



# **User defined protocol**

User Manual

# Catalog

<b>1</b>	<b>Introduction.....</b>	<b>3</b>
<b>2</b>	<b>Procedure.....</b>	<b>4</b>
2-1	Have a view of destination device protocol.....	4
2-2	Register and establish protocol file .....	6
2-3	Program a protocol file.....	9
2-3-1	Introduction .....	9
2-3-2	Where do program protocol code ?.....	9
2-3-3	Structure of protocol code .....	9
2-3-4	Procedure .....	10

# 1 Introduction

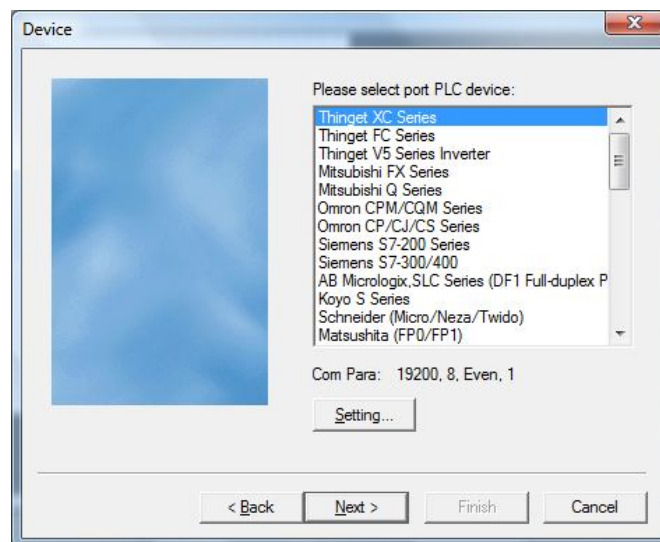
## ■ What is user-define protocol?

User-define protocol is a communication protocol programmed by users to support those devices which are not included in device list.

## ■ Why do we use user-define protocol?

As a communication rule, communication protocol is an agreement made by both transmission parties, including data structure, synchronous method, transmission speed, debug method, character define and so on. In other word, the data transmission between panels and device takes effect only when both parties comply with communication rule strictly.

As we can see from the *Touchwin* editing tool, most of the communication protocol for general PLC, inverter and other protocol in market already in the selection list of devices.



Devices list

If the destination device is not listed in the device list, please check whether the protocol of this device is same as those have existed in the list, such as 'Modbus' protocol. In this case , just select the same protocol, otherwise ,program a user-define protocol according to destination device .

## 2 Procedure

### 2-1 Have a view of destination device protocol

The following chapters take device V900(a virtual parameter) for example to describe how to program a user-define protocol.

Please find out the send and receive data information from the V900 communication protocol. In this example, the parameters including current weight, destination weight and flow, will appear on panels.

Address of assignment of V900			
Current weight	H42	Flow	H43
Destination weight	H44		

Data structure of read registers				
Request to V900	Station NO.	Function code (read registers)03	Starting Address	Checksum
Response from V900	Station NO.	High byte	Low byte	Checksum

Set communication parameters of V900 as follow:

Station NO: 1  
Baud rate:9600  
Even parity  
Data bit :8  
Stop bit:1

There is example of data structure based on V900

1. Read current weight:

Request: H01 H03 H42 SUMCHECK

Response: H01 Highbyte Lowbyte SUMCHECK

2. Read flow:

Request: H01 H03 H43 SUMCHECK

Response: H01 Highbyte Lowbyte SUMCHECK

3. Read destination weight:

Request: H01 H03 H44 sumcheck;

Response: H01 Highbyte Lowbyte SUMCHECK

## 2-2 Register and establish protocol file

### ■ Introduction

The purpose of this step is to add new device item named V900 into device list, and establish carrier file for this protocol.

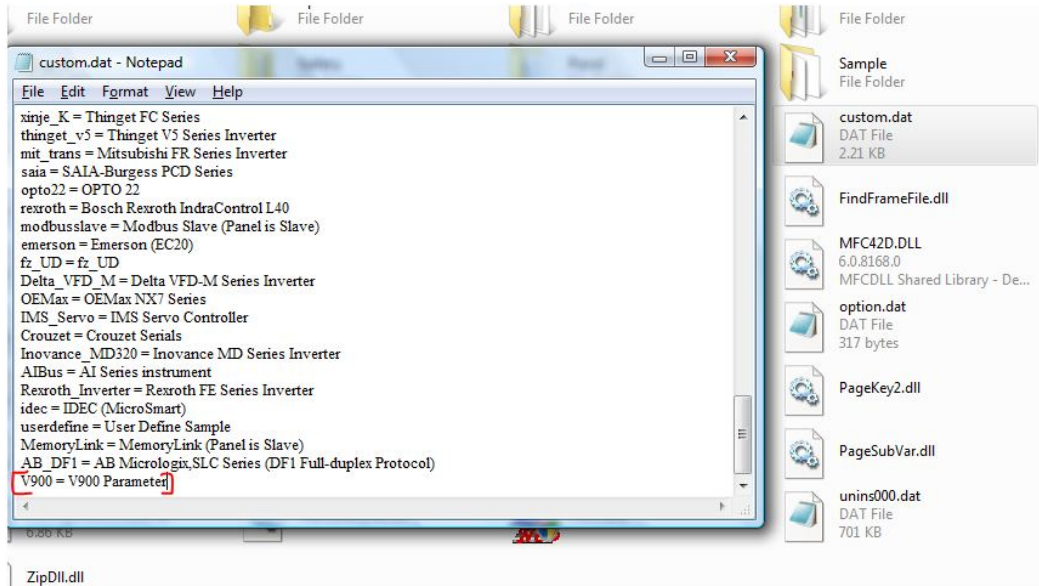
### ■ Procedure

1. Open the root directory of *Touchwin* software:



2. Find the *custom..dat* file and open in notepad format, add item 'V900=V900 parameter' in bottom part, save this and close. Please note that the left part of '=' is filename, the right part of '=' is device name existed in device list.

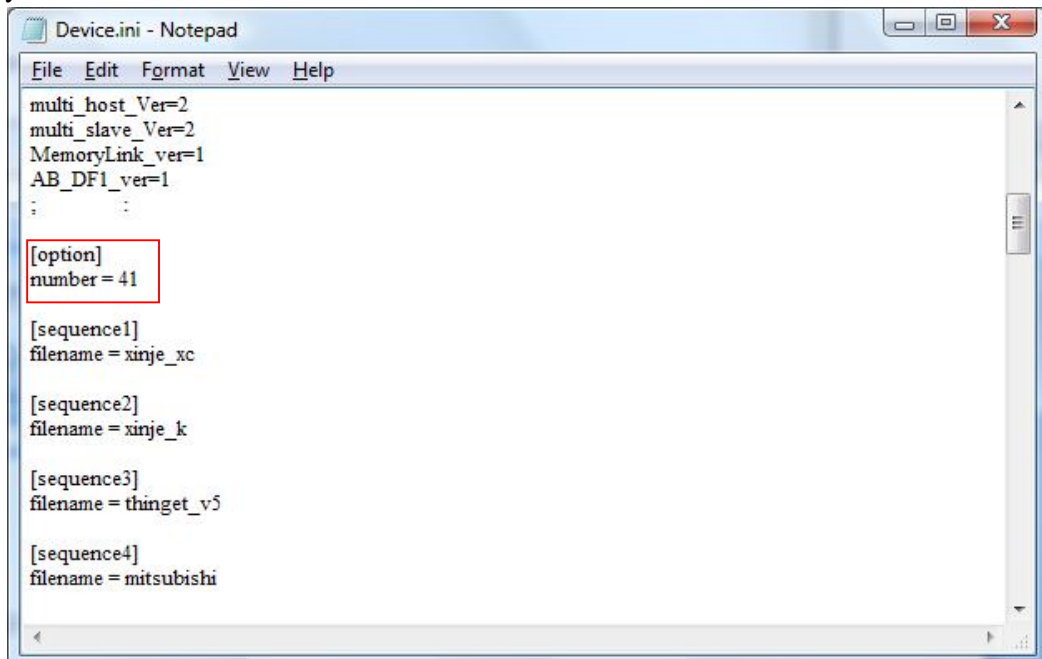
Note: please remove the attributes 'read-only' or 'hidden' if there is a problem in saving operation.



