TouchWin

C language function

User's manual

Xinje Electronic Co., Ltd.

Catalog

1. Introduction

C language function is added in the Touchwin software v2.C.6 and higher version. With this new function; TH and TP series HMI can realize more complicated operations.
We will explain the C function programming rules with simple examples.

2. C function

2.1 Functions

The C function writing mode is the same as C language. C functions include public functions and performance functions.

2.1.1 Public functions

Public functions: support basic call for performance functions. It needs to write function prototype, parameters and return value are allowed.

Example:

```
DWORD Crc(BYTE* pBytes, int Length)
{
   DWORD dwCrc = 0;
            …
   return dwCrc;
   }
```

Call performance functions in public functions:

```
void CallFunction()
  {
        Func1();
    }
```

2.1.2 Performance functions

Performance functions: no return value, no parameters. It no needs to write function prototype, only has to specify the function name, direct write the function body.
Performance functions can be called through the function button and function field in Touchwin software.

Example:

```
BYTE byArray[10];
DWORD dwCrc = 0;
dwCrc = Crc(byArray, 10);      // call public function
```

2.2 Predefined value type

```
UINT == unsigned int   // 32bits, 4 bytes, same as DWORD
DWORD == unsigned long
WORD == unsigned short
BYTE == unsigned char
BOOL == unsigned char
```

2.3 Predefined value table

2.3.1 Constant

```
TRUE == 1     // used to BOOL
FALSE == 0
NULL   == 0    // used to pointer initialization
```

2.3.2 Serial port

```
HMI_LOCAL_MCH = -1
DOWNLOAD = 0
PLC = 1
```

2.3.3 Internal registers

```
TYPE_PSB    = 0
TYPE_PSW    = 1
TYPE_PFW    = 2
TYPE_PRW    = 3
TYPE_PHW    = 4
TYPE_PUW    = 5
TYPE_PCW    = 6
```

2.3.4 Value length

```
TYPE_NONE     = 0     //
TYPE_BIT      = 1     // bit
TYPE_BYTE     = 2     // byte
TYPE_WORD     = 3     // word
TYPE_DWORD    = 4     // double words
TYPE_REGS     = 5     // register group
TYPE_BYTE_3   = 6     //
```

2.4 Macro

1. Max(a, b)
Example: Max(3, 4) == 4

2. Min(a, b)
Example: Min(3, 4) == 3

3. Combine two bytes in one word
MAKEWORD(lb, hb)
Example: MAKEWORD(0x01, 0x02)== 0x0201

4. combine two words in one double words
MAKELONG(lw, hw)
Example: MAKELONG(0x01, 0x02)== 0x00020001

5. Obtain the low byte of one word
LOBYTE(w)
Example: LOBYTE(0x0201) == 0x01

6. Obtain the high byte of one word
HIBYTE(w)
Example: HIBYTE(0x0201)== 0x02

7. Obtain the low word of one double words
LOWORD(l)
Example: LOWORD(0x00020001) == 0x0001

8. Obtain the high word of one double words
HIWORD(l)
Example: HIWORD(0x00020001) == 0x0002

2.5 Direct operation for HMI internal registers

2.5.1 PSW registers

PSW registers can be operated directly, the type is unsigned short (WORD).
Example:
PSW[300]++； // PSW[300]++ as the word

DWORD dwValue = *(DWORD*)(PSW + 300);
/ / assign the value of PSW[300] and PSW[301] to one double words

float fValue = *(float*)(PSW + 300);
// read the value of PSW[300] and PSW[301] as the float format

(DWORD)(PSW + 300) = dwValue;
// assign one double words to PSW[300] and PSW[301]

2.5.2 PSB operation functions

GetPSBStatus(PSB_No) // obtain the value of PSB
SetPSB(PSB_No) // PSB=1
ResetPSB(PSB_No) // PSB=0

Example:
// PSB(301) = PSB(300)
if(GetPSBStatus(300))
SetPSB(301);
else

ResetPSB(301);

2.6 Notice

1. When input API function, make sure the function name is together with "(" and no space between them. Dialog and tip box will pop up by doing this.
2. Capital and small letter is distinguished for the code.
3. It cannot assign initial value to the global variables defined in the public function. The default value of global variables is 0.
4. When define the variables, the type must be the same to the data source.
5. The performance function name must be English and cannot be the same with others.
6. Press F7 to compile the program.
7. When declare the variables (global or local variables), don't declare the array whose space larger than 128 bytes. It can use special space allocation function.
8. Cannot call malloc/ free directly, but please use Malloc / Free (first letter is capital).
9. The execution environment of performance function is multi-tasking.
The execution mode of performance function: parallel execution, sequential execution.
Sequential execution: The task which calls the performance function enables to do the next operation after the performance function execution is finished. So performance function must have suitable exit condition.
Parallel execution: The task which calls the performance function will build new task to execute the function. The task will do the next operation.
10. Use carefully as the multi-task system has task lock.

