Product introduction

PLC is a versatile high-performance programmable logic controller, which is widely used in plastics, packaging, textiles, food, medical, pharmaceutical, environmental, municipal, printing, building materials, elevators, central air conditioning, numerical control machine tools and other fields of systems and control equipment. In addition to its own various peripheral interfaces (digital input, digital output, analog input, analog output, high-speed counter, high-speed pulse output channels, power supply, communication ports, etc.), it can also extend all types of extension modules for flexible configuration.

PLC programming software is a programming software which is in accordance with IEC 61131-3 standard. It can be used for PLC programming. Furthermore, it supports three kinds of programming languages--LD (Ladder Diagram), FBD (Function Block Diagram) and IL (Instruction List). It can run on the systems of Win98, Win200X, WinXP, Win7, Win8 and the later versions.

PLC features

Ethernet + : Host PLC and remote modules support Ethernet communication, host PLC support Ethernet port and 5 other RS232/RS485 communication ports working simultaniously,support N:N network type,support remote programming,debug, monitoring and data exchange. Easy to network with other PLC modules,HMI and PC via Ethernet port.

The firmware upgrade function: Taking the lead in the function of realizing firmware upgrade in a small programmable controller. You can upgrade the system firmware through the firmware upgrade function free,therefore you bought the products previously can also have new features.

Rich network communication function: CPU host built two communication ports, which can be expandable to five communication ports, each port can be programmed and connects to network, and all of them can be used as masters or slaves. It can support 1: N, N: 1, N: N networking and a variety of man-machine interface and configuration software. It can also connect to network with any third-party devices which have communication capabilities (such as inverters, instruments, barcode readers, etc.).

Supporting for multiple communication protocols: It has internally installed Modbus RTU / ASCII protocol, Modbus TCP protocol, freedom communication protocol and the Speedbus, Speedbus TCP high-speed communication protocols to the most convenient communication instruction system, no matter what kind of communication protocols, it only simply needs a communication instruction when dealing with complex communication functions.You will no longer be troubled by the problems, such as communications port's conflicts, sending and receiving control, communications interrupt handling issues and you can use a variety of protocols to exchange data easily by mixing them up in the program.

High-speed pulse counting function: Supports 8-channel duplex high-speed (200KHz) pulse counting, counting mode supports 7 kinds (pulse / direction 1 octave, pulse / direction 2 octave, forward / reverse pulse 1 octave, forward / reverse pulse 2 octave, A / B phase pulse 1 octave, A / B phase pulse 2 octave, A / B phase pulse 4 octave), and three kinds of comparisons (single-stage comparison, the absolute mode comparison, the relative mode comparision), supports 8 segments comparision fixed value, with self-learning function.

High-speed pulse frequency measurement: Supports 16-channel (200KHz) high-speed pulse frequency measurement, support the ways of time or pulses to measure the frequency

High-speed pulse output: Supports 8-channel duplex high-speed (200KHz) pulse output, support for acceleration and deceleration pulse output, multi-segment envelope pulse output function, a unique sync pulse output function makes it easy to achieve precise synchronization control. Stand-alone support 16-channel pulse width modulation (PWM), can drive 16 servo or stepper motors.

Motion control function: Each model support for 8-channel(200KHZ) motion control, supports arbitrary 2-channel linear interpolation, circular interpolation, support follower pulse output, absolute address, relative address, backlash compensation, original point return, definition of electrical origin.

Edge capture and interrupts: CPU supports 8-channel up and down along the catch and interrupt functions, all digital inputs support signal filtering settings, all digital outputs remains set to support power output. Provide 52 real-time interrupt.

1. **Powerful analog processing function:** AI register accesses the analog input directly, analog input support engineering conversion, sampling frequency settings, and zero correction. Available AQ registers control the analog output directly, analog output support engineering conversion and can be configured to maintain output.

2. **Strong password protection function:** Three levels of password protection function (program files password, each block password, PLC hardware password) and prohibits the application to upload.

3. **Self-diagnostic function, power failure protection function, calendar (RTC), floating point operations, etc.**

Programming software features

1. Internal PLC simulator: PLC programming software realize the PLC program run in the simulation. During programming or the programming is completed, you can run PLC program in the simulation without online to check the program execution is correct or not. It can reduce on-site commissioning time greatly, reduce debugging difficult and improve debugging efficiency.

2. Communications simulator: It is used to the debug communication instruction simulation tools. It can be manually input simulately response message returned from salve, or you can use the computer's serial port to communicate with salve really, Simulate the process that PLC executes communication instruction really and process the return data from the salve.

3. Interpolation simulator: Track and draw the trajectory generated from motion control instructions such as the linear interpolation, circular interpolation, listing parameters of the pulse output channel of the motive plane and corresponding to each axis, display the current position of the channel, the

mechanical home position, output mode, you can set shaft length, unit pulses.

4. Function of generate PLC executable file: PLC program can be generated to executable file which is released and executed independently. So you do not need to send the PLC program to the user, it can be very easy, very safe to put the PLC excutable file to the user to download, but do not worry the user would can see the program content.

5. Facilitate innovation instruction set: On the basis of analysis and absorption of various PLC instruction, PLC launched many powerful innovations facilitate instruction. As communication instruction (COMM, MODR, MODW, HWRD, HWWR), data portfolio diversification instruction (BUNB, BUNW, WUNW, BDIB, WDIB, WDIW), PID control (PID), valve control (VC), upper and lower alarm (HAL , LAL), range transmitter (SC), temperature curve (TTC) ,etc. Any one instruction can realize the function but other PLC required to multiple instructions. These instructions are very easy to understand and use, greatly improve the programming efficiency and running speed.

6. Modular project structure: Create 31 blocks total (main program, sub program, interrupt program) and chose any programming language to program. The execution order of block can be adjusted at random. Each block can be imported and exported independently and has the same password protected of program projects.So we can fully realize the modular programming and program reuse dreams.

7. Instruction using table: Provides multiple instruction tables. Use these tables can reduce the amount of programs, saving program space, such as initialization data. Each table can be imported and exported independently and has the same password protection of program project**.**

8. Powerful online features: Search out all the PLC that connect with the PC. Show running status, fault status, RUN / STOP switch position, hardware configuration information, communication port parameters such detailed information of all

the online PLC. Select any PLC for online monitoring, program download, firmware upgrade, controlling PLC stop, adjusting PLC real-time clock, modifying password protection, modifying communication port parameters, modifying the watching-dog time and PLC station names.

9. Online monitoring and debugging functions: Provide 10 pages of component's monitoring table. It can choose in decimal, hexadecimal, binary, floating point and character to display data. Support component and register component monitoring hybridly and displaying component annotation at the same time. All instruction tables can be imported to the monitoring table.

10. Unique real-time curve function: Monitor any register component for real-time curve to debug process-control facilitately.

11. Humane input: Provide shortcuts, drag and drop, click and many other command input. Suggest effective components or range of values for each input and output terminals. It can be entered directly. Some data of combination (such as communication protocols etc.) can also double-click the instruction to configure the input data.

12. Convenient annotation: Provide the component comment, network comment, instruction comment, block comment, table comment, and project comment. After the component with "//" to input comments directly (eg: X0 // motor start). Comments can choose to download to the PLC for reading or modification facilitately.

13. Detailed tips and online help: Provide PLC resource window, instruction window, etc. All the instructions and detailed description of hardware modules can be found in powerful online help system which is open through clicking F1 key in the programming interface to find the answer. Even if use PLC programming software for the first time who can easily complete the preparation of control program.

14. Convenient editing functions: Support all conventional editing operations, searching and replacing, instruction up and

down, network up and down, copying and pasting between program projects.

15. Hardware configuration, sub program parameter passing, local components, indirection, print, preview, debugging, CRC calculation, password protection, etc.

Quick start

This section brief introduce general step of programming PLC control program , to help beginner as soon as possible know well PLC programming software programming operation.

General step of programming

First step:double click the programming software icon at computer desktop ,start programming software.

Second step: new construction a program project, use menu[file/ new construction program project],open " new construction program project" window, select PLC series. PLC MPU and others project attribute .

Third step: write control program ,build one or many program block, use menu [File/New...../master program block],open " New construction program block" window ,input the program block named . select programming language (LD. FBD. IL)and program block type (program block. subprogram. interrupt program).

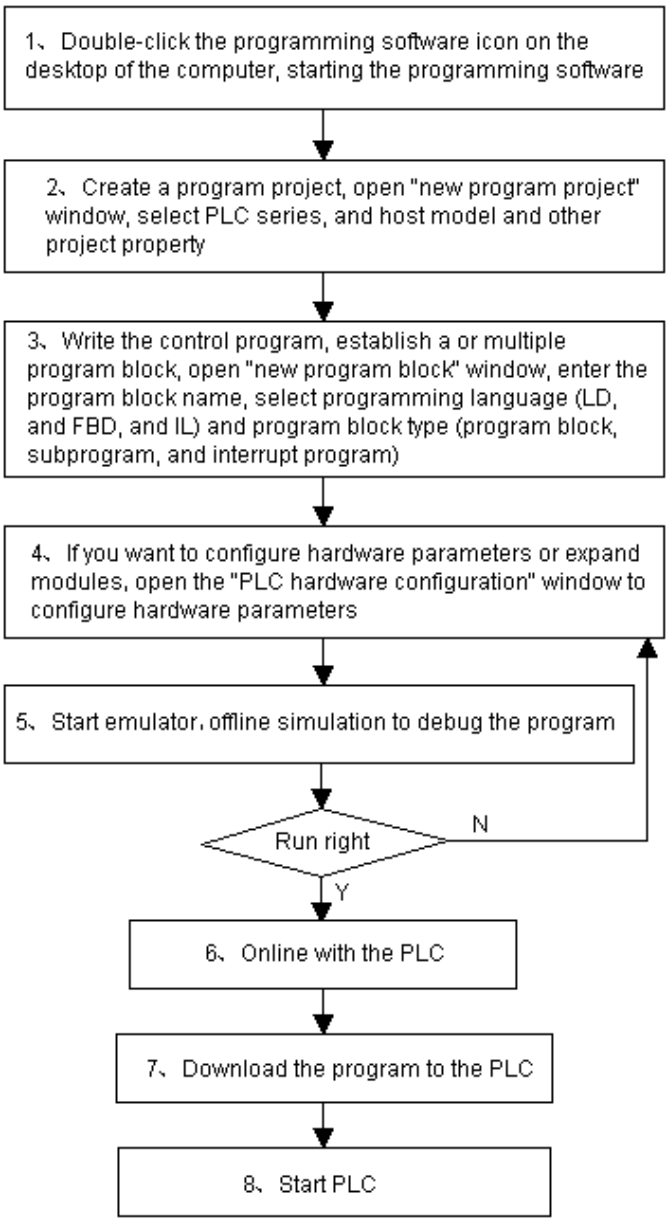
Fourth step: if need configuration hardware parameter or extend module ,Use menu [Look UP/PLC hardware configuration] open "PLC hardware configuration " window.

