor” and “Monitor Address identical to Write Address” functions set for each. After the settings, click the "Save" button on the toolbar to save the project, execute the "Offline" command in the "Download" item in the menu bar, and then click the bit buttons LB1, LB2, LB4 and LB6. In this case, the Alarm control and Dynamic Alarm Bar will display the alarm information of LB1, LB2, LB4, and LB6 in the "Digital Alarm" When click them again, "Alarm Control" and "Dynamic Alarm Bar" will display nothing since their values are 0 because of the alternation function.

Tips:

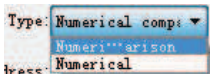
Note: Different alarm information is available for corresponding language through the function of switching language.

Note: Number of digital alarm is up to 10 for VS series HMI.

9.6.2 Analog Alarm Login

Open the analog alarm settings dialog box, as shown in Figure 7-27:

It has two types: Numerical value and Numerical comparison.



The numerical comparison alarm is as shown below:

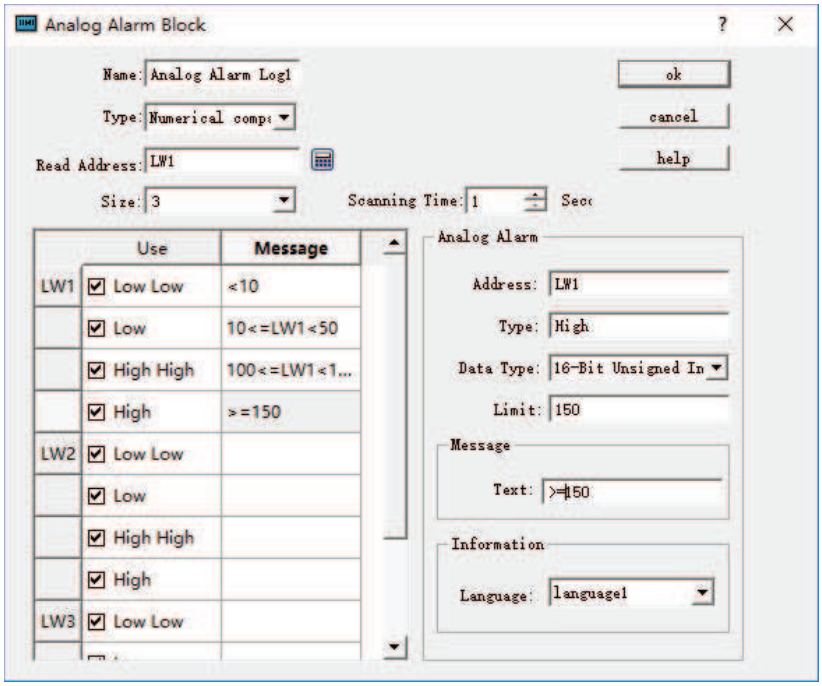


Figure 7-27 Analog Alarm Login dialog box

If user needs to use the analog alarm, steps are as follows:

- First input the address in the "Read Address" column to alarm. It is assumed here that the internal address LW1 is set.
- Select the total number of alarms in the "Size" option, which means the total number of addresses; data type decides the way to read address. Each address value is set firstly as the 16-bit data type and it is continuous. But the data type can be changed. As shown in Figure 7-28, the LW1 address chooses a 32-bit positive integer while LW3 and LW4 values are 16-bit positive integers. Therefore, when 32-bit data is selected for an address, the later address value to be read is added by 2 on the basis of the former address value; and if the 16-bit data type is selected, the later address value to be read is the former value plus 1.
- Users can set the "Scanning time" of the alarm, which means the scanning frequency.
- Click each item to input a value in the “Limit” box on the right; then enter the text information (alarm content) in the “Text" box. Here assume that the message is as shown in Figure 7-28: each address value can be classified into one of four ranges-- Low, Low Low, High, High High.

As shown in Figure 7-28:

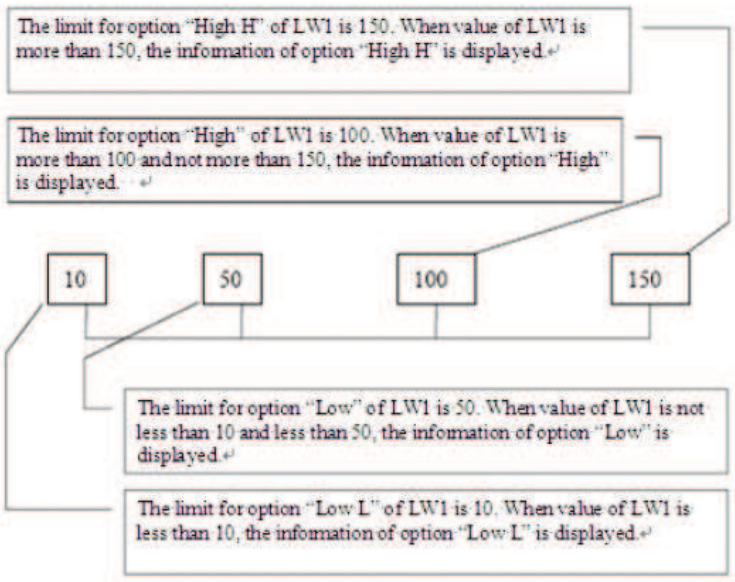


Figure 7-28 Diagram of the four address values for the LW1 address

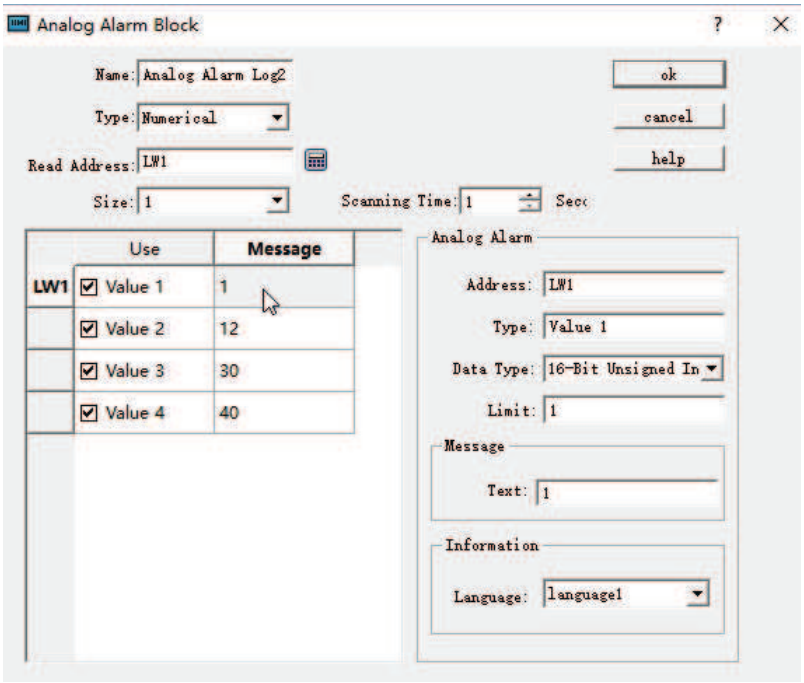
- Low Low: When the value of this item is less than the limit of this setting and the "Use" option is selected, the alarm information of this item will be displayed in "Alarm Control" or "Dynamic Alarm Bar";