3. Application

3.1 Purpose

Get 3 integral from the PLC, show the min and max ones on the screen.

3.2 Device

This project needs the following devices:

1. TH series HMI: TH465-MT 1pcs
2. XC series PLC: XC3-24R-E 1pcs
3. Software: Touchwin version 2.c.6 and higher
4. Cables: USB download cable 1pcs, PLC cable 1pcs, PC

3.3 Reference manual

1. XC series PLC manual (instruction part and hardware part)
2. TH series HMI manual

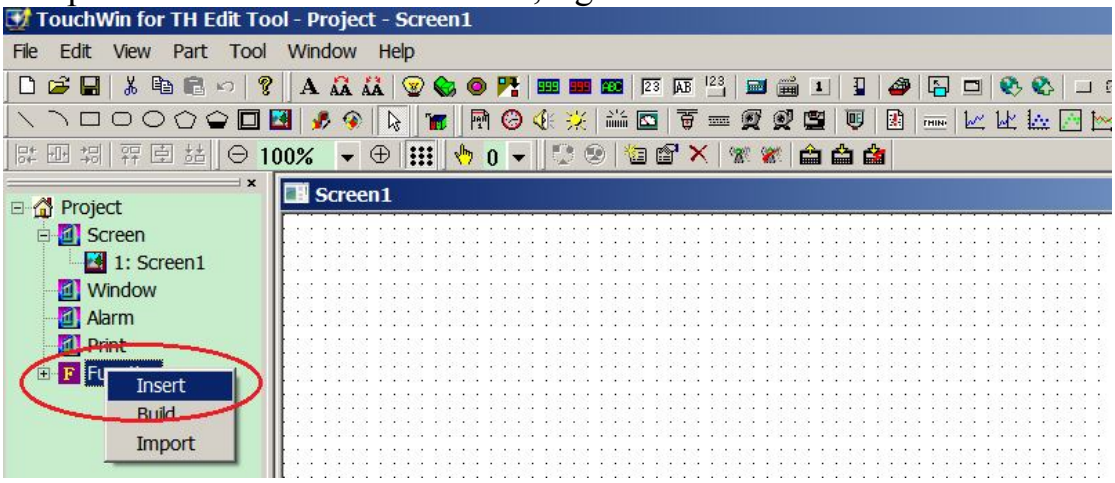
3.4 Steps

1. Set value in PLC register D0, D2, D4 via the HMI.
2. Send the value of D0, D2, D4 to the HMI.
3. Call the C function to compare the three values.
4. Show the max value in PSW300, the min value in PSW301.