Fifth step: Start simulator,off line simulate running debug program, running correct then next step, otherwise return to third step modify program .

Sixth step: Online with PLC.

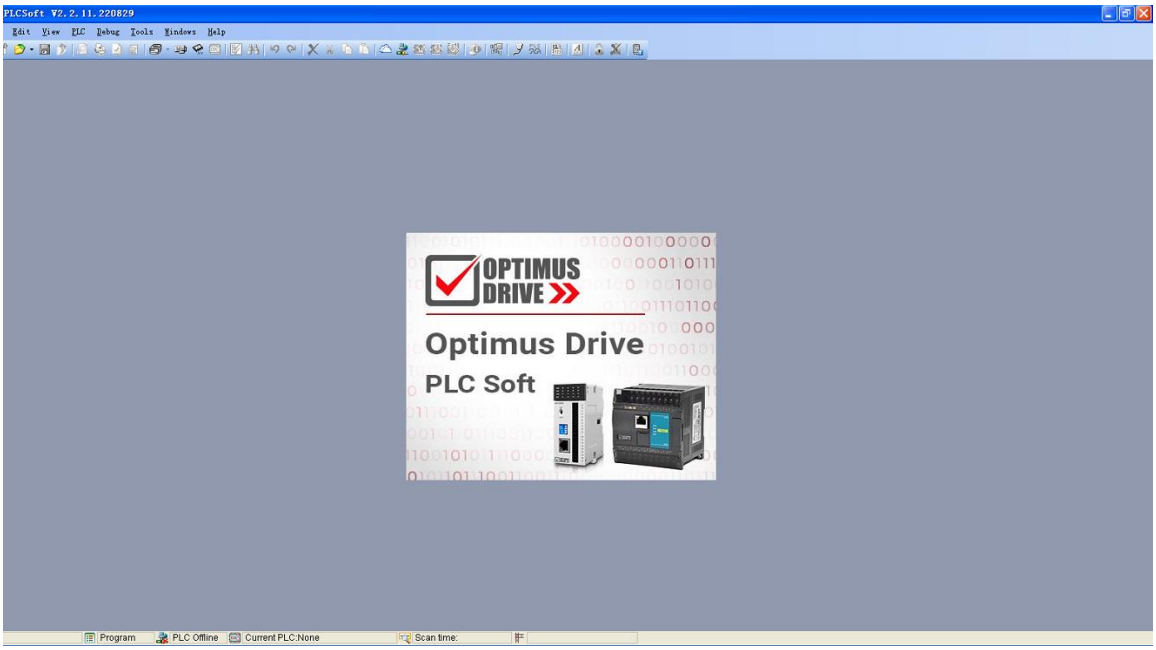
Seventh step :download program to PLC.

Eighth step :start PLC running.



First step:Start programming software

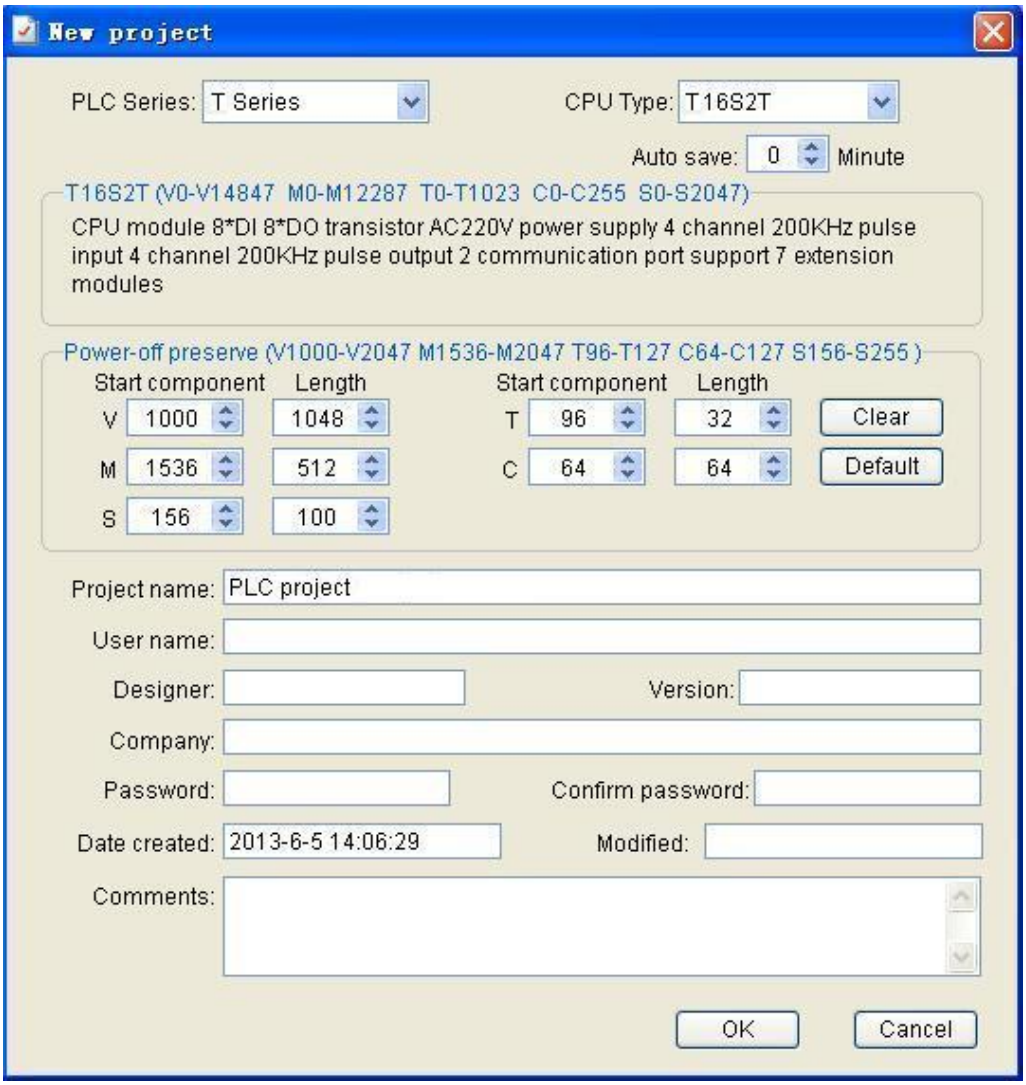
double click the programming software icon at computer desktop, start programming software, as follows:



Second step:New project

Open mode: A. Click menu [File /New project], B. Click tools bar button, C. use shortcut key Ctrl+N.

"New project " window as follow:



At "PLC series" among pull listing select "PLC series".

At "CPU type " among pull listing select "CPU also MPU type".

Power-off preserve area may be defined by user, may be set V. M. S. T. C five type component power-off preserve area start component and length. System default the Power-off preserve area listing as follow:

Component type	Power-off preserve area	Component number
V	V1000~V2047	1048
M	M1536~M2047	512
S	S156~S255	100
T	T96~T127	32
C	C64~C127	64

At " project name " input new project name, the project name will be display in the main frames of the project manager .

At "user name". "designer". "version". "company" etc. columns input relate to information.

If need password protect the new project, then at "password" and "verify password" input the password.

At "note" may be input relate to note information of the project.

Third step: Write control program

Control program realize the kernel of automatic control ,it essence is according to control object (as machine. automatic equipment. production line etc.) requirement details use programming software supplied many instructions realize automatic control of the control object.

According to the need build one or more program block, choice oneself well-informed program language (LD. FBD. IL), each block may be set alone password , realize local cypher function.

At program block use programming software supplied many instructions to program , realize all kinds of control logical and arithmetic.