- Low: When the value of this item is no less than the “Low Low” limit while less than the “Low” limit and the "Use" option is selected, the alarm information of this item will be displayed in "Alarm Control" or "Dynamic Alarm Bar";
- High: When the value of this item is greater than the “High” limit while no greater than the “High High” limit and the "Use" option is selected, the alarm information of this item will be displayed in "Alarm Control" or "Dynamic Alarm Bar".
- High High: When the value of this item is greater than the limit of this setting and the "Use" option is selected, the alarm information of this item is displayed in "Alarm Control" or "Dynamic Alarm Bar".

Refer to “Digital Alarm Login” for the application of “Use”.

Numerical value alarm means an alarm occurs when the value reaches the set limit value.

The settings are as follows: Alarm is triggered when the alarm value reaches 1, 12, 30 and 40.



Tips:

Note: Number of analog alarm is up to 10 for VS series HMI.

9.6.3 Alarm Display Controls

It can add alarm display control after the settings: Alarm Display, Alarm Bar and Historical Alarm Display.

Select in the toolbar menu to set the alarm bar display parameters in the dialog box, as shown in Figure 7-29:

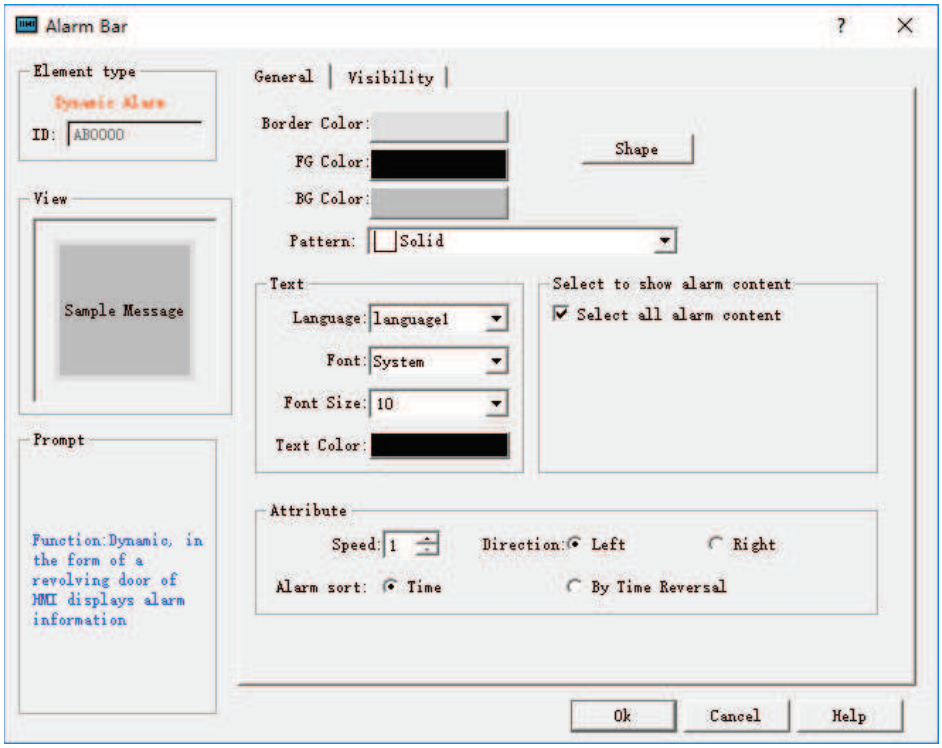


Figure 7-29 Alarm Bar dialog box

- Speed: the movement speed of the alarm information;
- Direction: The alarm bar moves from right to left or left to right;
- Alarm sort: The display order of alarms by chronological order or by traverse chronological order.

Select the alarm control in the toolbar menu to set the alarm display text and parameters in the dialog box, as shown in Figure 7-30:

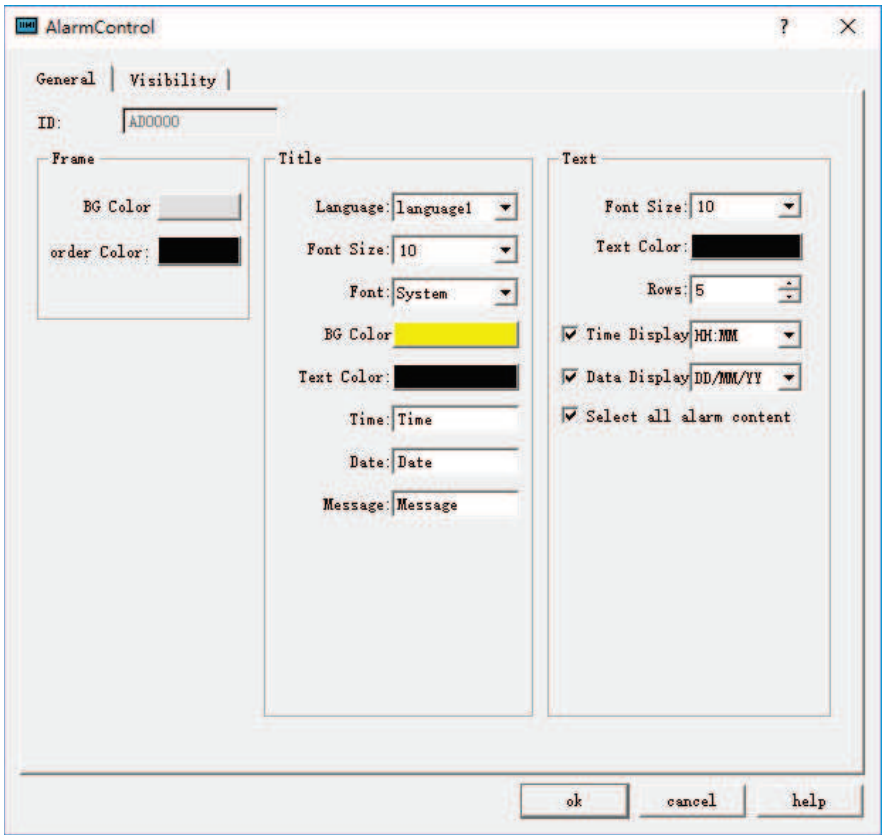
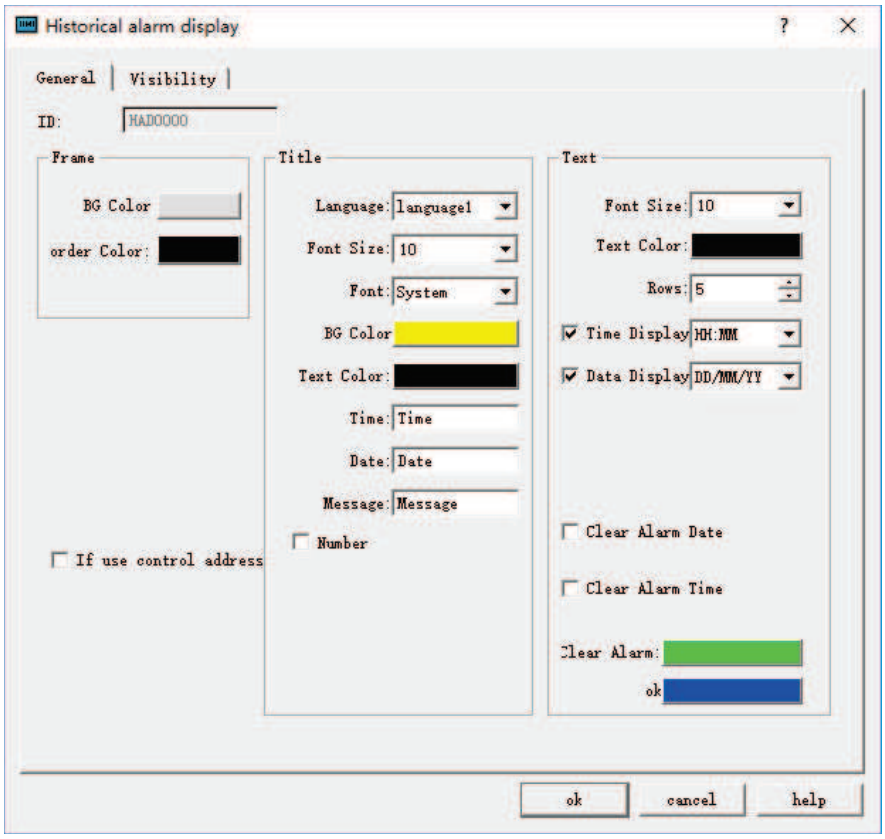


Figure 7-30 Alarm Display dialog box

Select the historical alarm display control in the toolbar menu to set the display content and parameters in the dialog box, as shown below:



Historical Alarm Display dialog box

9.7 Recipe

In the field of manufacturing, recipe is used to describe the proportions of materials used to produce a product, and is a set of parameter settings of variables used during production. For example, a basic recipe is necessary for bread making, and this recipe may list the weights or proportions of all materials used to make bread, such as water, flour, sugar, egg, oil, and so on. In addition, it may also list some optional materials such as fruit, nut, chocolate, and so on. These optional materials may be added to the basic recipe to make bread of different flavors. Take iron works for another example. A recipe in an iron works may be a set of machine parameters. For batch processing machines, a recipe may be used to describe different steps of the batch processing.

A machine may make both bread and cakes, including bread of different flavors and cakes of various forms. Here, we call the material proportions of bread as a recipe, and call the different material proportions of different flavors of bread as a file. Obviously, recipe records are contained in recipe files.

HMITOOL provides the function of recipe configuration that is available in the Recipe of the menu bar or in the Project Manager, you can enter the recipe configuration dialog box, add a new set of recipe functions. As shown in Figure 7-31:

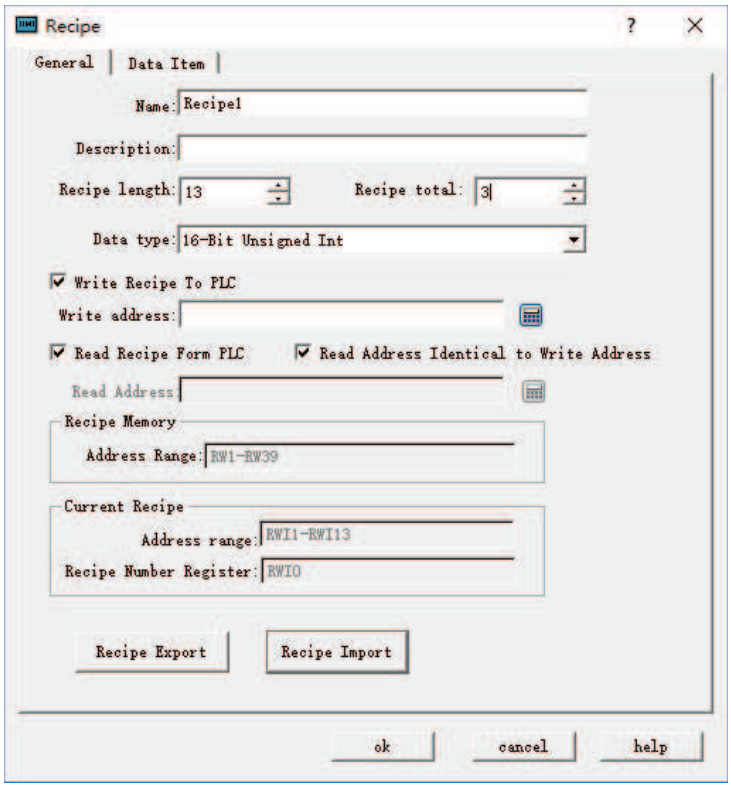


Figure 7-31 Recipe Property Settings Dialog Box

- Name: Name of the current recipe file.
- Description: Description: Name description of the current recipe file;
- Recipe length: Set the length and quantity of the recording address to be read by the current recipe. Addresses of recipe record form are arranged in sequence automatically by the system, and cannot be changed. The maximum length of recipe is 4096.
- Total Number: Number of recipes, up to 512;