3. Open the file 'Device' in the root directory and find configuration file 'Device.ini' to check following code:



In this example, the quantity of devices is  $n=41$ , after adding a new item, now the quantity is  $n= 41+1= 42$

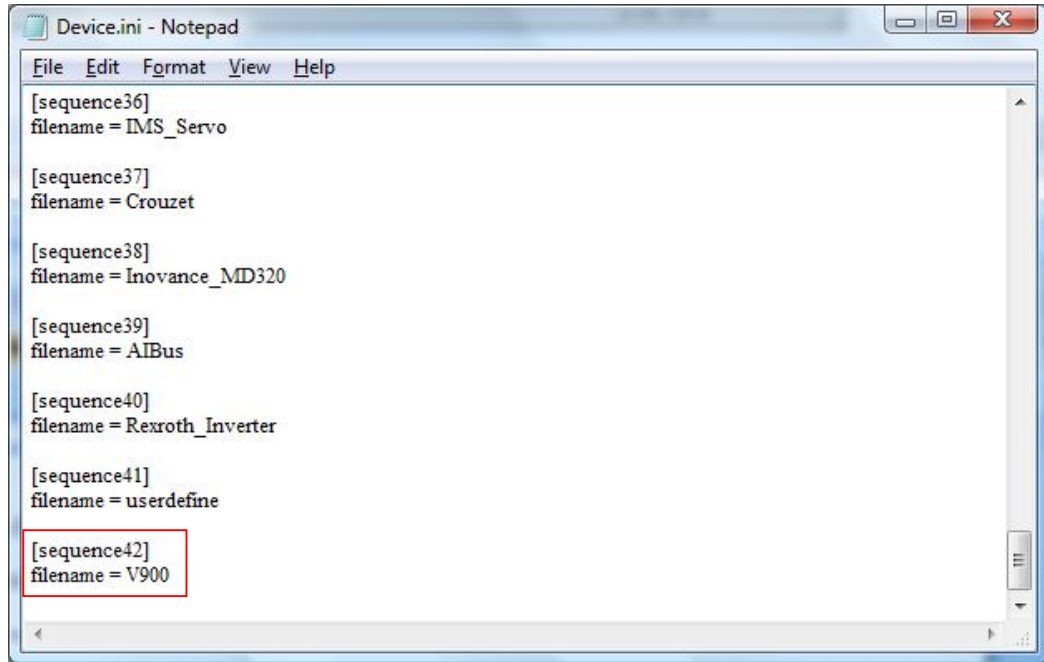


4. Please add the following content in the bottom part of 'Device.ini' file.

```
[sequence42]
```

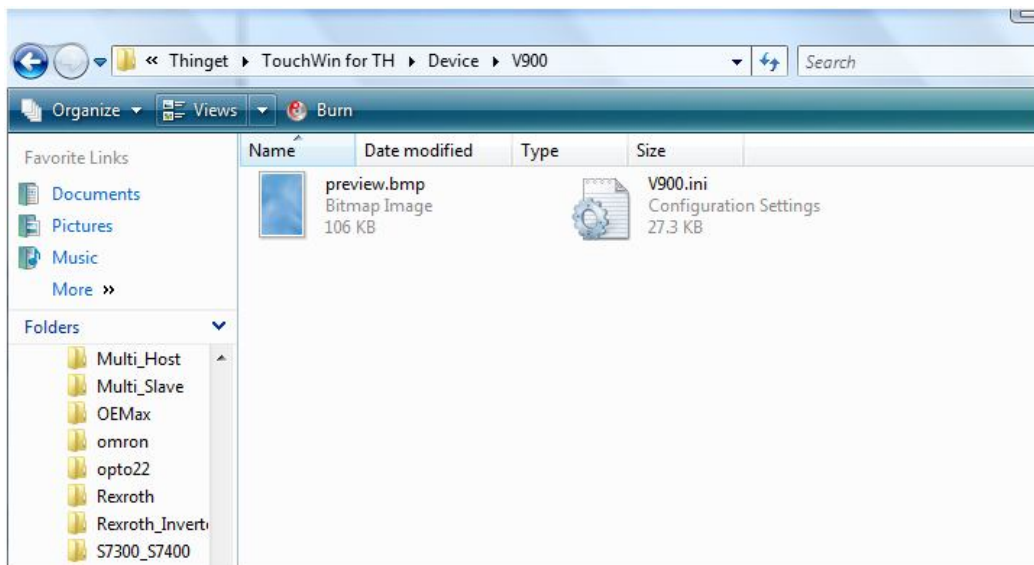
```
filename = V900
```

Save this change and exit.



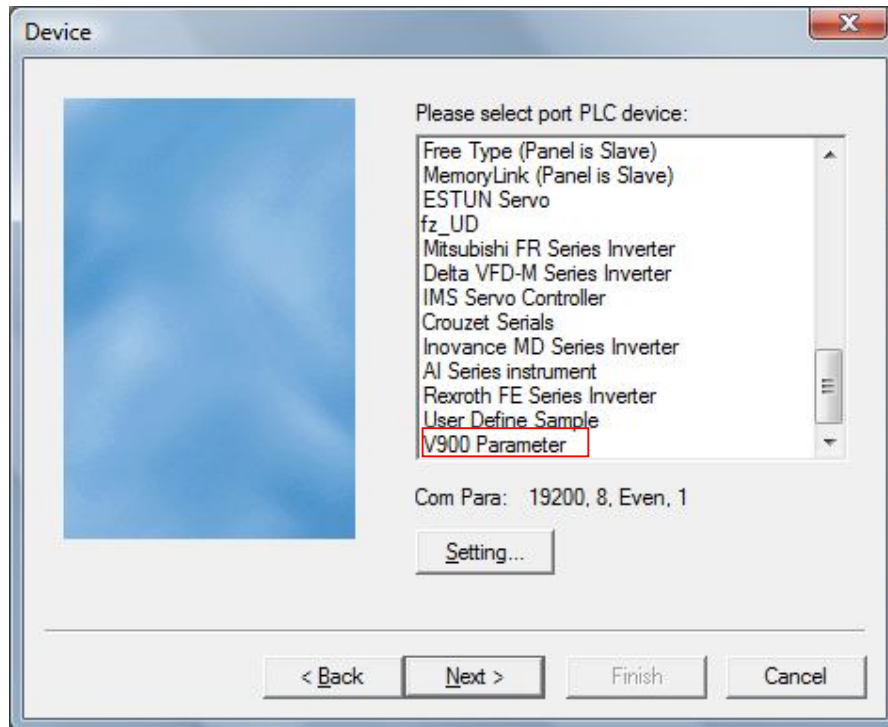
5. After this please return to 'Device' file and establish a new file named 'V900', open this new file and build the following two files.