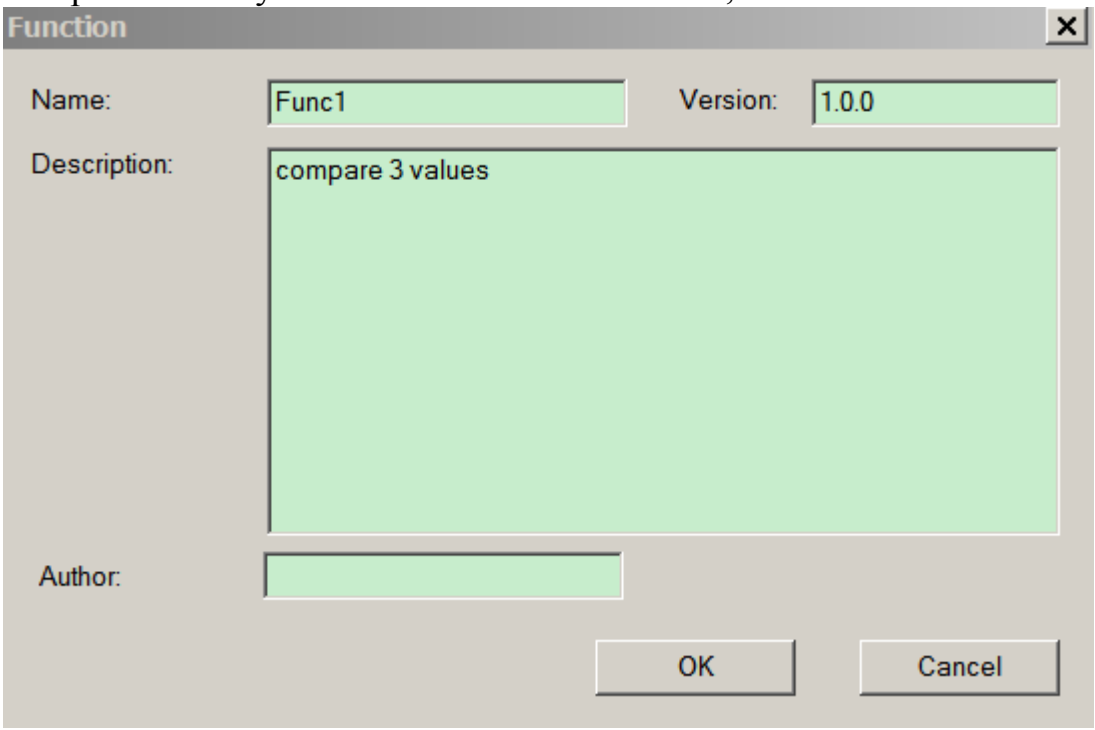
4. Make the project

4.1 Make C function

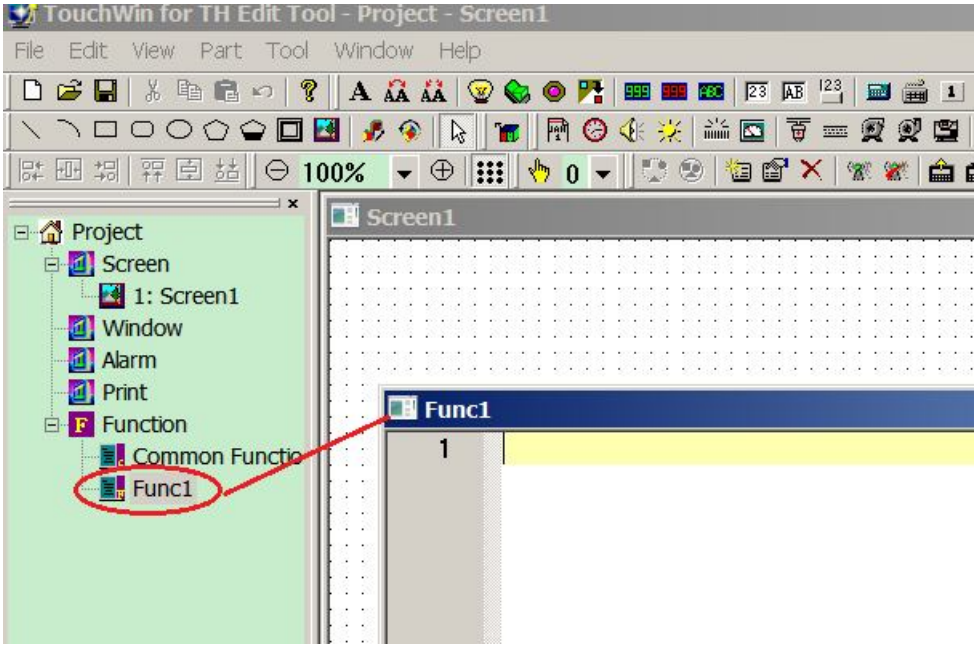
1. Open Touchwin V2.C.6 software, right click Function/Insert.