Tips:

Note: Recipe length * Bytes occupied by data type * Total number <= 512KB

- Data Type: Data type of the data register;
- Write recipe to PLC: Set related data register address of recipe and PLC;
- Read recipe from PLC: Set related data register address of recipe and PLC;
- Recipe memory: The memory address range of the recipe in the HMI that is automatically generated and cannot be changed;
- Current recipe: The recipe memory address in the HMI and the data register of PLC is automatically generated by order and cannot be changed;
- Recipe Number Register: the recipe is arranged automatically by the order of serial number, and it cannot be changed; it corresponds the order of recipe name in the data item.

The recipe data item, as shown in Figure 7-32:

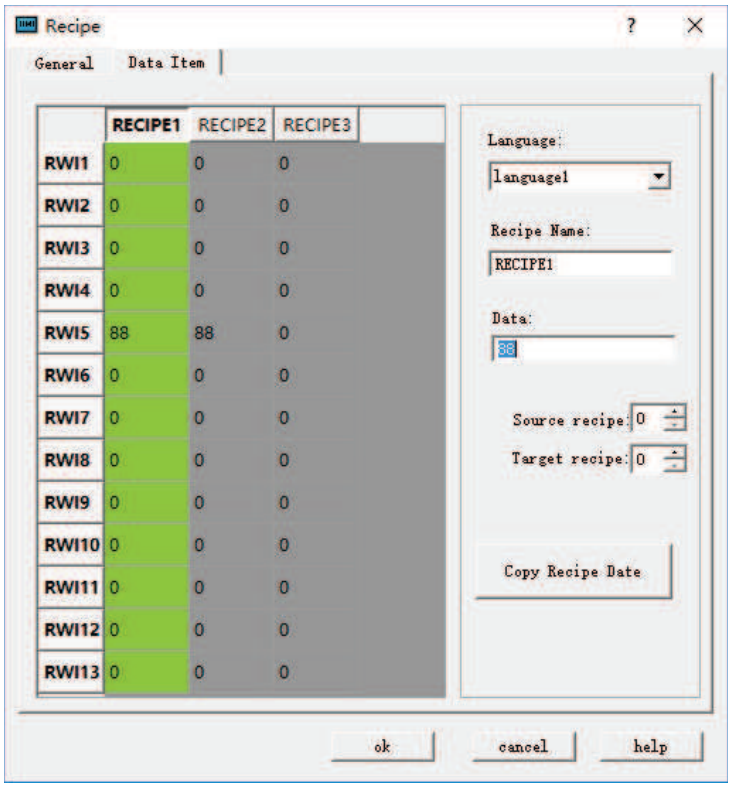


Figure 7-32 Recipe Data Item Page

- Name: The name of the current recipe;
- Data: Write and display each address value of the current recipe;
- Copy Recipe Data: Copy the data from the source recipe to the destination recipe. Click OK to save the current one and click Cancel to exit the dialog box. Once the recipe configuration has been completed, the interface can be designed to operate the recipe in the HMI.

Application example of Recipe:

1. Create a recipe configuration as described above. The effect is shown in Figure 7-33:

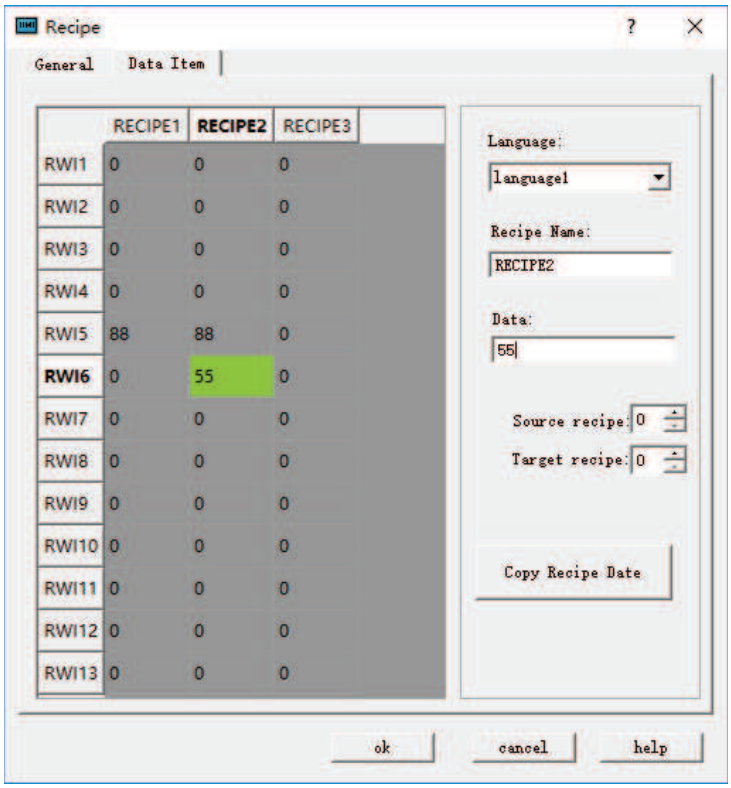


Figure 7-33 Function of recipe data copy

This recipe has a total of 4 sub-recipes, each containing 16 items. The storage addresses are from RWI: 1 to RWI: 16 in the memory of HMI.

2. After the recipe configuration, display the recipe on the screen， modify it or directly download it to the PLC. So we will use the following essential functional controls.

- Recipe display: Click numeric display / input (refer to "Numerical display / input")

  Select the internal memory address RWI: 0 to create a numerical display and input control on the screen. The input and display value is the recipe serial number. The value 0 of RWI0 means recipe serial number 1; the value 1 means recipe number 2 , etc.

- Click the function button F1: Refer to the "Function button" for operation.

Buttons can be created to write recipe to PLC, read recipe from PLC, save recipe, select the previous recipe and the later one through the function button with relevant function setting.