Note: you can build these two new files by copying from other device file , and rename as 'V900'



6. Now, the step of 'register and establish protocol file' is finished and you will find the

device named 'V900 parameter is existed in the device list already.



## 2-3 Program a protocol file

### 2-3-1 Introduction

This chapter will describe how to program protocol code.

### 2-3-2 Where do program protocol code ?

Open 'V900'file in the 'Device' file from root directory to double-click file 'V900.ini' file where we program protocol code.

### 2-3-3 Structure of protocol code

AS a communication rule, protocol code consists of following parts:

- File description
- Default communication setting

- Default Station NO
- Object description
- Communication rule
- Object Type optimization

## 2-3-4 Procedure

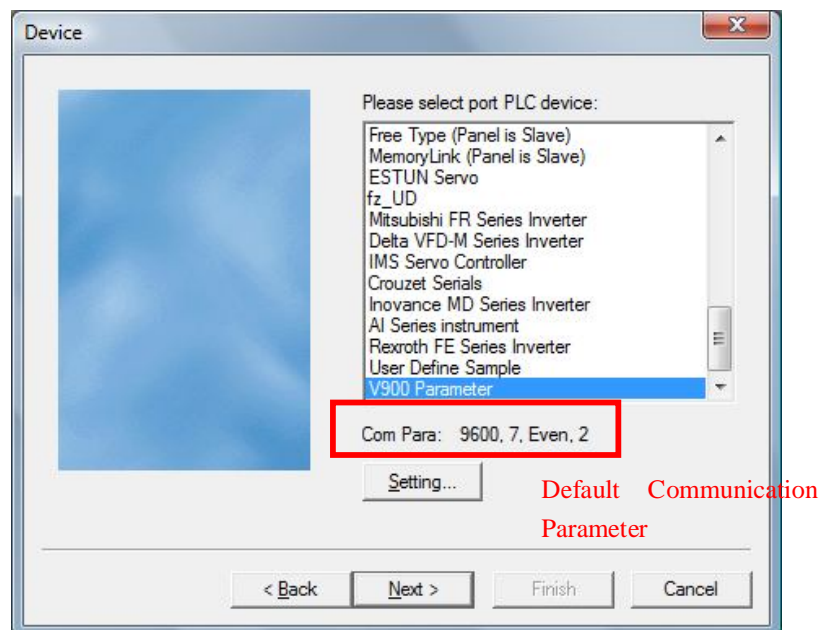
- Descript (File description)

Please write the following content in 'V900.ini' file with a standard form, as below:

```
[descript]
DeviceModelCode=22
FirmwareName=UserDefine
DownLoadDll = \Device\UserDefine\DownLoad.dll
SpecInfoDll = \Device\UserDefine\DownLoad.dll
DeviceType = 100
```

- Communication (Default communication settings)

Set the default communication parameters when you select the V900 device:



Write the following 'default communication setting' code as below:

```
[Communication]
BaudRate=9600      ; Baud rate
DataBits=8         ; Data bits
Parity=2           ; Parity 0-None, 1-Odd, 2-Even
StopBits=0         ; Stop bit 0-1 bit, 1-1.5 bits, 2-2 bits
```

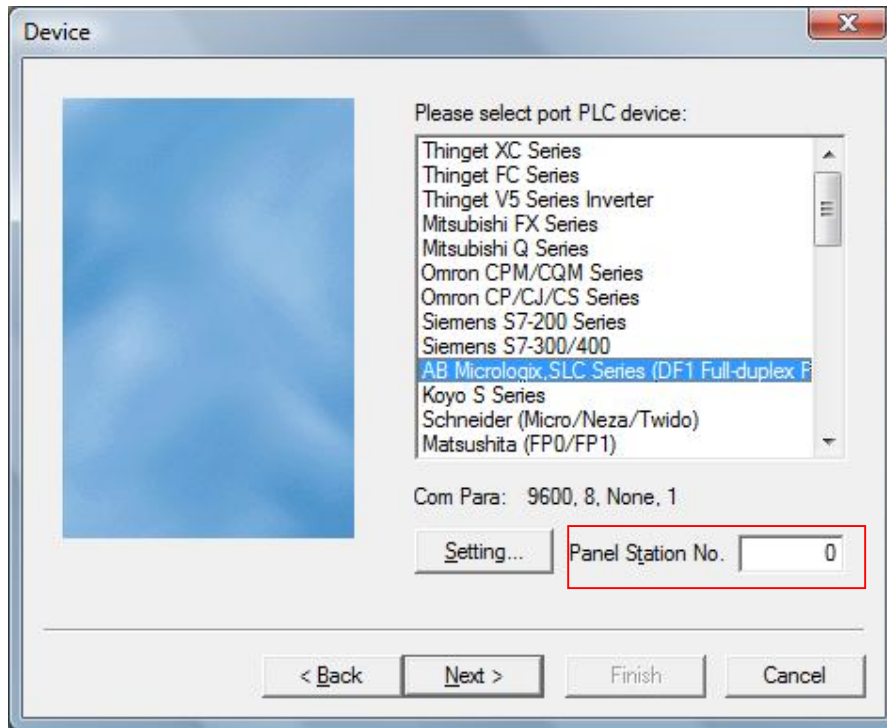
- **StationNo** (Default station NO)

This part is comprised of panel part and device part, code is showed as below:

```
[StationNo]
PanelWithStationNo=0 ; 0: station NO of panel is not permitted to set 1:settable [Note 1]
PanelDefaultStationNo= ;Default station NO of panel, 'there is no value because the station NO
                        ;is not permitted to set in this example';
PanelMinStationNo= ;the minimum Station NO of panel. 'there is no value because the
                    ;station NO is not permitted to set in this example';
PanelMaxStationNo= ;The maximum Station NO of panel . 'there is no value because the
                    ;station NO is not permitted to set in this example';
DeviceWithStationNo=1 ; 0: station NO of device is not permitted to set 1: settable [Note 1]
DeviceDefaultStationNo=1 ; Default station NO of device with 1
DeviceMinStationNo=0 ; The minimum Station NO of device
DeviceMaxStationNo=255 ; The maximum Station NO of device
```

**NOTE 1**

**Notation:** As a slave in communication system, the station NO of panel is settable with setting *PanelWithStationNo=1* as showed in the device list; when *PanelWithStationNo=0*, the settable information is not eyeable.



- Object (Object description)

This chapter takes example to describe how to write code for *object description*. If we need to read the current weight of V900 device, we should define the occupied spaces and communication rule.