2. Input necessary information in below window, then click OK.



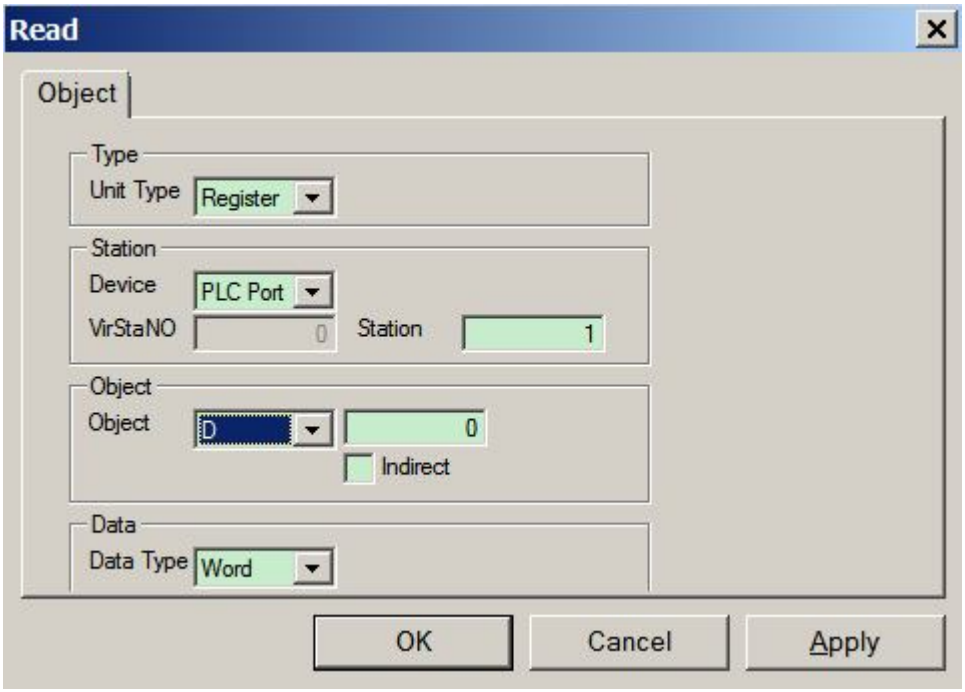
3. Click Func1 to edit the C function.



4.2 Edit the C function

1. Define the character according to the C writing rules.
WORD a,b,c,max,min;
Define the value type according to the data source type.

2. Collect the value to C function:
Read(PLC,1,XINJE_XC_REG_D,0,0,TYPE_WORD,&a);
//send the value of D0 to a
Read(PLC,1,XINJE_XC_REG_D,2,0,TYPE_WORD,&b);
//send the value of D2 to b
Read(PLC,1,XINJE_XC_REG_D,4,0,TYPE_WORD,&c);
//send the value of D4 to c
When you input “Read(”, it will pop up below window:



Unit type: the operation object type, select “register”
Station: input the PLC station No.
Object: input the object address D0
Data type: word
Notes: for details tips please refer to appendix.

3. Edit the compare program:

```
if(a>b)
    {max=a;min=b;}
 else
     {max=b;min=a;}
 if(max<c)
    max=c;
 if(min>c)
    min=c;
```

4. Send the result to the HMI:

```
Write(HMI_LOCAL_MCH,0,TYPE_PSW,300,0,TYPE_WORD,max);
//write max value to PSW300
Write(HMI_LOCAL_MCH,0,TYPE_PSW,301,0,TYPE_WORD,min);
//write min value to PSW301
```