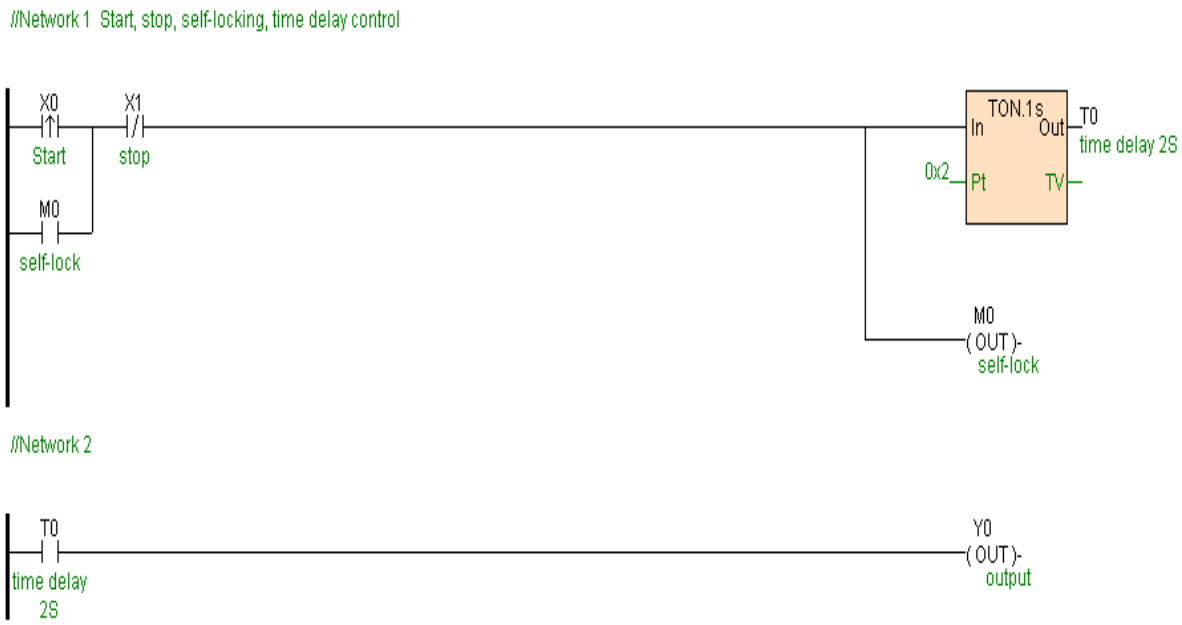
After complete the program, save program file, start simulator running, debug program whether attain the control requirement.

Detail operation refer to "programming operation manual "

[Example]

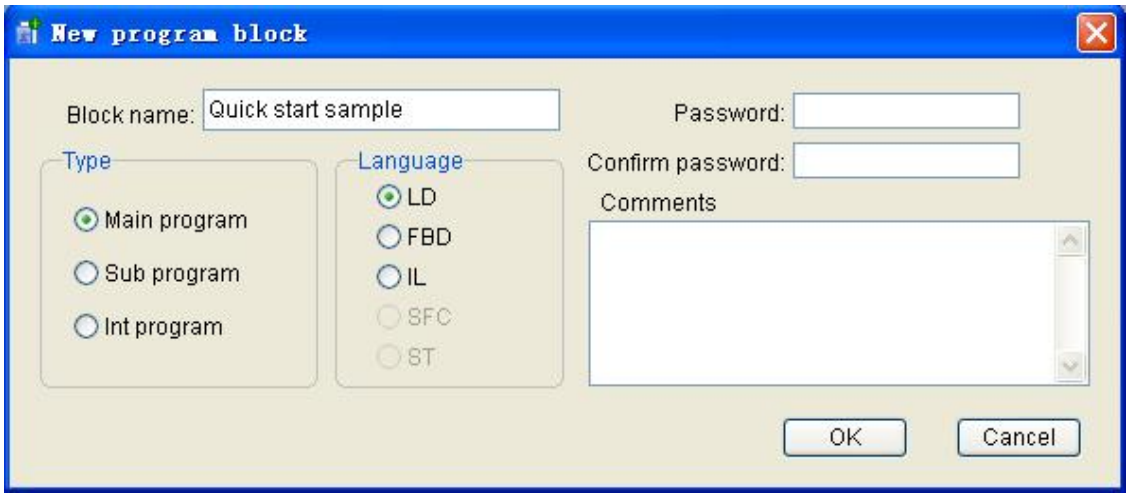
Below explain how to write the control program via the example, the example control requirement: press X0 button start, delay 2 second output Y0. press X1 button stop.

[Example program] Download



[Example program operation]

Create a new " Quick start example" main program block, open mode:A. Click menu [File/New...../Main program block],B. Click tools bar button "Main program block"." Create new program block " window as follow.



Click tool bar  button or press F9 shortcut key add a switch, input switch component "X0//start" (behind "//" " start " is the comment of component X0 ,the same below), press Ctrl+3 shortcut key change normal open to rising edge.

Press F10 shortcut key parallel a switch, input switch component "M0//self-lock"

Move cursor to the right side of X0 switch ,press F9 shortcut key series a switch, input switch component "X1//stop", press Ctrl+2 shortcut key change normal open to normal close.

Move cursor right, input "TON" return, add one "TON" instruction, continuous press enter, when input box at "Pt" item input set value 2 of the timer.

Press F11 shortcut key add parallel output instruction "OUT", input component "M0".

Press tool bar  button or press Ctrl+L shortcut key add a new network.

Press F9 shortcut key add a switch, input switch component "T0//delay 2S".

Press F11 shortcut key add output instruction "OUT", input component "Y0//output".

Come here program wrote complete. press Ctrl+S shortcut key to save the program file, may be start the simulator executing, debug program whether attain control requirement .

Fourth step:PLC hardware configure

If you need configure hardware parameter (as configure AI input channel signal type . quantities etc.) or need add extend module etc. relate to hardware, then do this step, otherwise skip this step .

Double click " Project manager " window directory tree "PLC hardware configure " point or use menu [Check/PLC hardware configure] to open "PLC hardware configure" window.

Click open "Project manager" window directory tree " PLC ,select want to add module , use mouse drag the module to right side hardware configure list.

Click hardware configure list module ,under the list will display the attribute of the module be defined and configured.

Define and configure the attribute of each module.

Detail operation refer to "PLC hardware configure ".

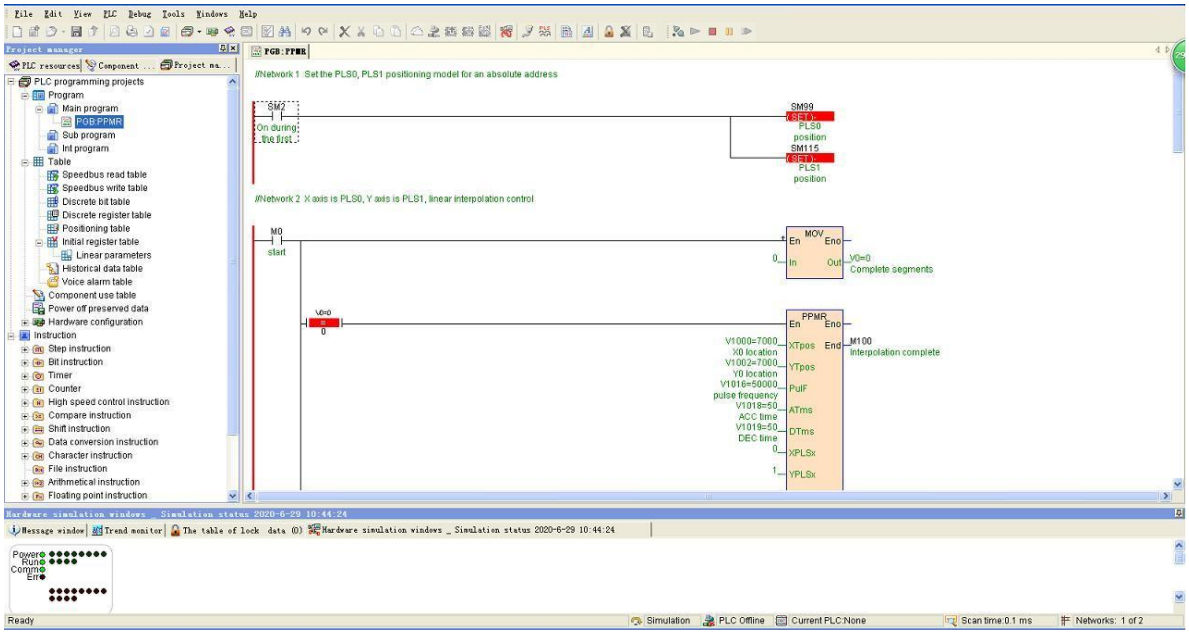
Fifth step:Off line simulate debug

During write the program or after complete the program , may be use simulator under the condition of completely off line PLC simulate running PLC program, use for check program executed whether correct, vastly reduce local debug time ,reduce debug difficulty, improve debug efficiency.

Click menu [Debug/Start simulator]or click tools bar  button start the simulator, system will automatic compile the current program project. Detail operation of the simulator refer to "Simulate and on line debug".

If compile the program error, simulator can not running, user must modify the program according to the compiler suggestive error information.

Compile no error or only alarm, then start simulator .simulator interface as follows:



Double click program "X0" switch force X0=ON, then M0=ON self lock, timer T0 start timing, when TV0=2 time time to then T0=ON,Y0=ON.

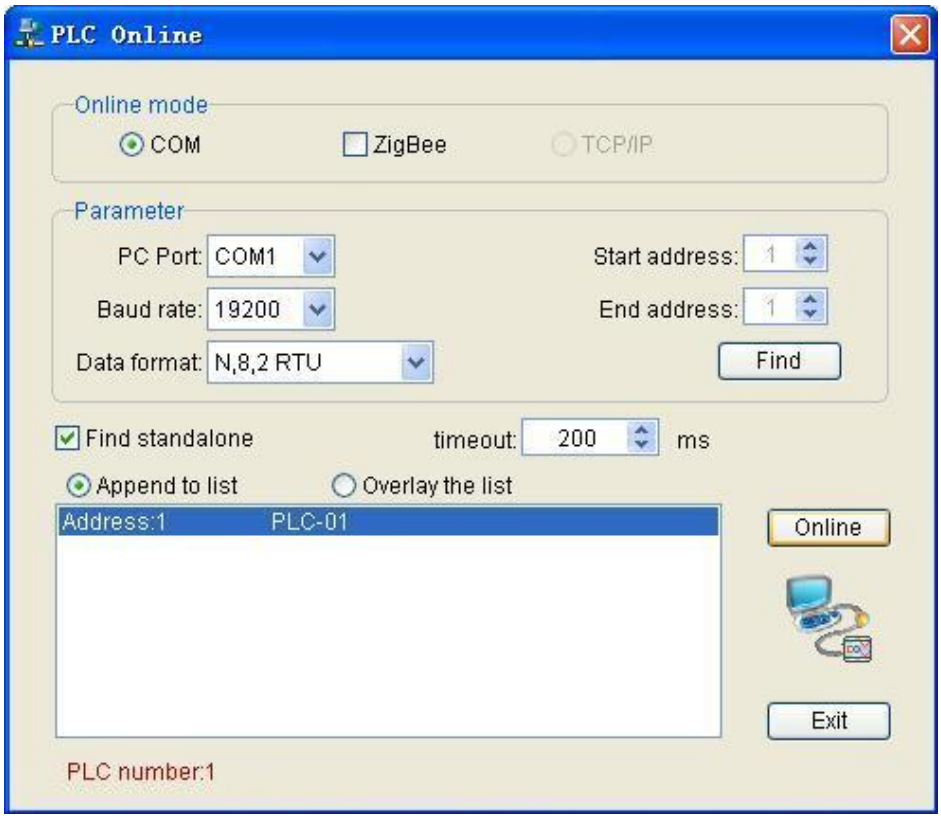
Double click program "X1"switch force X1=ON,X1 normal close switch no electricity, then M0=OFF,T0=OFF,Y0=OFF.