[Object]

ObjectNum=13 ; Numbers of objects

[Object1] ; The first object

CanAct=1 ; Space unit occupied by objects, 0-bit; 1-register; 2-register group [Note 1]

IDSymbol= Current Weight ; Item name lied in software [Note 2]

TypeNo=0 ; Corresponding 'Object Type optimization' NO.

bitlength = 16 ; 'current weight' occupies 16 bits

DescripNum=1 ; Section No. of *object description* [Note 3]

CanSelectType=4 ; Data type selection: (0000 0100) ;5-3 bytes,4-n Regs,3-dword, 2-word,1-byte,0-bit 【Note 4】

ReadWord = 1 ; Called communication rule No. of 'read register'

[Object1Descrip1] ; Object description Section 1

Caption= ; standard form

Type=0 ; Type of digital input in software

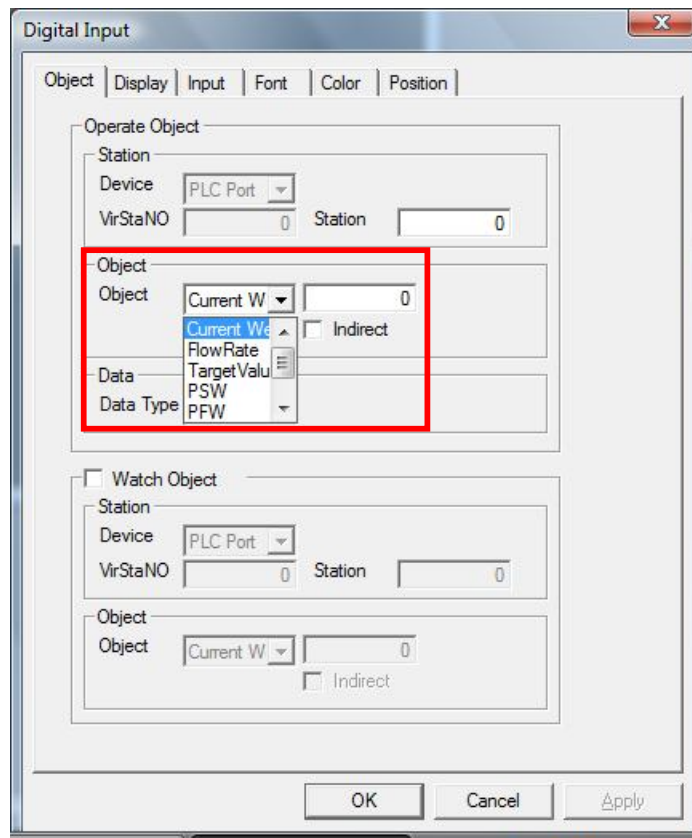
0-Number,1-(0-7),2-(00-07),3-(0-15),4-(00-15),  
5-(0-F),6-(00-0F),7-(0-31) 【Note 5】

DefaultNumber=0 ; Default value of digital input in software  
MinNumber=0 ; Min number of digital input in software  
MaxNumber=0 ; Max number of digital input in software  
format=10 ; Data format of digital input in software  
NumberStep=1 ; Standard format

Notation :

**【Note 1】:** This item is used to define the *object type* that this item belongs to: bit, register or registers. When we select with Bit, this means the object name will appear on *bit* component like ‘*lamp button*’; when we select with register, this means the object name will appear on *word* component like ‘*digital input, digital display*’; when we select registers, that means object name will appear on *registers* components like ‘*character input*’.

**【Note 2】:** Component will contains items--‘*current weight*’, as below:



**【Note 3】 :** *Section No.* of object description, we set with ‘1’ in this example which means there is only one description value with this object type, as above.

But it is noted that there are two description values for some object type in some protocol. For example, we can see that the object type D in *Omron PLC* protocol consists of two sections, like *D100B* is comprised of 100 which is in Decimalist format and B which is in range from 0 to F, thus we can program code as follows (note : A part is code for 100, B part is code for B):

```
[Object9]
CanAct=0
IDSymbol=D
TypeNo=0
bitlength = 16
```

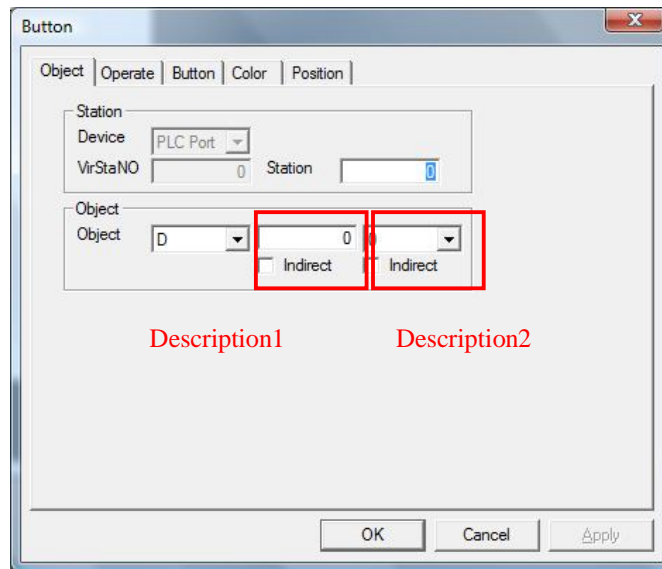
DescripNum=2  
 CanSelectType=1  
 ReadBit=16  
 SetBitOn=23  
 SetBitOff=23

[Object9Descrip1] **A part** ;

Caption=  
 Type=0  
 DefaultNumber=0  
 MinNumber=0  
 MaxNumber=99999  
 format=10  
 NumberStep=1

[Object9Descrip2] **B part**

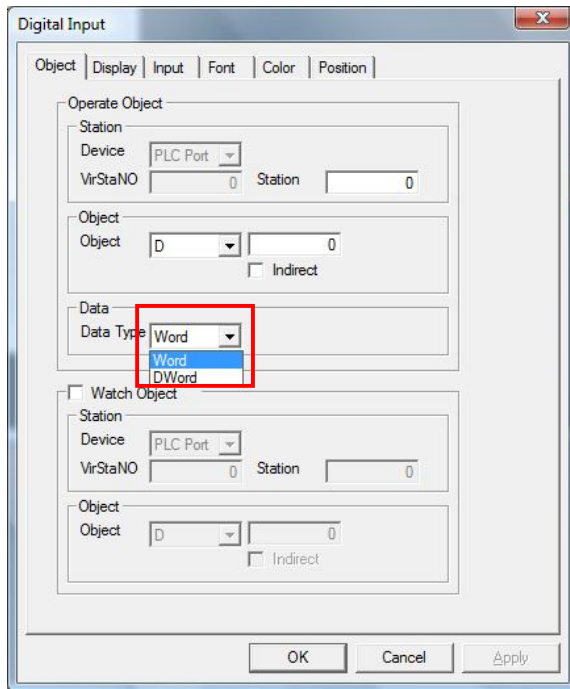
Caption=  
 Type=3  
 DefaultNumber=0



**【Note 4】 : Define data type which is comprised of 8 bits in binary system:**

Bit	7	6	5	4	3	2	1	0
1	-	-	3 bytes	n Regs	dword	word	byte	bit
0	-	-	-	-	-	-	-	-

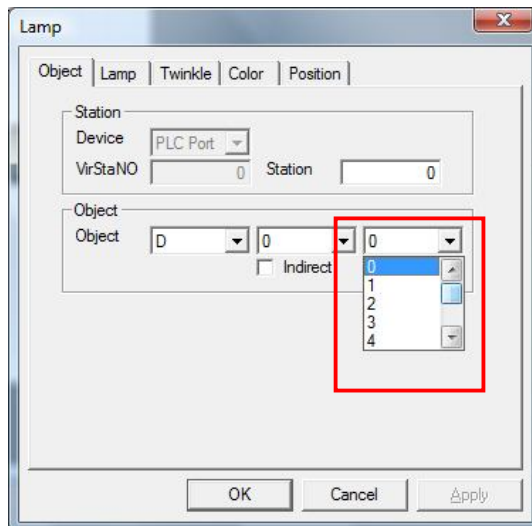
If we need option both word and Dword, we can have value 00001100 , in decimalist form is 12, thus we can write code CanSelectType=12, as follow:



**【Note 5】: Type of digital input in software**

Standard Value.	Type of digital input
0	0~ the maximum value
1	(0-7)
2	(00-07)
3	(0-15)
4	(00-15)
5	(0-F)
6	(00-0F)
7	(0-31)

When we select standard value with 5 , the display in software is showed as below:



- CommuRule (Communication rule)

This chapter describes the data structure of request as a hardcore of a protocol.