Application of multiple recipes: HMITOOL provides new function of multi-group recipe, but it supports only ten groups, otherwise a limit message will be prompted as shown in 7-34-1

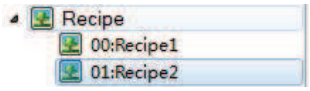


Figure 7-34-1

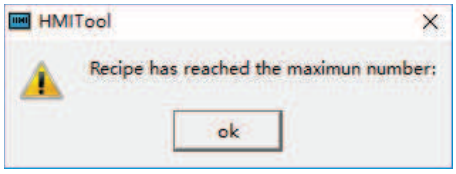


Figure 7-34-2

The LW61141 is employed to enter the recipe group. The original RWI0 is also used to input the recipe number. Therefore, it needs to set values of LW61141 and RWI0 at the same time when select one. For example: Enter recipe group 0 to LW61141, serial number 2 for RWI0; it means selecting the recipe data of Recipe 2 in recipe group 0.

3. According to the above operation, a screen can be created as shown in Figure 7-35:

Assume that all monitor address to be written is D0, the length of the recipe is 16 and the total number is 4.

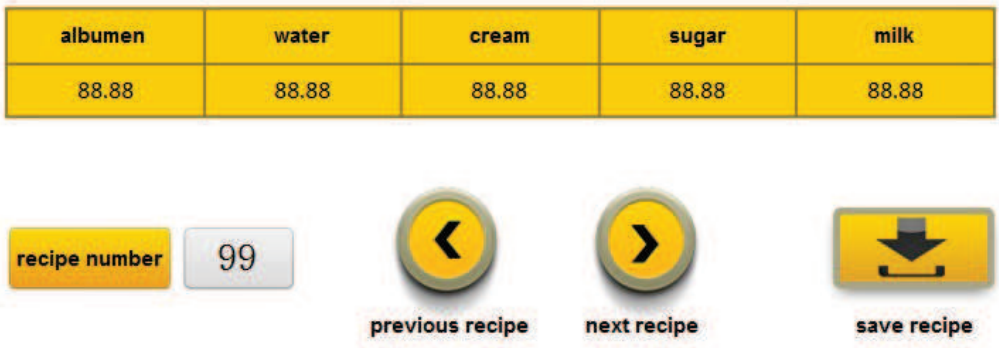


Figure 7-35 Effect of the Recipe

It is easily available to modify, preserve, read and write recipe through the function buttons and numerical display / input settings. But it should pay attention to the following points:

- Recipe list should be established before use of Recipe functions.
- When writing recipe parameters into the recipe list, pay attention to the data type. 16-bit datum occupies only one word, and 32-bit datum occupies two words. Consistent data type should be chosen. Particularly, pay special attention to use of 32-bit data address. As 32-bit datum occupies two words, prevent address overlapping when entering write-in address and monitored address.

- Address RWI0 is for fixed use, and can be used to change recipe number only. If value of RWI0 is 0, it indicates the first recipe number, and so on.
- Recipe parameter addresses are continuous.

9.8 Data Transmission

Used to transmit data between PLC and HMI or between PLC and PLC. Click "Data Transmission" in the Project Manager to show the current data transmission list.

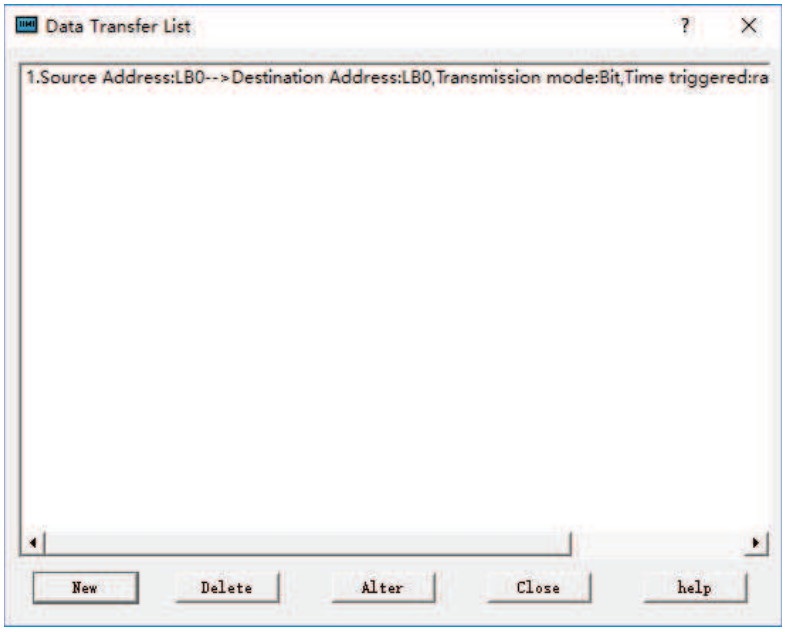


Figure 7-36

The transmission list lists details of all the current transmission items. Users can create, delete and modify them. HMITOOL configuration software supports up to 512 data transmission items.

To create a new one or modify an existing transmission item, it needs to enter the Property Setting page:

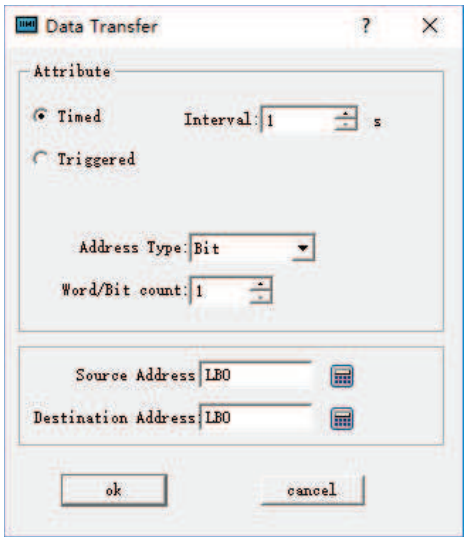


Figure 7-37

1. Attribute

Timing / Trigger: Select the trigger mode for data transfer.

Timing: Perform data transfer at the set interval.

Triggering: The transmission signal is controlled by specified address. Execute this command when the address value is 1.

Automatic reset: When the value of address trigger is 1, set it automatically to 0.

Address Type: Select the address type to transfer data: Bit / Word / Double Word.

Word / Bit: Input the length of the data to be transferred. The length range is 1~ 64.