Via the simulator running verify, program running result correct.

Sixth step:Online with PLC

Connect to one or many PLC in the network . Only after PLC online may be control operation the PLC, such as: upload or download etc..

Click menu [PLC/PLC networking] or click tool bar  button , open "PLC online" window.



Click "Online" button (general condition, direct use default parameter), already online PLC will automatic add to the listing box, now click "exit " close the window.

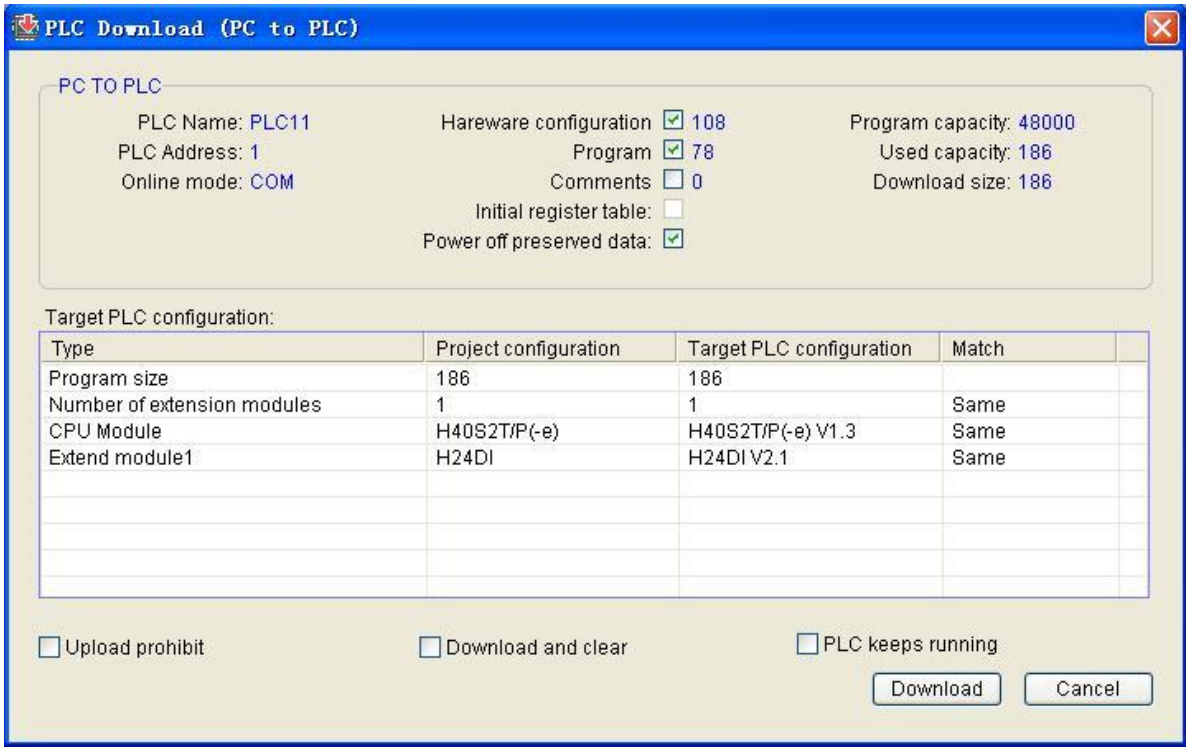
PLC online and parameter setting refer to "Online control PLC" section "PLC online".

Seventh step:Download program

Download current program project (hardware configure and program etc. content) to online PLC. Before download system automatic compile

the current program project, if error during compile, then list all errors, after user modify the program no any error then can be downloaded.

Click menu [PLC/PLC program download] or click tool bar button , open "PLC program download" window, click "download " button download the program to PLC.



Note: detail operation refer to " Online control PLC" section "Download program".

Eighth step:Start PLC

After download complete, if PLC already in running status (RUN indicator go on), may skip the step. Otherwise turn the PLC running switch in to "RUN" position.

PLC Register and Data

Overview

PLC have configurated many component in the system (also call: variable. address),as:X. Y. M. AI. AQ. V etc..Components have Bit register and Word register,bit register occupy one bit express boolean variables ,word register occupy one word (16 bits,2 bytes) express data variables,they can be used again and again in program.In PLC use the soft component to operated in the program for arithmetic and control function.

Data

1. Data type

Data type	Explain	Data length	Range
BOOL	bit	1 bit component	1(ON). 0(OFF)
INT	integer with sign	16 bits,1 register component	-32768~32767
DINT	long integer with sign	32 bits,2 register components	-2147483648~2147483647
REAL	floating point	32bits,2 register components	-3.402823e+38~3.402823e+38
CHAR	character string	1 character occupy one byte	

2. Component matching data type

Data type	Component type																
BOOL		X	Y	T	C	M	SM	LM	S								
INT	constant									AI	AQ	TV	CV	V	LV	SV	P
DINT	constant											TV	CV	V	LV	SV	
REAL	constant													V	LV		

CHAR	constant												V	LV		

3. Constant

Constant type	Example	Valid range	Declare
16 bits integer with sign	1234. -7890	-32768~32767	
32 bits integer with sign	12345678. -9876543	-2147483648~2147483647	
16 bits constant in hexadecimal	0x2EF8. 0x9A12	0x0~0xFFFF	
32 bits constant in hexadecimal	0xA76DCFE9	0x0~0xFFFFFFFF	
floating point constant in single precision	3.1415926. -0.02341	-3.402823e+38~3.402823e+38	accord with IEEE-754

4. Storage and use for 32 bits data

1. The storage pattern in the register of 32 bits data :the data type of DINT. REAL are 32 bits length,but one register occupy 16 bits length,so need 2 continuous address registers for store 32 bits data.When store 32 bits data,according to fist low word. after high word ,e..g.:32 bits long integer data 0xA76DCFE9 store in V0V1 registers,then 0xCFE9 stored in V0,0xA76D stored in V1.
2. Register component can stored integer. long integer,if the operands in instructions is register,then the component is 16 bits integer. 32 bits long integer or floating point,is according the data type of the operand in the instruction.e.g.:"MOV -23, V10",the MOV is 16 bits integer move instruction,the register V10 is a 16 bits integer;"D.MOV 7891223, V10",the D.MOV is 32 bits integer move instruction,the register V10 is a 32 bits long integer (occupy V10V11);"FMOV -9.223, V10",the FMOV is floating point move instruction,the register V10 is a floating point (occupy V10V11).
3. The operand of floating point instruction is floating point,the operand of other instruction is integer(16 bits integer,32 bits long integer),data type converted with convert instruction.

Component overview

1. PLC bit component

Component	Name	Range	Read/Write attribute	Declare
X	External input relay	X0~X1023	read	
Y	External output relay	Y0~Y1023	read/write	
M	Auxiliary relay	M0~M12287	read/write	default power-off preserve:M1536~M2047,512 point
T	Timer	T0~T1023	read/write	time base of T0~T251. T256~T1023 can be set 10ms. 100ms. 1s default power-off preserve:T96~T127,32 point time base of T252~T255 is 1ms
C	Counter	C0~C255	read/write	default power-off preserve:C64~C127,64 point
SM	System status bit	SM0~SM215	all be read/some be wrote	
S	Step relay	S0~S2047	read/write	default power-off preserve:S156~S255,100 point
LM	Local relay	LM0~LM31	read/write	function in local area (program block. sub program. interrupt program)

2. PLC register component

Component	Name	Rang	Read/Write attribute	Declare
AI	Analog input register	AI0~AI255	read	
AQ	Analog output register	AQ0~AQ255	read/write	
V	Internal data register	V0~V14847	read/write	default power-off preserve:V1000~V2047,1048 point
TV	Current value of timer	TV0~TV1023	read/write	time base of T0~T251. T256~T1023 can be set 10ms. 100ms. 1s default power-off preserve:TV96~TV127,32 point time base of T252~T255 is1ms
CV	Current value of counter	CV0~CV255	read/write	the CV register is a 16 bits register,in CV48~CV79 register are 32 bits register,default power-off preserve:CV64~CV127 ,64 point

Component	Name	Rang	Read/Write attribute	Declare
SV	System register	SV0~SV1 54	all be read/some be wrote	
LV	Local register	LV0~LV31	read/write	function in local area (program block. sub program. interrupt program)
P	Indexed addressing point	P0~P29	read/write	special register use in indexed addressing

Note:power-off preserve range of T(TV). C(CV). M. S. V in the table are default configure of the system,the range can be changed by user.

External input relay [X]

External input relay X:Corresponds to external input points(e.g.:Switch. Button etc.), attain the external input point state (ON or OFF) access to PLC.

Sign by X,e.g.:X0. X1. … X8. X10. X11. …. Identifiered from X0,In "PLC hardware configure" to configure.Automatic assign the address by the system.

External output relay [Y]

External output relay Y:Corresponds to external output points,use for drive the external output points Y ON-OFF (ON or OFF).One Y relay can be reuse of output in the program ,but recommendation use only once, in order to improve the reliability and readability of the program.