[CommuRule]

CommuRuleNum = 19 ; Numbers of communication rule  
 IsUseASC = 0 ; Use ASC format or not 0:No use ASC, 1: use ASC as Transmit Data  
 LowBitToASC = 0 ; Ignore high bit or not ? 0:NO, 1:YES 【Note 1】  
 IsHighBitNext = 0 ; Based on ASC convert, is high bit or low bit in the first 0:NO, 1:YES 【Note2】  
 IsHighByteNext = 0 ; Take word as basic unit, is high byte in the first position? 0:No, 1:YES  
 IsHighWordNext = 0 ; Take Dword as basic unit, is high word in the first position? 0:NO, 1:YES

[CommuRule1]

IsInherit = 0 ; This communication rule is inherit or base? base=0 inherit=1 【Note 3】  
 GroupNum = 2 ; Group Number of data transmission, set this value with 2 because the data transmission consists of two steps: request and response.  
 Group1Type= 0 ; Group 1 is defined as data request  
 Group1Blocks = 4 ; Block quantity of group 1 is 4  
 Group2Type = 1 ; Group 2 is defined as data response  
 Group2Blocks = 3 ; Block quantity of Group 2 is 3  
 Group1Block1 = 4 ; The 1<sup>st</sup> block of group1: Station No. of device 【Note 4】  
 Group1Block2 = 5 ; The 2<sup>nd</sup> block of group1: Function code. 【Note 4】  
 Group1Block2\_CmdNo = 3  
 Group1Block3 = 1 ; the 3<sup>rd</sup> block of group1: Static data 【Note 4】  
 Group1Block3\_StaticValue = 0x42 ;  
 Group1Block4 = 11 ; The 4<sup>th</sup> block of group1: check 【Note 4】  
 Group1Block4\_StartBlock = 1  
 Group1Block4\_EndBlock = 3  
 Group1Block4\_SelectMethod = 1  
 Group2Block1 = 4 ; The 1<sup>st</sup> block of group2: 【Note 4】  
 Group2Block2 = 10 ; The 2<sup>nd</sup> block of group2: data 【Note 4】  
 Group2Block2\_DataAreaType = 1  
 Group2Block3 = 11 ; The 3<sup>rd</sup> block of group2: check 【Note 4】  
 Group2Block3\_StartBlock = 1  
 Group2Block3\_EndBlock = 2  
 Group2Block3\_SelectMethod = 1

[CommuRule2]

IsInherit = 1 ; This communication rule is inherit or base? base=0 inherit=1  
 InheritRule = 1 ; Where does this communication rule inherit from?

Group1Block3 = 1  
 Group1Block3\_StaticValue = 0x44 } Enumerate the block need to redefine

**Notation**

**【Note1】 :**

There is an example for how to interpret this code, for example, after converting to ASC format ,0x56 becomes 0x35,0x36 , if we set this with *LowBitToASC = 0*, so the code 0x56 becomes 0x36 .

**【Note2】 :**

There is an example for how to interpret this code, for example, after converting to ASC format,0x56 becomes 0x35,0x36, if we set with *IsHightBitNext = 1*, so the code 0x56 becomes 0x36, 0x35.

**【Note3】**

**Base:** define each block of communication rule.

**Inherit:** only to enumerate the block need to redefine when the communication rule is similar to any base communication rule.

**【Note4】 :** Nearly all transmission data structure consists of these parts: station No, function code, data content, data quantity, check sum. And how about the description in Twin user-define protocol, there is an example:

**Group1Block4 = 11**  
**Group1Block4\_StartBlock = 1**  
**Group1Block4\_EndBlock = 3**  
**Group1Block4\_SelectMethod = 1**

As above, we can see that *Group1Block4 = 11* defines the function of *Group1Block4*, '11' is the code name which means check sum. And *Group1Block4\_StartBlock* , *Group1Block4\_EndBlock* , *Group1Block4\_SelectMetho* are the extended definition of *Group1Block4*, Thus, each block is comprised of function definition and extended definition, for some simple function ,there only need to define function, such as *Group1Block1 = 4*

The following chapter describes the details of code name of function definition(X: group No, Y: block No.):


Code name of function definition	1	
Meaning	Static data: use in the case that block value is defined	
Extended definition	Group X Block Y _StaticValue	

Code name of function definition	2	
Meanings	Block length: Bytes length from starting block to end block.	
Extended definition	Group X Block Y _LowBitToASC	Only remain low bit after ASC conversion. For example, 0x56 ASC Convert to 0x35, 0x36, after pass through LowBitToASC , left 0x36.

	Group X Block Y _StartBlock	Start Block number, this block is included in count.
	Group X Block Y _EndBlock	End Block number, this block is included in count.

Code name of function definition	4	
Meanings	Station No.: define the device station No.	
Extended definition	Group X Block Y _LowBitToASC	Only remain the low bit after ASC conversion.