5. The Func1 program:

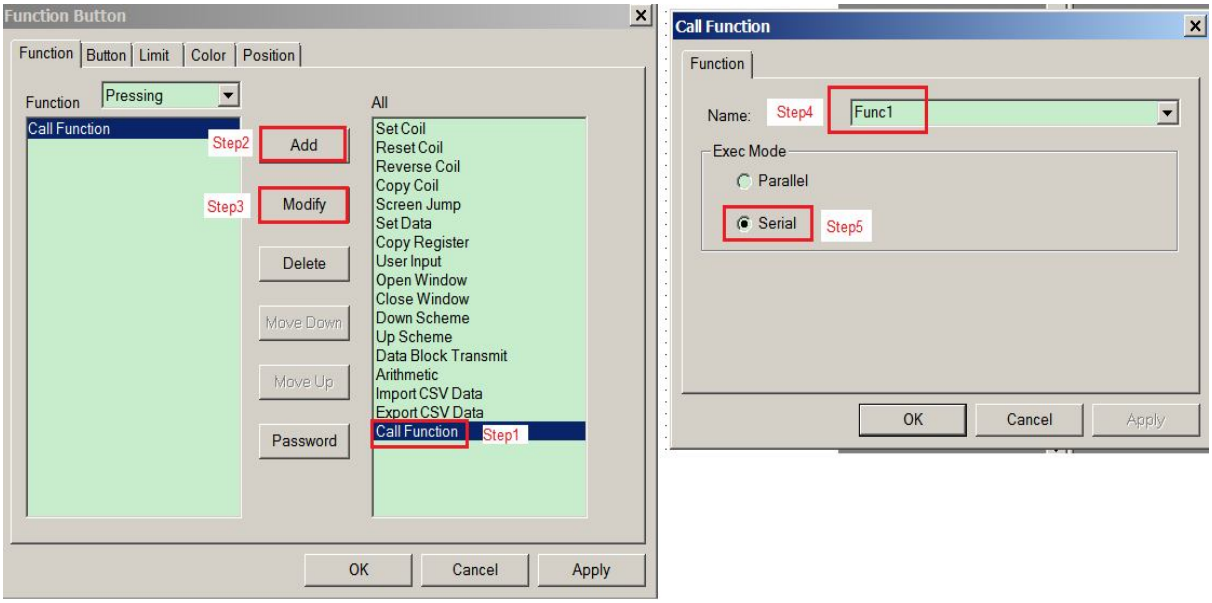
Func1

```
1   WORD  a,b,c,max,min;
2
3   Read(PLC,1,XINJE_XC_REG_D,4,0,TYPE_WORD,&c);
4   Read(PLC,1,XINJE_XC_REG_D,2,0,TYPE_WORD,&b);
5   Read(PLC,1,XINJE_XC_REG_D,0,0,TYPE_WORD,&a);
6
7   if(a>b)
8   {max=a;min=b;}
9   else
10  {max=b;min=a;}
11  if(max<c)
12  max=c;
13  if(min>c)
14  min=c;
15
16  Write(HMI_LOCAL_MCH,0,TYPE_PSW,300,0,TYPE_WORD,max);
17  Write(HMI_LOCAL_MCH,0,TYPE_PSW,301,0,TYPE_WORD,min);
18
```

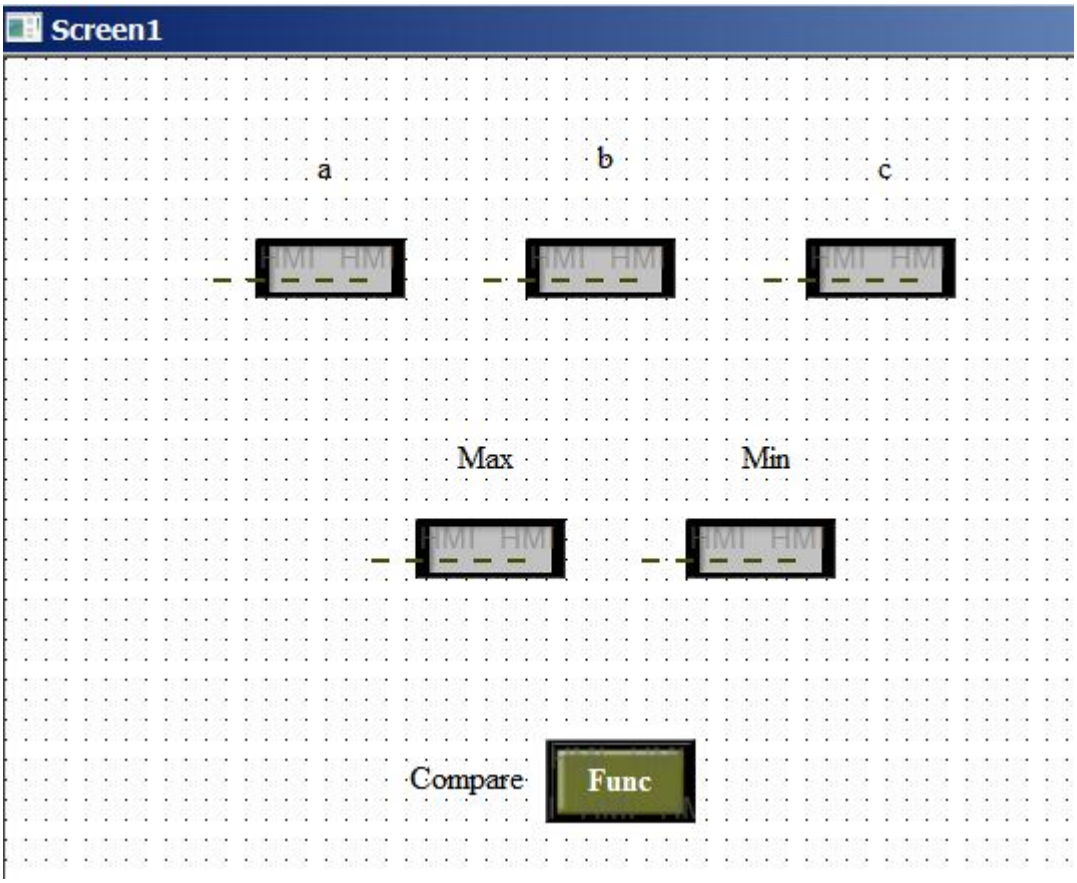
6. Press F7 to compile the C function. If the program is correct, it will show below window:

4.3 Make the HMI program

1. Put 3 digital input buttons on the screen. Set the address to D0, D2 and D4.
2. Put 2 digital display buttons on the screen. Make the address to PSW300 and PSW301.
3. Put 1 function button on the screen to call the Func1.



4. The final screen:



5. Download the HMI program to the TH465. Connect PLC with the TH465, power on. Input any value in a, b, c. Click Func button, the max and min value will show on the screen.

Appendix 1 System tips

```
WORD      a,b,c,g[4];
Read(PLC, 1, XINJE_XC_BIT_M, 0, 0, TYPE_BIT, &b);
   //Read the value of M0 to b
Read(PLC,1,XINJE_XC_REG_D,2,0,TYPE_WORD,&b);
   //Read the value of D2 to b
Reads(PLC, 1, XINJE_XC_REGS_D, 10, 4, g);
   //Read the value of D10~D13 to g
Read(HMI_LOCAL_MCH, 0, TYPE_PSB, 300, 0, TYPE_BIT, &a);
   //Read the value of PSB300 to a
Read(HMI_LOCAL_MCH, 0, TYPE_PSW, 300, 0, TYPE_WORD, &b);
   //Read the value of PSW300 to b
Reads(HMI_LOCAL_MCH, 0, TYPE_PFW, 300, 4, g);
   //Read the value of PFW300~PFW303 to array g
Write(PLC, 1, XINJE_XC_BIT_M, 0, 0, TYPE_BIT, 0);
   //Set OFF M0
Write(PLC, 1, XINJE_XC_REG_D, 50, 0, TYPE_WORD, a);
   //Write the value of a to D50
Writes(PLC, 1, XINJE_XC_REGS_D, 20, 4, g);
   //write the value of array g to D20~D23
Write(HMI_LOCAL_MCH, 0, TYPE_PSB, 300, 0, TYPE_BIT, 1);
   //Set ON PSB300
Write(HMI_LOCAL_MCH,0,TYPE_PSW,300,0,TYPE_WORD,max);
   //write the value of max to PSW300
Writes(HMI_LOCAL_MCH, 0, TYPE_PFW, 300, 4, g);
   //write the value of array g to PFW300~PFW303
```

Appendix 2 API functions

Take XC as an example:

```
/*************************************************************
************************
Function: read and write register (for bit and register)
comID: serial port (HMI_LOCAL_MCH = -1
                                        DOWNLOAD = 0,
                                        PLC = 1)
staID: station number
objType: register address type
add1,add2: register address
dataType: TYPE_BIT = 1          occupy 1 byte
          TYPE_BYTE = 2         occupy 1 byte
          TYPE_WORD = 3         occupy 2 bytes
          TYPE_DWORD = 4        occupy 4 bytes
pValue: value buffer (the length must match to dataTyp)
return value: TRUE / FALSE (successful/ failed)
*************************************************************
************************/
BOOL Read (int comID,  int staID,  int objType,  int add1,  int add2,
int dataType,  void* pValue);
BOOL Write(int comID,  int staID,  int objType,  int add1,  int add2,
int dataType,  DWORD dwValue);
Example:
BOOL bValue = FALSE;
WORD wValue = 0;
Read(PLC, 1, XINJE_XC_BIT_M, 0, 0, TYPE_BIT, &bValue);
      // read the bit
Read(PLC, 1, XINJE_XC_REG_D, 0, 0, TYPE_WORD, &wValue);
      //read D[0]
Read(HMI_LOCAL_MCH,  0,  TYPE_PFW,  300,  0,  TYPE_WORD,
&wValue); //read PFW[300]
Write(HMI_LOCAL_MCH,  0,  TYPE_PFW,  300,  0,  TYPE_WORD,
wValue);  //write PFW[300]
```