Sign by Y,e.g.:Y0. Y1. … Y8. Y10. Y11. ….Identifiered from Y0,In "PLC hardware configure" to configure.Automatic assign the address by the system.

Timer [T]

Timer T:each timer compose output relay T and current value TV.

Target = time base * set value.Time base of T0~T251. T256~T1023 can be set 10ms. 100ms. 1s,time base of T252~T255 is 1ms.

In one program,each timer can be used only once,But the output relay T and current value TV of the same No.can be used unlimited.

Timer can be devided power-off preserve and no power-off preserve:the output relay T and current value TV of the no power-off preserve timer will be reset when PLC STOP .The output relay T and current value TV of the power-off preserve timer will be retain when PLC STOP.

Counter[C]

Counter C:each counter compose output relay C and current value CV.

In one program,each counter can be used only once,But the output relay C and current value CV of the same No.can be used unlimited.

Counter can be devided 16 bits counter and 32 bits counter, among CV48~CV79 are 32 bits counter,rest are 16 bits counter.

Counter can be devided three type in accordance with the count mode:increase count CTU. decrease count CTD. increase/decrease count CTUD.

Counter can be devided power-off preserve and no power-off preserve :the output relay C and current value CV of the no power-off preserve counter will be reset when PLC STOP .The output relay C and current value CV of the power-off preserve counter will be retain when PLC STOP.

Auxiliary relay[M]

Auxiliary relay M:Used for internal logic operation,we can use the combination of auxiliary relays in control logic,but can not drive the load direct.

M can be devided power-off preserve and no power-off preserve :the state of no power-off preserve M will be reset when PLC STOP .The state of power-off preserve M will be retain when PLC STOP.

Step relay[S]

Step relay S:Used for step control instruction,each step relay stand for one step,can be used the same as auxiliary relay if there is not step instruction in the program .

S can be devided power-off preserve and no power-off preserve :the state of no power-off preserve S will be reset when PLC STOP .The state of power-off preserve S will be retain when

PLC STOP.

System status bit [SM]

System status bit SM is a group of special internal relay of the system,can be used unlimited in the program, each SM has a special function.

See detail "SM System status bit"

Local relay [LM]

Local relay LM:they are special internal relay which function in local area (program block. sub program. interrupt program) ,different with the internal relay M.

The function range of M:in the all program in the project,all of the scan cycle.

The function range of LM:in local area (program block. sub program. interrupt program) ,state (ON/OFF) only keep one scan cycle ,state will be reset when the next scan cycle started,state reset to OFF,general use for temporary variable or the parameter of the program block be called.

Analog input register [AI]

Analog input register AI:each analog input channel corresponding one analog input register AI,analog input channel connect to external device,used for measure the continuous change of the external singal ,e.g. temperature . pressure. quantity of flow etc..

When the signal of external analog input channel be changed,can be reflect to the analog input register AI immediately.

The signal type. quantities. up and down rang of quantities of each analog input channel can be configured in "PLC hardware configure".Automatic assign the address by the system.

Analog output register [AQ]

Analog output register AQ:each analog output channel corresponding one analog output register AQ,analog output channel connect to external device.

When the value of analog output register AQ be changed, corresponding the signal of the analog output channel be changed immediately.

The signal type. quantities. up and down rang of quantities of each analog output channel can be configured in "PLC hardware configure".Automatic assign the address by the system.

Current value of timer [TV]

Current value of timer TV: indicate the keep time of the timer.

See detail "Timer T"

Current value of counter [CV]

Current value of counter CV:indicate the current count of the counter.

See detail "Counter C"

Internal data register [V]

Internal data register V:for store data ,16 bits register(b0~b15),integer can express the range -32768~+32767,2 continuous 16 bits register is 32 bits register,long integer can express the range -2147483648~+2147483647,float point can express the range -3.402823e+38~3.402823e+38.

V can be devided power-off preserve and no power-off preserve :the value of no power-off preserve V will be reset to 0 when PLC STOP .The value of power-off preserve V will be retain when PLC STOP.

System register [SV]

System special register SV is a group of special internal register of the system,can be used unlimited in the program, each SV has a special function.

See detail "SV system register"

Local register [LV]

Local reggister LV:they are special internal register which special function in local area (program block. sub program. interrupt program) ,not the same as internal register V.

The function range of V:in the all program in the project ,all of the scan cycle.

The function range of LV:in local area (program block. sub program. interrupt program) ,value only keep one scan cycle ,value will be reset to 0 when the next scan cycle started,general use for temporary variable or the parameter of the program block be called.

Indexed addressing point [P]

Indexed addressing point P:special register use in indexed addressing .

How to input the register indexed and representation in the program:base address of the register+indexed addressing point P.e.g.:V100P6 presentation the base address is V100,use the value of P6 as excursion value for indexed addressing,as if the value of P6 is 10,indicate access the register is V110.

Reality access register =base address of the register + excursion of indexed address point.

Note:

1. When indexed addressing use the P,If base address of the register + excursion of indexed address point (reality access register)exceed the uplimit ,it will be error of the indexed addressing instruction,the instruction cannot be execute.

2. LM, LV, S, P component type does not support indexing

3. Part of the instruction does not support indexing:RESH. HHSC. HCWR. SPD. PWM. PLSY. PLSR. ZRN. SETZ. PPMR. CIMR. SPLS. SYNP. COMM. MODR. MODW. HWRD. HWWR. RCV. XMT. SORT. ENO

PLC instruction set

PLC have a set of abundance high-efficiency instruction system, depend on absorb instructions of others PLC , support up to 200 application instructions , among there are many powerful innovate easy instructions .as commucation instructions (COMM. MODR. MODW. HWRD. HWWR). character conversion instructions (ITOC. CTOI. FTOC. CTOF). data combination disperse instructions (BUNB. BUNW. WUNW. BDIB. WDIB. WDIW). bound alarm instructions(HAL. LAL). valve control instructions(VC). temperature curve(TTC) etc.

Instruction set table as follows:

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
Compare switch	=	LB.= HB.=	D.=	Equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	<>	LB.<> HB.<>	D.<>	Unequal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	>	LB.> HB.>	D.>	Greater than compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	>=	LB.>= HB.>=	D.>=	Great than or equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	<	LB.< HB.<	D.<	Less than compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	<=	LB.<= HB.<=	D.<=	Less than or equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
	F.=			Floating-point number equal to compare switch	√		
	F.<>			Floating-point number unequal to compare switch	√		