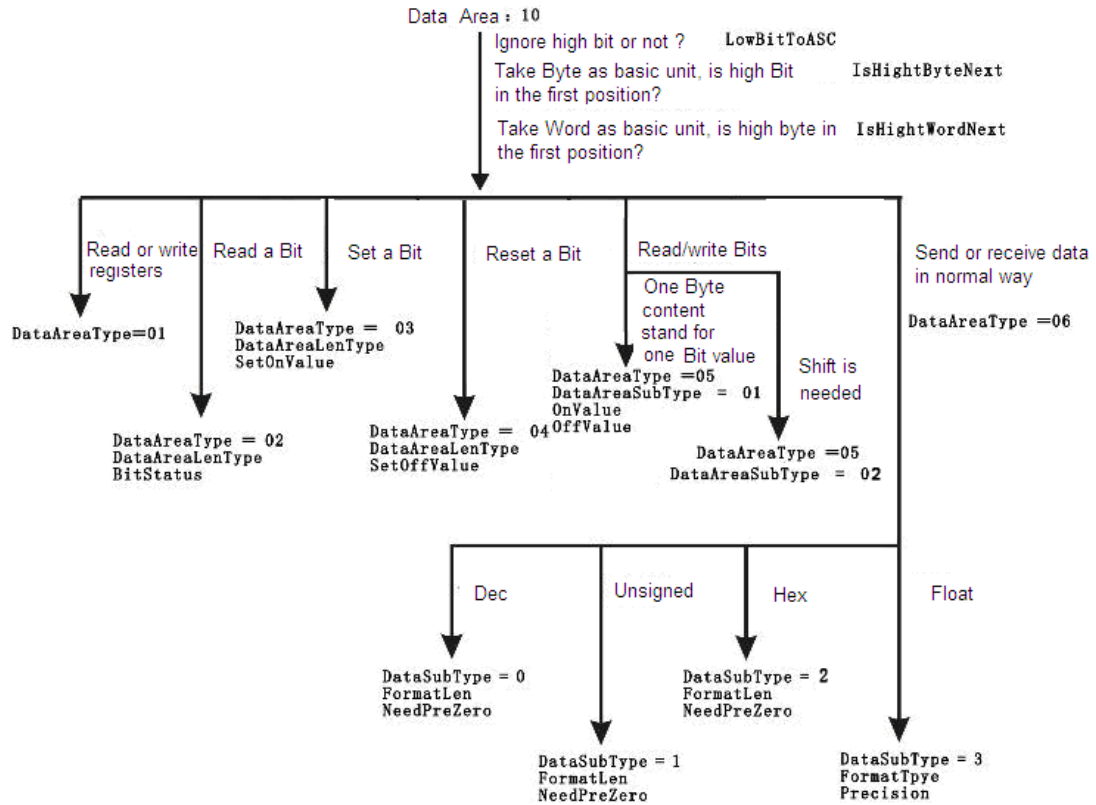
Code name of function definition	5	
Meanings	Command code: define code for 'write' or 'read' operation.	
Extended definition	Group X Block Y _LowBitToASC	Only remain the low bit after ASC conversion.
	Group X Block Y _CmdNo	Take Modbus for example, we use function code 3 for 'read holding registers', so it is expressed as <i>Group X Block Y _CmdNo=3</i>

Code name of function definition	7		
Meanings	<p>Parameter address: example, we need to set object with D1000 if we want to read the value of D1000,thus ,1000 is taken as parameter address, as below:</p>  <p>When parameter in software is set as 1000, after converting to <i>hexadecimal</i> format, it becomes 03 E8.</p> <p>When parameter in software is set as 1000, after converting to ASC ,it becomes 31 30 30 30.</p>		
Extended definition	Parameter address convert to <i>hexadecimal</i> format	Group X Block Y _ DispMode=0	Mode selection(DispMode=0, convert address to <i>hexadecimal</i> format)
		Group X Block Y _LowBitToASC	Only remain low bit

			after <i>ASC</i> conversion
		Group X Block Y_IsHightByteNext	For word, high byte in the first position with low byte followed
		Group X BlockY_IsHightWordNext	For Dword, high word in the first position with low word followed.
		Group X Block Y _ ParaAddress	Expression of address conversion
		<p>The reason why we need expression of address conversion is that the parameter address face to users is different with the ones in data transmission, such as Xinje PLC: the address face to user is TD0, but in transmission process it is H3000, so we need a expression to describe the relation between them. As we can see that ,the address in transmission data in TD N is <math>N+0x3000</math>, thus the expression is <math>Block Y\_ParaAddress = N+0x3000</math>.</p>	
		Group X Block Y _HoldSpaceSize	(1: Byte; 2: Word; 3: DWord) which means space size, for example , Group X BlockY_HoldSpaceSize =2 defines a word size.
Parameter address convert to <i>ASC</i> format		Group X Block Y _ DispMode = 1	Mode selection(DispMode=1 convert address to <i>ASC</i> format)
		Group X Block Y _LowBitToASC	Only remain low bit after <i>ASC</i> conversion
		Group X Block Y_IsHightByteNext	For word, high byte in the first position with low byte followed
		Group X BlockY_IsHightWordNext	For Dword, high word in the first position with low word followed.
		Group X Block Y _FormatLen	space size with unit Byte.
		Group X Block Y _NeedPreZero	(0: No need; 1: Need) Need 0 lead or not
		Group X Block Y _ ParaAddress	Expression of address conversion

<b>Code name of function definition</b>	<b>8</b>	
<b>Meanings</b>	Numbers of parameters: the numbers of parameters to operate. Sometime it needs to operate several continuous parameters in one transmission data to improve efficiency.	
<b>Extended definition</b>	<b>Group X Block Y _LowBitToASC</b>	Only remain low bit after ASC conversion
	<b>Group X Block Y _IsHightByteNext</b>	For word, high byte in the first position with low byte followed
	<b>Group X BlockY _IsHightWordNext</b>	For Dword, high word in the first position with low word followed.
	<b>Group X Block Y _HoldSpaceSize</b>	which means space size, for example , <i>Group X BlockY_HoldSpaceSize=2</i> defines a word size.
	<b>Group X Block Y _CountMethod</b>	(1: BITS; 2: BYTES; 3: WORDS; 4: DWORDS)

**Code name of function definition: 10, which describes the data area, the structure is showed as below:**



<b>Code name of function definition</b>	<b>10</b>	
<b>Meanings</b>	Data area: save space for transmit data. For example, if we want write 100 into register D100, thus value 100 is the destination to be saved, meanwhile , this area also save the read data.	
<b>Extended definition</b>	Group X Block Y _LowBitToASC	Only remain low bit after ASC conversion
	Group X Block Y _IsHighByteNext	For word, high byte in the first position with low byte followed.
	Group X BlockY _IsHighWordNext	For Dword, high word in the first position with low word followed.
	Group X BlockY _DataAreaLenType	(1: byte; 2: word; 3: dword) Group X BlockY _HoldSpaceSize= (2, 3, 4) Take effect
	Group X BlockY _BitStatus	Group X BlockY _HoldSpaceSize= 2 Take effect, the read bit value
	Group X BlockY _SetOnValue	Group X BlockY _DataAreaType= 3 take effect
	Group X BlockY _SetOffValue	Group X BlockY _DataAreaType= 4 take effect
	Group X BlockY _DataAreaSubType	Group X BlockY _DataAreaType= 5 take effect
	Group X BlockY _OnValue	Group X BlockY _DataAreaType= 5 Group X BlockY _DataAreaSubType=1

		Take effect The status seems ON when the corresponding Byte is equal to 'OnValue'	
Group X BlockY _OffValue		Group X BlockY _DataAreaType= 5 Group X BlockY _DataAreaSubType=1 take effect	
Group X BlockY _DataSubType		(0: Dec 1: Unsigned 2: Hex) Group X BlockY _DataAreaType= 6 take effect	
Group X BlockY _FormatLen		Formatlenn describes the Max.length of formatting Group X BlockY _DataAreaType= 6 Group X BlockY _DataSubType=(0, 1, 2) take effect Note: <i>Formatlen</i> describes the Max. length of formattin, but when the value is set as '0', it means the data length adjusts automatically, and <i>NeedPreZero</i> is not available.	
Group X BlockY _NeedPreZero		(0: No need; 1: Need) Group X BlockY _DataAreaType= 6 Group X BlockY _DataSubType= (0, 1, 2) take effect	
Group X BlockY _FormatType		(0: expressed in 'dddd.dddd' format ) Group X BlockY _DataAreaType= 6 Group X BlockY _DataSubType= 3 take effect	
Group X BlockY _Precision		precision of decimal Group X BlockY _DataAreaType= 6 Group X BlockY _DataSubType= 3 take effect	
Group X BlockY _ModNumber		data shift Group X BlockY _DataAreaType=5 Group X BlockY _DataAreaSubType=2 take effect	
<b>GroupXBlockY _DataAreaType: This parameter defines the data area type for saving data: Bit, Byte, Word or other type.</b>			
<b>GroupXBlo ckY_DataA</b>	<b>Value</b>	<b>Meanings</b>	<b>Description</b>
	1	Read/write register/registers	
	2	Read single Bit	Read the needful Bit status from response data word or byte. There, we take the Xinje V5series inverter for example to read the rotation direction. The address of rotation direction is H2101, the