2. Address

Source Address: The source address of the transmitted data, which must be of the same type as the type of address set in the "Property".

Destination Address: The destination address of the transmitted data, which must be of the same type as the type of address set in the "Property".

Destination address and Source address cannot be on the same PLC.

9.9 Global Macro

Set a macro that has been successfully compiled into a global macro, meaning executing this macro when the configuration starts running, and it will be executed during the run time without being limited by screens. A maximum of 64 global macros can be added, and they are executed in the set order.

Click "Global Macro" on the menu "Settings", as shown in the dialog box:

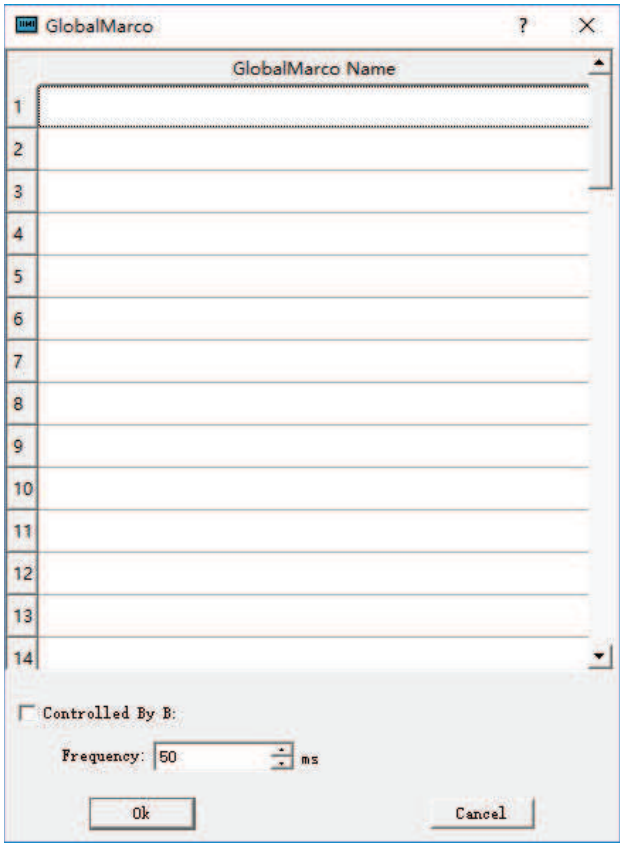


Figure 7-38

Click the list item to list all the macro names that have been compiled successfully, and then decide whether the macro is controlled by bit and the execution frequency. The selected macro is executed as a global macro during configuration run time.

9.10 Init Macro

Set a macro that has been successfully compiled into an Init macro, meaning executing this macro only one time without being limited by screens. A maximum of 64 init macros can be added, and they are executed in the set order.

Click "Init Macro" on the menu "Setting", as shown in the dialog box:

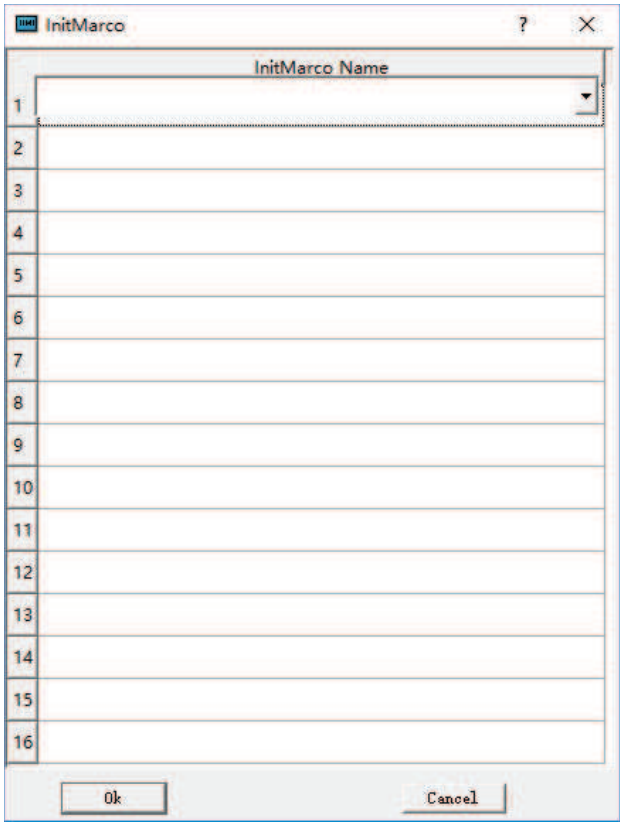


Figure 7-39

Click the list item to list all the macro names that have been compiled successfully. The selected macro is executed as a init macro during configuration run time.

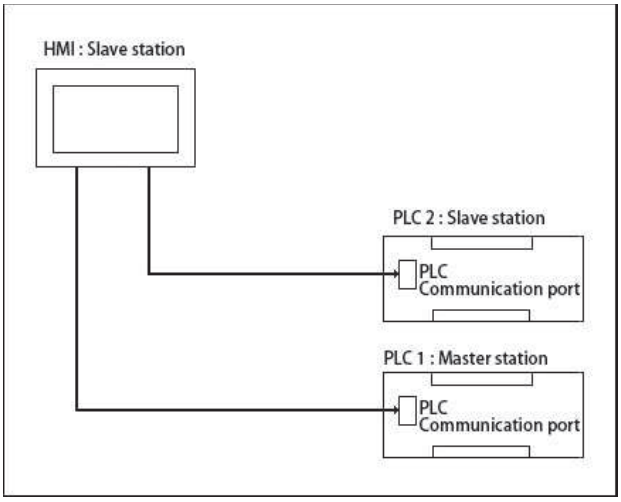
9.11 Data forwarding

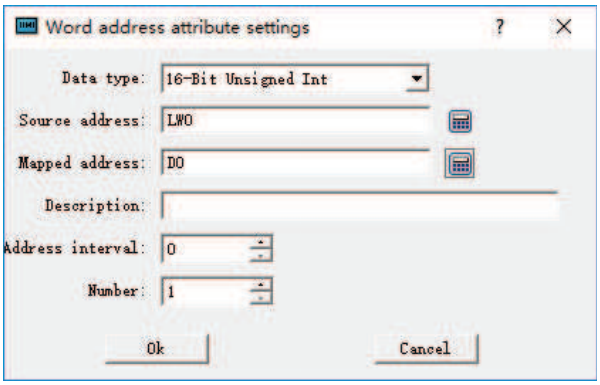
Function of Data Forwarding:

1、Data Forwarding apply to transfer data between devices connected to two ports of HMI.