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	F.>			Floating-point number greater than compare switch	√		
	F.>=			Floating-point number greater than or equal to compare switch	√		
	F.<			Floating-point number less than compare switch	√		
	F.<=			Floating-point number less than or equal to compare switch	√		
Step instruction	STL			Step start	√		
	SFROM			Step combine	√		
	STO			Step jump	√		
Bit instruction	AND			Logic AND		√	√
	OR			Logic OR		√	√
	XOR			Logic XOR		√	√
	OUT			Coil output	√	√	√
	SET			Setting	√	√	√
	RST			Reset	√	√	√
	ALT			ON/OFF alternately output	√	√	√
	ZRST			Batch reset	√	√	√
	ENO			Get ENO output			√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
Timer	TON			Delay ON	√	√	√
	TOF			Delay OFF	√	√	√
	TP			Pulse timer	√	√	√
Counter	CTU		D.CTU	Increase counter	√	√	√
	CTD		D.CTD	Decrease counter	√	√	√
	CTUD		D.CTUD	Increase and Decrease counter	√	√	√
High speed control instruction	SHC			Single high speed counter	√	√	√
	RESH			IO refresh	√	√	√
	HHSC			High speed counter	√	√	√
	HCWR			Write high speed counter	√	√	√
	SPD			Speed detection	√	√	√
	PWM			Pulse width modulation	√	√	√
	PLSY		D.PLSY	Pulse output	√	√	√
	PLSR		D.PLSR	Accelerate and decelerate pulse output	√	√	√
	ZRN			Origin point return	√	√	√
	SETZ			Set electric origin point	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	PPMR			Linear interpolation	√	√	√
	CIMR			Circular interpolation	√	√	√
	SPLS			Single pulse output	√	√	√
	SYNP			Synchronization pulse output	√	√	√
	PSTOP			Stop pulse output	√	√	√
Compare instruction	CMP		D.CMP	Compare instruction	√	√	√
	ZCP		D.ZCP	Regional comparison	√	√	√
	MATC		D.MATC	Numerical match	√	√	√
	ABSC		D.ABSC	Absolute cam comparison	√	√	√
	BON			ON bit determine	√	√	√
	BONC		D.BONC	ON bit numbers	√	√	√
	MAX		D.MAX	Maximum	√	√	√
	MIN		D.MIN	Minimum	√	√	√
	SEL		D.SEL	Selection of conditions	√	√	√
	MUX		D.MUX	Multi-choice	√	√	√
Shift instruction	LBST			Low byte evaluation	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	HBST			High byte evaluation	√	√	√
	MOV		D.MOV	Move	√	√	√
	BMOV			Block move	√	√	√
	FILL			Fill	√	√	√
	XCH			Byte swap	√	√	√
	BXCH			Block swap	√	√	√
	SHL			Bit left shift	√	√	√
	SHR			Bit right shift	√	√	√
	WSHL			Word left shift	√	√	√
	WSHR			Word right shift	√	√	√
	ROL			Bit rotate left shift	√	√	√
	ROR			Bit rotate right shift	√	√	√
	WROL			Word rotate left shift	√	√	√
	WROR			Word rotate right shift	√	√	√
	BSHL			Byte left shift	√	√	√
	BSHR			Byte right shift	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	ATBL			Append to array	√	√	√
	FIFO			First in first out	√	√	√
	LIFO			Last in first out	√	√	√
	SORT			Data sort	√	√	√
Data conversion instruction	ENCO			Encoder	√	√	√
	DECO			Decoder	√	√	√
	BTOW			Bit convert to word	√	√	√
	WTOB			Word convert to bit	√	√	√
	HEX	HEX.LB		ASCII convert to hexadecimal	√	√	√
	ASCI	ASCI.LB		Hexadecimal convert to ASCII	√	√	√
	BUNB			Discrete bit combination to continuous bit	√	√	√
	BUNW			Discrete bit combination to continuous word	√	√	√
	WUNW			Discrete word combination to continuous word	√	√	√
	BDIB			Continuous bit disperse to discrete bit	√	√	√
	WDIB			Continuous word disperse to discrete bit	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	WDIW			Continuous word disperse to discrete word	√	√	√
	BCD		D.BCD	BIN convert to BCD	√	√	√
	BIN		D.BIN	BCD convert to BIN	√	√	√
	ITOL			Integer convert to long integer	√	√	√
	GRAY			BIN convert to GRAY code	√	√	√
	GBIN			GRAY code convert to BIN	√	√	√
Character instruction	GHLB			Obtain high low byte	√	√	√
	GETB			Capture byte string	√	√	√
	BCMP	BCMP.LB		Byte string comparison	√	√	√
	ITOC		D.ITOC	Integer convert to character	√	√	√
	CTOI			Character convert to integer	√	√	√
	FTOC			Floating point convert to character	√	√	√
	CTOF			Character convert to floating point	√	√	√
Arithmetical instruction	WNOT		D.WNOT	Negation	√	√	√
	WAND		D.WAND	AND operation	√	√	√
	WOR		D.WOR	OR operation	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	WXOR		D.WXOR	XOR operation	√	√	√
	ADD		D.ADD	Addition	√	√	√
	SUB		D.SUB	Subtraction	√	√	√
	INC		D.INC	Increase 1	√	√	√
	DEC		D.DEC	Decrease 1	√	√	√
	MUL		D.MUL	Multiplication	√	√	√
	DIV		D.DIV	Division	√	√	√
	ACCU		D.ACCU	Accumulation	√	√	√
	AVG		D.AVG	Average	√	√	√
	ABS		D.ABS	Absolute value	√	√	√
	NEG		D.NEG	Two's complement	√	√	√
Floating point instruction	FCMP			Floating point comparison	√	√	√
	FZCP			Floating point regional comparison	√	√	√
	FMOV			Floating point move instruction	√	√	√
	FADD			Floating point addition	√	√	√
	FSUB			Floating point subtraction	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	FMUL			Floating point multiplication	√	√	√
	FDIV			Floating point division	√	√	√
	FACCU			Floating point accumulation	√	√	√
	FAVG			Floating point average	√	√	√
	FMAX			Floating point maximum	√	√	√
	FMIN			Floating point minimum	√	√	√
	FTOI			Floating point convert to integer	√	√	√
	ITOF		D.ITOF	Integer convert to floating point	√	√	√
	FABS			Floating point absolute	√	√	√
	FSQR			Floating point square root	√	√	√
	FSIN			Sine	√	√	√
	FCOS			Cosine	√	√	√
	FTAN			Tangent	√	√	√
	FASIN			Arcsine	√	√	√
	FACOS			Arc cosine	√	√	√
	FATAN			Arctangent	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	FLN			Natural logarithm	√	√	√
	FLOG			The base-10 logarithm of a number	√	√	√
	FEXP			Nature exponential	√	√	√
	FRAD			Angle convert to radian	√	√	√
	FDEG			Radian convert to angle	√	√	√
	FXY			Exponent	√	√	√
Clock instruction	TCMP			Real time clock comparison	√	√	√
	TACCU			Time accumulative total	√	√	√
	SCLK			Setup system clock	√	√	√
	TIME			Time switch	√	√	√
	DATE			Date switch	√	√	√
	INVT			Count down	√	√	√
Communication instruction	SUM	SUM.LB		SUM verify	√	√	√
	BCC	BCC.LB		BCC verify	√	√	√
	CRC	CRC.LB		CRC verify	√	√	√
	LRC	LRC.LB		LRC verify	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	COMM	COMM.LB		Free communications	√	√	√
	MODR			Modbus read	√	√	√
	MODW			Modbus write	√	√	√
	HWRD			Speedbus read	√	√	√
	HWWR			Speedbus write	√	√	√
	RCV			Receive communication data	√	√	√
	XMT	XMT.LB		Sent communication data	√	√	√
	FROM			Extend module CR register read	√	√	√
	TO			Extend module CR register write	√	√	√
	TCPMDR			Modbus TCP read	√	√	√
	TCPMDW			Modbus TCP write	√	√	√
	TCPHWR			Speedbus TCP read	√	√	√
	TCPHWW			Speedbus TCP write	√	√	√
Interrupt instruction	ATCH			Interrupt binding	√	√	√
	DTCH			Interrupt release	√	√	√
	ENI			Enable interrupt	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	DISI			Disable interrupt	√	√	√
Program control instruction	MC			Master control	√	√	√
	MCR			Master control clear	√	√	√
	FOR			Loop command	√	√	√
	NEXT			Loop end	√	√	√
	WAIT			Delay wait	√	√	√
	CALL			Call subroutine	√	√	√
	EXIT			Condition exit	√	√	√
	REWD			Scanning time reset	√	√	√
	JMPC			Condition jump	√	√	√
	LBL			Jump label	√	√	√
Special function instruction	GPWM			General pulse width modulation	√	√	√
	FTC			Fuzzy temperature control	√	√	√
	PID			PID control	√	√	√
	HAL		D.HAL	Upper limit alarm	√	√	√
	LAL		D.LAL	Lower limit alarm	√	√	√

Instruction type	Instruction name	8 bit model	32 bit model	Instruction function	Support language		
					LD	FBD	IL
	LIM		D.LIM	Range limitation	√	√	√
	SC		D.SC	Linear conversion	√	√	√
	VC			Valve control	√	√	√
	TTC			Temperature curve control	√	√	√
	APID			Self-tuning PID	√	√	√

General declare of the instruction

1. En enable input :En is the enable input item of the instruction ,Only En have electricity (ON), the instruction executed, otherwise not executed.
2. Eno Enable output: Eno is the Enable output item of the instruction, indicate the instruction is executing. When En have electricity (ON) and instruction executed properly then Eno output have electricity (ON), when En have not electricity (OFF) or instruction executed error (e.g:parameter not property of the instruction) then Eno output have not electricity (OFF). The application instruction in LD. FBD language ,the great mass of the instruction have Eno Enable output item, All IL instructions have not Eno output item,it will be instead of the ENO instruction in IL language.
3. In LD language the AND. OR. XOR instructions, will be instead of logic link.
4. 32 bit instruction at 16 bit instruction name "D.", indicate use 2 continuous register.Such as ADD,16 bit addition is ADD,32 bit addition is D.ADD.
5. 8 bit instruction at 16 bit instruction behind the name plus ".LB", indicate only use the low byte of the register .Such as COMM,16 bit instruction is COMM,8 bit

instruction is COMM.LB.

6. When the parameter items of many instruction which autoOccupy several continuous register, pay special attention to them when programming , avoid reusing the register to program execution incorrect.

Note: except CV48~CV79 are 32 bit register (total 32 entries),PLC other registers (AI. AQ. V. SV. LV. TV. CV. P) all are 16 bit register, one 16 bit register have 2 byte compose, one 32 bit register have 2 continuous 16 bit registers compose.

Compare switch

Compare switch used in LD program language dedicated, divide into:16 bit compare instruction. 32 bit compare instruction. floating point compare instruction. low byte compare instruction. high byte compare instruction.

Compare mode have:equal to (=). unequal to (<>). greater than(>). greater than or equal to (≥). less than (<). less than and equal to(≤) six type.

Program example:Download,instruction list as follows:

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
=	LB.= HB.=	D.=	Equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
<>	LB.<> HB.<>	D.<>	Unequal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
>	LB.> HB.>	D.>	Greater than compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
>=	LB.>= HB.>=	D.>=	Great than or equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
<	LB.< HB.<	D.<	Less than compare switch ,have 16 bit/32 bit /low byte/high byte model	√		
<=	LB.<= HB.<=	D.<=	Less than or equal to compare switch ,have 16 bit/32 bit /low byte/high byte model	√		

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
F.=			Floating-point number equal to compare switch	√		
F.<>			Floating-point number unequal to compare switch	√		
F.>			Floating-point number greater than compare switch	√		
F.>=			Floating-point number greater than or equal to compare switch	√		
F.<			Floating-point number less than compare switch	√		
F.<=			Floating-point number less than or equal to compare switch	√		

Step instruction

Step instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
STL			Step start	√		
SFROM			Step combine	√		
STO			Step jump	√		

[step instruction declare]

1. Subroutine. interrupt routine not support step instruction,FBD,IL language not support step instruction.
2. In the step not support jump . loop instruction.
3. If a step is ON ,the program within the step be executed, then not executed.

4. Support step branch . step combine process.

5. Jump between the step, the last step state and OUT instruction outputs . timer coil T and current value TV . counter coil C and current value CV within the step will be clear .SET instruction will be not reset.

6. When the step roll-out ,Y output want keep ON be use SET instruction drive the output , want clear the output to OFF, use RST instruction.

7. Step number Sn cannot repeat , without step instruction in the program the S relay can be used as general internal relay.

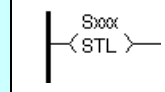
8. If want terminate the step , use RST Sx to reset the step, batch reset use the ZRST instruction.