	reaType			<p>third bit indicates the status of direction: value '0' shows forward, value '1' shows reverse , code is programmed as follow:</p> <p>[CommuRule12] ;read status of direction  IsInherit = 1  InheritRule = 1  Group1Block3 = 7  Group1Block3_HoldSpaceSize = 2  Group1Block3_ParaAddress = N+0x2101  <b>Group2Block4 = 10</b>  <b>Group2Block4_DataAreaType = 2</b>  <b>Group2Block4_DataAreaLenType = 2</b>  <b>Group2Block4_BitStatus = P &amp; (1&lt;&lt;2)</b></p> <p>Please note the red part, we already get value from <i>H2101</i>, but how to get the status of the third Bit?</p> <p style="padding-left: 40px;">Group2Block4_BitStatus = P &amp; (1&lt;&lt;2)</p> <p>We use character P stand for the value of <i>H2101</i>; and the (1&lt;&lt;2) means left shift two bits to 0000 0001, then it becomes 0000 0100, after that , do the 'and 'operation to P and 0000 0100, thus, we get the third bit value.</p> <p>If P &amp; (1&lt;&lt;2)=0, the status is OFF,  If P &amp; (1&lt;&lt;2)=not 0, the status is ON,  <b>【 More details regarding 'expression' please refer to 'expression' parts in this chapter】</b></p>
		3	Set one Bit to ON	<p>This form is always matched with inverters. On panel, we can control to start or stop the inverter with button component: ON means start inverter and OFF means control inverter to a stop.</p> <p>Command controlled to start with forward is :write value <i>H0002</i> to address <i>H2000</i>;  Command controlled to stop: wire value <i>H0007</i> to address <i>H2000</i>;</p> <p>The above means: when control button is in pressing status, the panel send command :write value <i>H0002</i> to address <i>H2000</i>;when control button is in releasing status, the panel send command : wire value <i>H0007</i> to address <i>H2000</i>;</p> <p>The following is the code to control the inverter to 'forward /stop':</p> <p>[Object12] ; forward /stop  CanAct=0 ; Bit</p>

			<p> <b>IDSymbol=</b>  <b>TypeNo=11</b>  <b>DescripNum=1</b>  <b>bitlength = 1</b>  <b>CanSelectType=1</b>  <b>ReadBit=13</b>  <b>SetBitOn=17</b>  <b>SetBitOff=18</b> </p> <p>           -----some code is omitted-----         </p> <p> <b>[CommuRule17]</b> ; set to forward         </p> <p> <b>IsInherit = 1</b>  <b>InheritRule = 15</b>  <b>Group1Block3 = 7</b>  <b>Group1Block3_HoldSpaceSize = 2</b>  <b>Group1Block3_ParaAddress = N+0x2000</b>  <b>Group1Block4 = 10</b>  <b>Group1Block4_DataAreaType = 0x03</b>  <b>Group1Block4_DataAreaLenType = 0x02</b>  <b>Group1Block4_SetOnValue = 0x0002</b> </p> <p> <b>Group2Block3 = 7</b>  <b>Group2Block3_HoldSpaceSize = 2</b>  <b>Group2Block3_ParaAddress = N+0x2000</b> </p> <p> <b>[CommuRule18]</b> ; coast to a stop/stop         </p> <p> <b>IsInherit = 1</b>  <b>InheritRule = 15</b>  <b>Group1Block3 = 7</b>  <b>Group1Block3_HoldSpaceSize = 2</b>  <b>Group1Block3_ParaAddress = N+0x2000</b>  <b>Group1Block4 = 10</b>  <b>Group1Block4_DataAreaType = 0x04</b>  <b>Group1Block4_DataAreaLenType = 0x02</b>  <b>Group1Block4_SetOffValue = 0x0007</b> </p> <p> <b>Group2Block3 = 7</b>  <b>Group2Block3_HoldSpaceSize = 2</b>  <b>Group2Block3_ParaAddress = N+0x2000</b> </p>
		4	<p> <b>Set single Bit to ON</b> </p> <p> <b>Same as above</b> </p>
		5	<p> <b>Read/write Bits</b> </p> <p> <b>1. DataAreaSubType = 0x01</b>  <b>Byte value read from data area means status</b> </p>

			<p>of Bit (Note: Byte value only means one Bit value. For example: value H24 means ON status, H45 means OFF status)</p> <p>Code is showed as below:</p> <p style="padding-left: 40px;">DataAreaSubType = 0x01 OnValue = 0x24 OffValue = 0x45</p> <p>2. DataAreaSubType = 0x02</p> <p>For Xinje PLC, after panel send the command to read value of M10, the response Byte is read into buffer register, then the first bit is the value of M10. If the first bit is the M10, there is no need to shift, then <i>Group2Block4_ModNumber = 1</i>; if the Nth bit is the M10, shift is needed, then <i>Group2Block4_ModNumber = n</i>.</p> <p>The following is the code used for read the Coil M with Xinje PLC:</p> <p style="padding-left: 40px;">Group2Block4 = 10 Group2Block4_IsHightByteNext = 1 Group2Block4_DataAreaType = 5 Group2Block4_DataAreaSubType = 2 Group2Block4_ModNumber = 1</p>
6	Send/ receive data in normal way'		<p>This code is used to send/ receive data in normal way.</p> <p>For example: 3000 in decimal notation convert to Hex notation is BB8, then divided into High Byte-0B and Low Byte-B8, but here ,the normal way we use is : divided 3000 in decimal notation to 3 0 0 0, then convert into ASC format: 33 30 30 30;</p> <p>This normal way can be divided into two kinds:</p> <ol style="list-style-type: none"> <li>1. Dec , Unsigned, Hex ,</li> <li>2. Float <ul style="list-style-type: none"> <li>• Dec , Unsigned, Hex ,</li> </ul> </li> </ol> <p>Example 1: If the data read from one object device is time information: 31 31 30 31 34 32, so the time is:11: 01:42, but how to display this time on the panel? The code is showed as below:</p> <p>Group2Block7=10</p> <p>Define the 7<sup>th</sup> block in receive area is data area.</p>