Example：

Select Modbus RTU Slave protocol for Port 1; Take HMI as slave station; master station Device 1 read the address of LW0 in HMI.
Select Modbus RTU Master protocol for Port 2; HMI reads the address of 4x11 of Slave station 2.





10. Reserved Registers of HMITOOL System

HMITOO reserves some registers for special use, and users must refer to the related instructions when using these registers, including LB (Local Memory Bit); LW (Local Memory Register) and RWI (Formula Index).

For example:

LB: LB0-LB65535 and LB50000-LB51999 belong to the area of power failure protection;

LW: LW0-LW65535 and LW50000-LW51993 belong to the area of power failure protection. And LW51994~LW51999 are used by interior, not external;

RWI: RWI0~RWI65535.

Contents：

10.1 LB

Register Address	Meaning	Note	Read/Write
60000-60099	Initialization Settings ON	Enable the initialization ON when system starts	R/W
60100	Download Formula	Write the data of this RWI to device if it is triggered; reset after execution	R/W
60101	Upload Formula	Read the formula data of device to RWI register if it is triggered; reset after execution	R/W
60102	Preserve Formula	Write Formula from RWI to FLASH if it is triggered; reset after execution	R/W
60103	Formula Downloading Indication	ON if formula is being written to device; OFF after the end of downloading	R
60104	Formula Uploading Indication	ON if formula is being read to device; OFF after the end of uploading	R
60105	Restart HMI	Restart system if it is triggered	R
60106	Backlight Control	Close Backlight when it is ON; Open Backlight when it is OFF	R/W
60107	Communication State of COM1	ON means normal communication; OFF means communication failure or no communication	R
60108	Communication State of COM2	ON means normal communication; OFF means communication failure or no communication	R
60109	Modification preservation of COM1 communication parameters	Write modified communication parameters to FLASH if it is triggered; OFF after execution	R/W
60110	Modification preservation of COM2 communication parameters	Write modified communication parameters to FLASH if it is triggered; OFF after execution	R/W
60111	Modification confirmation of system parameters	Write modified communication parameters to FLASH if it is triggered; OFF after execution	R/W
60112	Abnormal communication window	ON means permission of exception window popping up; OFF means prohibition	R/W
60113	Mouse cursor control	ON means disolay of mouse censor; OFF means prohibition (Only available for touch)	R/W
60114	Connection state of USB	ON means with USB device connection; OFF means no connected USB device	R
60115	Coonection state of SD card	ON means with SD card connection; OFF means no connected SD card	R
60116	Touch Sound	Whether enable sound when touch or mouse click	R/W
60117	Alarm Sound	Whether enable alarm sound when alarm occurs	R/W
60118	Clear historical alarm record	Clear historical alarm record (Reset automatically)	R/W
60119	Clear operation record	Clear operation record (Reset automatically)	R/W
60120	Clear historical data	Clear historical data (Reset automatically)	R/W
60121	Reload Formula	Reload the formula data of the current one while it is ON; OFF when the reloading ends	R/W
60123	Save historical alarm	Save historical alarm record to disk when it is ON; OFF after the preservation	R/W
60124	Save historical data	Save historical data to disk when it is ON; OFF after the preservation	R/W
60125	Send historical alarm file to SD card	Send historical alarm file to SD card; OFF automatically after the sending	R/W
60126	Send historical data file to SD card	Send historical data file to SD card; OFF automatically after the sending	R/W
60127	Send historical alarm file to U disk	Send historical alarm file to U disk; OFF automatically after the sending	R/W
60128	Send historical data file to U disk	Send historical data file to U disk; OFF automatically after the sending	R/W
60129	Send the current project to U disk	Send the current project to U disk; OFF automatically after the sending	R/W
60130	Send the current project to SD card	Send the current project to SD card; OFF automatically after the sending	R/W
60133	User Logout	Cancel User Login when it is ON; Reset automatically after the cancelation	R/W
60141	Whether there is a real-time alarm	ON when a real-time alarm occurs	R
60142	Screen saver state of system	ON when it is under screen saver stat	R
60143	Copy operation record to U disk	Send operation record to U disk in case of ON; OFF automatically after the sending	R/W
60144	Copy operation record to SD card	Send operation record to SD card in case of ON; OFF automatically after the sending	R/W
60145	Copy all formulas to U disk	Send formulas to U disk in case of ON; OFF automatically after the sending	W

60146	Copy formulas to SD card	Send formula files to SD card incase of ON; OFF automatically after the sending	W
60147	End mark of Qrcode scanning USB	Triggered in case of ON	R/W
60148	Import historical trigger bit from U disk	Triggered in case of ON; reset automatically. Historical data file must be histdata.csv; otherwise the execution is invalid.	R/W
60149	Import historical trigger bit from SD card	Triggered in case of ON; reset automatically	R/W
60150	Export specific ID formula to U disk trigger bit	Triggered in case of ON; reset automatically	R/W
60151	Export specific ID formula to SD card trigger bit	Triggered in case of ON; reset automatically	R/W
60152	Export specific ID formula from U disk to HMI trigger bit	Triggered in case of ON; reset automatically	R/W
60153	Export specific ID formula from SD card to HMI trigger bit	Triggered in case of ON; reset automatically	R/W
60154	Clear trigger bit of specific ID formula trigger bit	Triggered in case of ON; reset automatically	R/W
60155	Cover formula trigger bit	Triggered in case of ON; reset automatically	R/W
60156	Read formula and save trigger bit	Triggered in case of ON; reset automatically	R/W
60163	Export formulas to U disk by groups		R/W
60164	Export formulas to SD card by groups		R/W