/**
****
Function: read and write register array
comID: serial port (HMI_LOCAL_MCH = -1
DOWNLOAD = 0,
PLC = 1)
staID: station number
objType: register address type
add1: register address
regs: register quantity
pRegs: value buffer (the length must be match to the read&write register array)
return value: TRUE / FALSE (successful/failed)
****
**********/
BOOL **Reads**(int comID, int staID, int objType, int add1, int regs, void* pRegs);
BOOL **Writes**(int comID, int staID, int objType, int add1, int regs, void* pRegs);
Example:

```
WORD wArray[10];
Reads(PLC, 1, XINJE_XC_REGS_D, 0, 10, wArray);
Writes(PLC, 1, XINJE_XC_REGS_D, 0, 10, wArray);
```

/**
****
Function: Enter, Leave: signal control, ensure the communication is synchronization mode. Use together with send and receive.
****
**********/
void **Enter**(BYTE ComID);
void **Leave**(BYTE ComID);
/**
****
Function: send data of serial port
comID: serial port (DOWNLOAD = 0, PLC = 1)
SndBuf: sending buffer, the type is byte
Len: sending data length, count as byte
Return value: TRUE / FALSE (successful/failed)
****
**********/
BOOL **Send**(BYTE ComID, BYTE *SndBuf, WORD Len);
/**

**********/
Function: receive the data of serial port
comID: serial port (DOWNLOAD = 0, PLC = 1)
RcvBuf: receiving buffer, the type is byte
Len: receiving data length, count as byte
Timeout: receiving timeout time (0:always waiting). Unit: ms
TimeOutByte: receiving timeout time of bytes (set to 6)
Return value: received data length, count as byte
/**********
**********/
WORD Receive(BYTE ComID, BYTE *RcvBuf, WORD Len, WORD TimeOut, BYTE TimeOutBytes);
Exp:

```
BYTE byArray[10] = {0x00, ….};

Enter(PLC);                    // apply the serial port
Send(PLC, byArray, 10);
Receive(PLC, byArray, 10, 0, 6);
Leave(PLC);                    // release the serial port
```

/**********
****
Function: Malloc, Free: instead of malloc, free.
Note: please release the applied space in time
****
**********/
/**********
****
Function: apply heap space
Size: apply the space size(bytes)
Return: the applied space address, NULL means the application is failed
****
**********/
void *Malloc(UINT size)
/**********
****
Function: release heap space
pBuffer: the space ready to release
****
**********/
void Free(void *pBuffer)
Example: BYTE* pBuffer = Malloc(10);

```
Free(pBuffer)
```

```
/*************************************************************
************************
Function: Lock, UnLock: task lock, use them in pairs
Note: the locked area: access manage for global variables, make the
locked area as small as possible
Suitable case: the task lock is needed when there are many performance
functions need access for one global variable
*************************************************************
************************/
void Unlock (void);
void Lock (void);

/*************************************************************
************************
Function:  Delay
ms:  delay time (unit:ms), the max delay time = 0xFFFF * delay precision
delay precision: TP series(except TPA61-T) and OP series are 10ms;
TPA61-T and TH series are 5ms
*************************************************************
************************/
void Delay( UINT ms);
Example:
        Delay(1000);                    // delay 1s
```

Appendix 3 The calling limit for C function

This chapter will introduce the restricted library functions.
Most C functions can be used normally (except heap functions). However, the following functions are limited when using.

1. The functions in alloca.h cannot be called, they are related to heap.
2. The assert functions in assert.h cannot be called.
3. The stream functions in stdio.h cannot be called, only the functions for string (sscanf, sprint) can be used normally.
4. The heap functions in stdlib.h cannot be called, but they can be instead by API functions.

TouchWin

Xinje Electronic Co., Ltd.
4th Floor Building 7,Orignality Industry park, Liyuan Development Zone, Wuxi City, Jiangsu Province 214072
Tel: (510)85134136
Fax: (510)85111290
www.xinje.com