9. Any S component can be used the start step , start step use STO or SET start , step jump use STO instruction.

10. The program can activate 10 different step process at the same time .

STL(Step start)

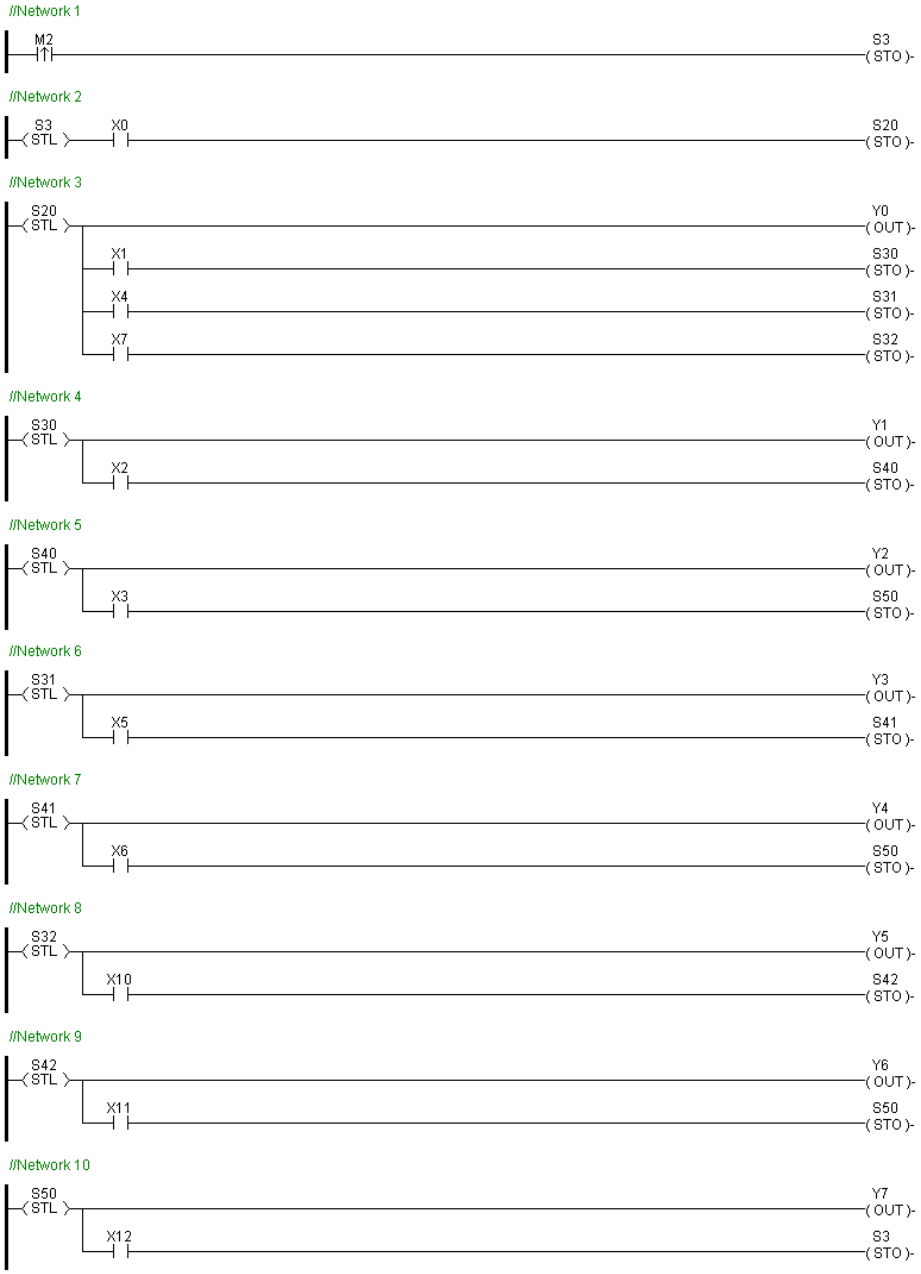
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format		Without	Without	Download

[Instruction function and effect declare]

STL instruction represent a step start , if the step have electricity ,the program within the step be executed, then not executed.

[Instruction example]



[Program description]:

1. When M2=ON then start step S3,by now S3=ON

2. When S3=ON, moreover jump condition X0=ON, then go to step S20,by now S20=ON. S3=OFF

3. When S20=ON then Y0=ON, when jump condition X1=ON , then go to step S30,by now S30=ON. S20=OFF. Y0=OFF

, when jump condition X4=ON , then go to step S31,by now S31=ON. S20=OFF. Y0=OFF

, when jump condition X7=ON , then go to step S32,by now S32=ON. S20=OFF. Y0=OFF

(step selectivity branch)

4. When S30=ON then Y1=ON, when jump condition X2=ON , then go to step S40,by now S40=ON. S30=OFF. Y1=OFF

5. When S40=ON then Y2=ON, when jump condition X3=ON , then go to step S50,by now S50=ON. S40=OFF. Y2=OFF

6. When S31=ON then Y3=ON, when jump condition X5=ON , then go to step S41,by now S41=ON. S31=OFF. Y3=OFF

7. When S41=ON then Y4=ON, when jump condition X6=ON , then go to step S50,by now S50=ON. S41=OFF. Y4=OFF

8. When S32=ON then Y5=ON, when jump condition X10=ON , then go to step S42,by now S42=ON. S32=OFF. Y5=OFF

9. When S42=ON then Y6=ON, when jump condition X11=ON , then go to step S50,by now S50=ON. S42=OFF. Y6=OFF

10. When S50=ON then Y7=ON, when jump condition X12=ON , then go to step S3,by now S3=ON. S50=OFF. Y7=OFF, circulate repeatedly.

SFROM (Step combine)

In

struction format and parameter specification

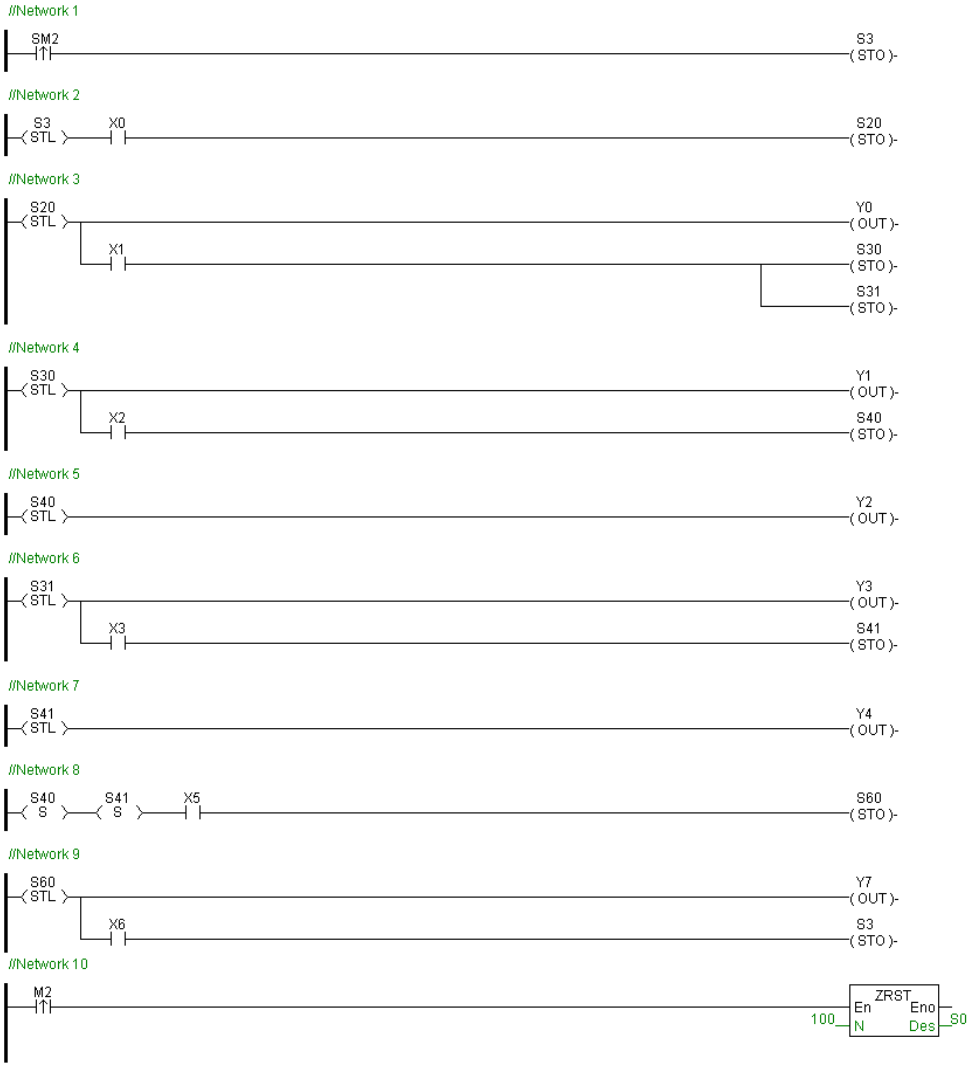
Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format	Sxxx S	Without	Without	Download

[Instruction function and effect declare]

SFROM use for combine after step parallel branch.

[Instruction example]



[Program description]:

1. When program start the first scanning cycle SM2=ON, start step S3,by now S3=ON

2. When S3=ON, moreover jump condition X0=ON , then go to step S20,by now S20=ON. S3=OFF

3. When S20=ON then Y0=ON, when jump condition X1=ON , then go to step S30 S31 (step parallel branch)at the same time,by now S30=ON. S31=ON. S20=OFF. Y0=OFF

4. When S30=ON then Y1=ON, when jump condition X2=ON , then go to step S40,by now S40=ON. S30=OFF. Y1=OFF

5. When S40=ON then Y2=ON

6. When S31=ON then Y3=ON, when jump condition X3=ON , then go to step S41,by now S41=ON. S31=OFF. Y3=OFF

7. When S41=ON then Y4=ON

8. When S40=ON moreover S41=ON (step parallel branch combine), moreover jump condition X5=ON时, then go to step S60,by now S60=ON. S40=OFF. Y2=OFF. S41=OFF. Y4=OFF

9. When S60=ON then Y7=ON, when jump condition X6=ON , then go to step S3,by now S3=ON. S60=OFF. Y7=OFF, circulate repeatedly.