10.2 LW

Register Address	Meaning	Note	Read/Write
60000	Local time: second	Bcd code; effective value range: 0-59	R/W
60001	Local time: minute	Bcd code; effective value range: 0-59	R/W
60002	Local time: hour	Bcd code; effective value range: 0-23	R/W
60003	Local time: day	Bcd code; effective value range: 1-31	R/W
60004	Local time: month	Bcd code; effective value range: 1-12	R/W
60005	Local time: year	Bcd code; effective value range: 0-9999	R/W
60006	Local time: week	Bcd code; effective value range: 1-7	R
60007	System running hours	Hours of system running time	R
60008	System running minutes	Minutes of system running time	R
60009	System running seconds	Seconds of system running time	R
60010	Serial number of the current screen window	Serial number of the current screen	R
60012	Current language	Corresponding language serial number	R
60013	Upper limit of value input	Upper limit of allowable input in user-defined keyboard; ASC display	R
60023	Lower limit of value input	Lower limit of allowable input in user-defined keyboard; ASC display	R
60033	Data contents shown in keyboard	Data contents shown in keyboard; characters entered via keyboard; ASC display	R
60065	Current Logged in User	Current user serial number	R
60066	Current Logged in User Password	Display the current user password; 16 characters; ASC display	R
60074	Current logged in User name	Display the current user name; 32 characters; ASC display	R
60090	Whether enable User 1	0-inactivate; 1-activate	R
60091	Whether enable User 2	0-inactivate; 1-activate	R
60092	Whether enable User 3	0-inactivate; 1-activate	R
60093	Whether enable User 4	0-inactivate; 1-activate	R
60094	Whether enable User 5	0-inactivate; 1-activatev	R
60095	Whether enable User 6	0-inactivate; 1-activate	R
60096	Whether enable User 7	0-inactivate; 1-activate	R
60097	Whether enable User 8	0-inactivate; 1-activate	R
60098	Name of User 1	32 characters; ASC display	R
60114	Name of User 2	32 characters; ASC display	R
60130	Name of User 3	32 characters; ASC display	R
60146	Name of User 4	32 characters; ASC display	R
60162	Name of User 5	32 characters; ASC display	R
60178	Name of User 6	32 characters; ASC display	R
60194	Name of User 7	32 characters; ASC display	R
60210	Name of User 8	32 characters; ASC display	R
60226	Password of User 1	16 characters; ASC display	R
60234	Password of User 2	16 characters; ASC display	R
60242	Password of User 3	16 characters; ASC display	R
60250	Password of User 4	16 characters; ASC display	R
60258	Password of User 5	16 characters; ASC display	R
60266	Password of User 6	16 characters; ASC display	R
60274	Password of User 7	16 characters; ASC display	R
60282	Password of User 8	16 characters; ASC display	R
60290	Screen saver time	Display specific screen saver time; setting range: 0-60	R/W
60291	Flicker circle of Indicator light	Flicker circle of Indicator light; the minimum value is 0.1s	R/W
60292	X position in case of touch	Position of X coordinate in case of touch	R

60293	Y position in case of touch	Position of Y coordinate in case of touch	R
60294	X position at the end of touch	Position of X coordinate at the end of touch	R
60295	Y position at the end of touch	Position of Y coordinate at the end of touch	R
60296	Touch state	1-Touch; 0-Release	R
60297	Checking of COM1	Communication parameters, checking: 0-NONE, 1-EVEN, 2-ODD	R/W
60298	Baud rate of COM1	Communication parameters, baud rate: 0-1200,1-2400,2-4800,3-9600,4-19200,5-38400,6-57600,7-115200	R/W
60299	Stop Bit of COM1	Communication parameters, stop bit: 0-1stops,1-2stops	R/W
60300	Data Length of COM1	Communication parameters, data length: 0-7bits; 1-8bits	R/W
60301	HMI Address of COM1	Communication parameters, HMI address	R/W
60302	PLC Address of COM1	Communication parameters, PLC address	R/W
60303	Continuous address interval of PLC to COM1	The maximum length of contiguous address that can be read by a single communication	R/W
60304	Communication time of COM1	Delay time of communication	R/W
60305	Retry number of COM1	Number of retries in case of abnormal system	R/W
60306	Address Mode of COM1	Address mode: 0-standard mode; 1-extended mode	R
60307	Time of COM1 timeout	Set communication timeout; unit: ms	R/W
60308	Current waiting time of COM1 communication	Current waiting time of communication; unit : ms	R/W
60309	HMI Site of COM1	0-Machine; 1-Far-end	R/W
60312	Checking of COM2	Communication parameters, checking: 0-NONE, 1-EVEN, 2-ODD	R/W
60313	Baud rate of COM2	Communication parameters, baud rate: 0-1200,1-2400,2-4800,3-9600,4-19200,5-38400,6-57600,7-115200	R/W
60314	Stop Bit of COM2	Communication parameters, stop bit: 0-1stops,1-2stops	R/W
60315	Data Length of COM2	Communication parameters, data length: 0-7bits; 1-8bits	R/W
60316	HMI Address of COM2	Communication parameters, HMI address	R/W
60317	PLC Address of COM2	Communication parameters, PLC address	R/W
60318	Continuous address interval of PLC to COM2	The maximum length of contiguous address that can be read by a single communication	R/W
60319	Communication time of COM2	Communication delay time	R/W
60320	Retry number of COM2	Number of retries in case of abnormal system	R/W
60321	Address Mode of COM2	Address mode: 0-standard mode; 1-extended mode	R
60322	Time of COM2 timeout	Set communication timeout; unit: ms	R/W
60323	Current waiting time of COM2 communication	Current waiting time of communication; unit : ms	R/W
60324	HMI Site of COM2	0-Machine; 1-Far-end	R/W
60327	Number of historical alarm	Sum of recorded historical alarm	R/ Double word
60329	Number of historical data	Sum of recorded historical data	R/ Double word
60333	Keyboard Language Switch	1-Chinese keyboard, 0-English keyboard; aim at standard keyboard	R/W
60334	CPU usage	Current CPU usage rate during the running of system	R
60335	Backlight luminance value	Backlight luminance value	R/W
60337	Creation time of the current project: year	Creation time of the current project: year	R
60338	Creation time of the current project: month	Creation time of the current project: month	R
60339	Creation time of the current project: day	Creation time of the current project: day	R
60340	Current language	Set current language	W
60359	Copy historical alarm between specific serial number range to U disk when the value is 1		R/W
60360	Copy historical alarm between specific serial number range to SD card when the value is 1		R/W
60361	Copy historical alarm between specific time range to U disk when the value is 1		R/W
60362	Copy historical alarm between specific time range to SD card when the value is 1		R/W
60363	Export the start serial number of historical alarm		R/W
60365	Export the end serial number of historical alarm		R/W