			<p><b>Group2Block7_DataAreaType = 0x06</b>  Define this block to receive data in normal way.  <b>Group2Block7_DataSubType = 1</b>  Define the receive data to display in 'unsigned' format  <b>Group2Block7_FormatLen = 6</b></p> <p>Define the length of receive data is 6</p> <p><b>Example 2: NeedPreZero</b> plays an important role in 'write' operation, but seems no meaning in 'read' operation.  For example:  Write value 100 into one parameter of a object device, code is showed as below:  <b>Group1Block7 = 10</b>  Define the 7<sup>th</sup> block in receive area is data area.</p> <p><b>Group1Block7_DataAreaType= 6</b>  Define this block to receive data in normal way</p> <p><b>Group1Block7_DataSubType= 1</b>  Define the receive data to display in 'unsigned' format</p> <p><b>Group1Block7_FormatLen= 6</b>  Define the length of receive data is 6</p> <p><b>Group1Block7_NeedPreZero = 1</b>  Define : lead 0 or not( 0: no need, 1:need )  If <b>Group1Block7_NeedPreZero = 1</b>, then the send data is 30 30 30 31 30 30;  If <b>Group1Block7_NeedPreZero = 0</b> then the send data is 31 30 30  Obviously, the result is different.</p> <ul style="list-style-type: none"> <li>• <b>Float</b>  <b>Example:</b> if value read from an object device is <i>31 34 33 37 32 32 36 34 2E 31 30</i>, it means the value is <i>14372264.10</i>, so, how to describe this data area? The code is showed as below: <p><b>Group2Block18= 10</b>  Define the 18<sup>th</sup> block in receive area is data area.</p> </li></ul>
--	--	--	---

				<p><b>Group2Block18_DataAreaType = 6</b> Define this block to receive data in normal way</p> <p><b>Group2Block18_DataSubType = 3</b> Define the receive data to display in 'float' format</p> <p><b>Group2Block18_FormatType = 0</b> Define the output format as dddd.dddd; this item is only active for 'write' operation, not useful for 'read' operation.</p> <p><b>Group2Block18_Precision = 2</b> Define the decimal length is 2</p>
--	--	--	--	---

<b>Code name of function definition</b>	11	
<b>Meanings</b>	<p>Checkout code: this is used as checking program to get check result. When the code with checkout (there we call this checkout as checkout A) is entered into communication system, the checking program get the standard checkout (called checkout B), then compare these two checkout. If they are same, it shows the checkout A is correct and will be permitted into communication system, if not, enter the correct code again.</p> <p style="text-align: center;"><b>An example of CRC checkout:</b></p> <p style="text-align: center;"> <b>Group1Block7 = 11</b>  <b>Group1Block7_StartBlock = 1</b>  <b>Group1Block7_EndBlock = 6</b>  <b>Group1Block7_SelectMethod = 3</b>  <b>Group1Block7_HoldSpaceSize = 2</b> ;  <b>Group1Block7_InitValue = 0xffff</b> </p>	
<b>Extended definition</b>	<b>Group X Block Y _LowBitToASC</b>	Only remain low bit after ASC conversion
	<b>Group X Block Y _ IsHighByteNext</b>	For word, high byte in the first position with low byte followed.
	<b>Group X Block Y _ IsHighWordNext</b>	For Dword, high word in the first position with low word followed.
	<b>Group X Block Y _ StartBlock</b>	Symbol of start block, this block is included in count

	<b>Group X Block Y _ EndBlock</b>	<b>Symbol of end block, this block is included in count</b>
	<b>Group X Block Y _ SelectMethod</b>	<b>Checkout way: (0: Lrc; 1: Sum; 2: -Sum; 3: CRC_Modbus; 4: CRC_IBM; 5: CRC_ITU)</b> <b>Note: -sum check means the sum of all data and checkout is zero, and checkout is in Byte format.</b>

<b>Code name of function definition</b>	<b>13</b>
<b>Meanings</b>	<b>Incertitude block: this can be used for the uncertain block or indifference block.</b>

- **ObjectType : Object Type optimization**

This part is used to optimize the communication rules with less quantity of transmission data. For example: we want to deal with the continuous 5 registers in the panel, without optimization, we have to send 5 read messages each cycle, but if we use this function, in convert into read registers mode automatically and read values of 5 registers each time. The following part describes a series of optimized codes:

[ObjectType]

ObjectTypeNum = 2 ; numbers of optimize objects

[ObjectType0] ; Optimize object type 1 **【Note 1】**

IsConvert = 0 ; Convert to other object type or not ? (0: no need 1: need)

Regs = 1 ; Which communication is this object type belong ? In this example, use the first communication rule.

MaxLength = 32 ; **【Note 2】**

BitLength = 16 ; Bit length of this object type.

[ObjectType1] ; Optimize object type 2 **【Note 3】**

IsConvert = 1 ; Convert to other object type or not ? (0: no need 1: need)

ToObjectType = 0 ; convert to which object type? In this example, the value is zero  
TargetAddress = H+0x3000 ; Conversion expression of object address  
BitLength = 16 ; Bit length of this object type

Notation:

**【Note 1】**: From the 'Object type' part, we know that each object has description of object optimization, there is an example:

[Object1]

CanAct=1

IDSymbol=D

TypeNo=0 ;The matched 'object optimization' No. in this example it is ObjectType0

bitlength = 16;

DescripNum=1

CanSelectType=12

**【Note 2】** : It is advised to set the 'Maxlength' with appropriate value, because redundant object would lead to inadequate device addresses and faulty in data transmission.

**【Note 3】** : Obviously, the structure of ObjectType0 is easy to interpret, but why do we convert ObjectType1 to ObjectType0? There we take Xinje PLC for example to explain.

[Object1]

CanAct=1

IDSymbol=D

TypeNo=0

bitlength = 16;

DescripNum=1

CanSelectType=12

ReadWord = 1

ReadDWord = 1

WriteWord = 47

WriteDWord = 8

[Object2]

CanAct=1

IDSymbol=TD

TypeNo=1

bitlength = 16;

DescripNum=1

CanSelectType=12

ReadWord = 2

ReadDWord = 2  
WriteWord = 9  
WriteDWord = 9

----- some code is omitted-----

[ObjectType]  
ObjectTypeNum = 2

[ObjectType0]  
IsConvert = 0  
Regs = 1  
MaxLength = 32  
BitLength = 16

[ObjectType1]  
IsConvert = 1  
ToObjectType = 0  
TargetAddress = H+0x3000  
BitLength = 16

Please note the red part of above codes, as we can see, the common registers D is matched with object optimization type 0, but time registers TD is matched with object optimization type 1.

Although there is difference between these two type registers in the functions of PLC program, but in the view of physics space, they are completely same, more details can refer to Xinje XC series PLCs- communication based on Modbus protocol.

Then we find that the Modbus address of D0 is 4x0, and TD0 is 4x3000, but the data structure of them are same during the transmission ,in other word ,on the layer of data transmission with devices, they are only have difference in address, thus, object type conversion is needed.

## ■ Expression

Eg.  $0x1600+N/8$

arithmetic: +, -, \*, /, %, (,), <<, >>, &, |

digital type: Dec without sign ( 123 ), Hex ( 0xe1; 0XE1 )

Note: don't support the expression of negative, such as -123

Special symbol :N(read or write the 1st description value); M(read or write the 1st description value),R (group quantity of registers;P(response data from 'data area')

Note: capials are not distinguished.

Levels of arithmetic (from high to low) :

The 1st level: ( , )

The 2nd level: \* , / , %

The 3rd level: + , -

The 4th level: << , >>

The 5th level: &

The 6th level: |



**XINJE Electric Co., Ltd.**

4th Floor, Building 7th, No.100 Dicui  
Rd, Wuxi, China

Tel: 86-0510-85134139

Fax: 86-0510-85111290

[www.xinje.com](http://www.xinje.com)

Email: [cheerfiona@gmail.com](mailto:cheerfiona@gmail.com)