10. If M2=ON ,then batch reset 100 steps start from s0 ,that is S0~S99.

STO(Step jump)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	Sxxx —(STO)-	Without	Without	Download

[Instruction function and effect declare]

STO use for start next step process, or bring the program process go to the specified step number.

[Instruction example]

[Program description]:

1. When M2=ON , start step S3,by now S3=ON

2. When S3=ON, moreover jump condition X0=ON , then go to step S20,by now S20=ON. S3=OFF

3. When S20=ON then Y0=ON,at the same time start timer T0, when T0 time is up , then go to step S30,by now S30=ON. S20=OFF. Y0=OFF. T0=OFF

4. When S30=ON then Y1=ON, when jump condition X1=ON , then go to step S60,by now S60=ON. S30=OFF. Y1=OFF

5. When S60=ON then Y7=ON, when jump condition X2=ON , then go to step S3,by now S3=ON. S60=OFF. Y7=OFF, circulate repeatedly.

, when exit condition X3=ON , then reset step S60,by now S60=OFF. Y7=OFF, terminate the step.

Bit instruction

Bit instruction list as follows

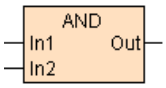
Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
AND			Logic AND		√	√
OR			Logic OR		√	√
XOR			Logic XOR		√	√
OUT			Coil output	√	√	√
SET			Setting	√	√	√
RST			Reset	√	√	√
ALT			ON/OFF alternately output	√	√	√
ZRST			Batch reset	√	√	√
ENO			Get ENO output			√

AND(Logic AND)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format	Without		AND In1, In2 [, … , In15], Out	Download

Parameter	Parameter define	Input	Output	Declare
In1	Operand 1	√		
In2	Operand 2	√		
…	…	…		
In15	Operand 15	√		
Out	Status output		√	

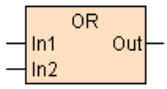
[Instruction function and effect declare]

AND instruction logic AND a group bit component of Input and then output ,only In1~In15 Input states all are ON the Out=ON ,otherwise Out=OFF,support 2~15 variable Input.

OR(Logic OR)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	Without		OR In1, In2 [, … , In15], Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
In1	Operand 1	√		
In2	Operand 2	√		
…	…	…		
In15	Operand 15	√		
Out	Status output		√	

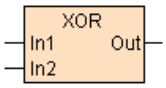
[Instruction function and effect declare]

OR instruction logic OR a group bit component of Input and then output ,if only one of In1~In15 Input states is ON the Out=ON ,otherwise Out=OFF,support 2~15 variable Input.

XOR(Logic XOR)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	Without		XOR In1, In2 [, … , In15], Out	Download

Parameter	Parameter define	Input	Output	Declare
In1	Operand 1	√		
In2	Operand 2	√		
…	…	…		

Parameter	Parameter define	Input	Output	Declare
In15	Operand 15	√		
Out	Status output		√	

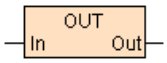
[Instruction function and effect declare]

XOR instruction logic XOR a group bit component of Input and then output , when the ON odd number of In1~In15 Input states the Out=ON ,otherwise Out=OFF,support 2~15 variable Input.

OUT(Coil output)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(OUT)-		OUT In, Out	Download

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Out	Output		√	

[Instruction function and effect declare]

OUT instruction assign Input state to Output,In=ON then Out=ON,In=OFF then Out=OFF.

SET(Setting)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(SET)-		SET In, Out	Download

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Out	Output		√	

[Instruction function and effect declare]

SET instruction according Input state to set output,In=ON then Out=ON,In=OFF then Out keep the original state.SET instruction general used edge Input executed.

RST(Reset)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(RST)-		RST In, Out	Download

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Out	Output		√	

[Instruction function and effect declare]

1. RST instruction according Input state to reset Output ,In=ON then Out=OFF,In=OFF then Out keep the original state.RST instruction general used edge Input executed.

2. If Out is timer Tx,at the same time reset the timer coil T and current value TV, if Output is counter Cx,at the same time reset the counter coil C and current value CV.

3. If Out is step relay S,except reset the step state, if the step is executing then reset the output of the OUT instruction . timer coil T and current value TV . counter coil C and current value CV within the step .

ALT(ON/OFF alternately output)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(ALT)-		ALT In, Out	Download

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Out	Output		√	

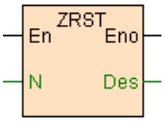
[Instruction function and effect declare]

ALT instruction negation the input state to output state, In=ON then Out negation it self,In=OFF then Out keep the original state.ALT instruction general used edge Input executed.

ZRST(Batch reset)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ZRST En, N, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable Input	√		
N	Component numbers to be reset	√		1~256
Eno	Enable output		√	
Des	Start address of component to be reset		√	

[Instruction function and effect declare]

1. ZRST instruction batch reset N component start from Des.ZRST instruction general used edge Input executed.

2. If Des is timer Tx, will reset timer coil T and current value TV,Iif Output is counter Cx, will reset counter coil C and current value CV.

3. if Des is step state S,except reset the step state, if the step is executing then reset the output of the OUT instruction . timer coil T and current value TV . counter coil C and current value CV within the step.

ENO(Get ENO output)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	Without	Without	EnO Out	Download

Parameter	Parameter define	Input	Output	Declare
Out	Output		√	

[Instruction function and effect declare]

1. IL Language All instruction without Eno Enable outpu item in IL Language ,for programmed by IL Language and FBD Language . LD Language the same function, in IL language special add ENO instruction, it's function equal to the Eno Enable output items of the application instruction in FBD. LD Language.

2. ENO instruction only one Output item,the state of Output items only relate to the first item near the ENO instruction in BD or LD language have Eno Output instruction.

[Instruction example]

```
    AND     X0,?X2,  M100
    MAX     M100,  V1000,  3,  V10 //V10 is V1000, V1001, V1002 maximum of the three register
    OUT     X2,  Y0
    ENO     M0 //Get MAX instruction Eno output，If MAX instruction execution is correct，then M0=ON
5   ADD     M0,  V10,  200,  AQ0
```

[Program description]

1. MAX instruction is a FBD or LD language instructionwhich have Eno Output item, moreover nearest ENO instruction(because OUT instruction have not Eno Output item), so in program, the output item state M0 of ENO instruction relate to MAX instruction executive condition.

2. When X0=ON (X2 normal close),M100=ON

3. M100=ON, MAX instruction execute ,V10 equal to V1000. V1001. V1002 the 3 registers maximum,Eno Output=ON of the MAX instruction

4. X2=OFF,Y0=OFF

5. ENO instruction get Eno Output of previous instruction,because MAX instruction Eno Output=ON,so the M0 state is ON

6. M0=ON,ADD instruction执行,AQ0=V10+200

Timer

Timer list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
TON			Delay ON	√	√	√
TOF			Delay OFF	√	√	√
TP			Pulse timer	√	√	√

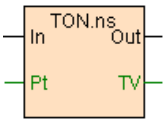
Note:T252~T255 time base fixed to 1ms.Others timer time base can be set arbitrarily to 10ms. 100ms. 1s.

TON(Delay ON)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			TON.ns In, Pt, Tx	Download

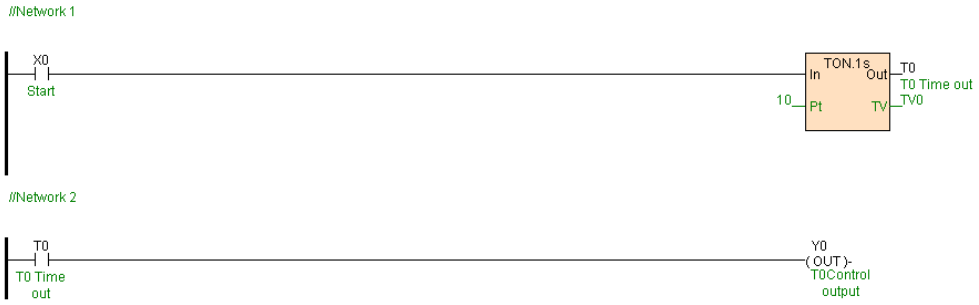
Parameter	Parameter define	Input	Output	Declare
ns	Base time value			T252~T255 time base are 1ms.Others can be set arbitrarily to 10ms. 100ms. 1s
In	Input	√		
Pt	Set time	√		
Out	Timer coil Tx		√	
TV	Current time		√	

[Instruction function and effect declare]

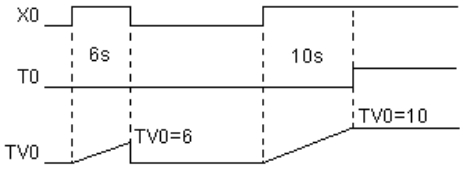
1. TON is delay ON instruction , when In=ON, start timer timing,TV is the current value of the timer, when TV equal to the set time (time time to),Out(timer output coil Tx)=ON,and stop timing.When In turn into OFF,Out(timer output coil Tx)=OFF, moreover TV value reset to zero.In the timing process(time non-arrival),In turn into OFF, then stop timing,Out(timer output coil Tx)=OFF, moreover TV value reset to zero.

2. Timing time= time base(ns)× setting time(Pt).E.g,:time base is 1s,setting time Pt=10, then delay on time is 1s × 10 = 10s.

[Instruction example]



[Program sketch map]



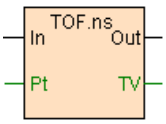
[Program description]

Timing time= time base(1s)× setting time (Pt=10)=10s.When X0=ON, timer T0 start timing, when TV0=10,T0=ON(Y0=ON) moreover stop timing .If X0=OFF,then T0=OFF(Y0=OFF),TV=0.

TOF(Delay OFF)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TOF.ns In, Pt, Tx	Download

Parameter	Parameter define	Input	Output	Declare
ns	Time base			T252~T255 time base is 1ms.Others can be set arbitrarily to 10ms. 100ms. 1s
In	Input	√		

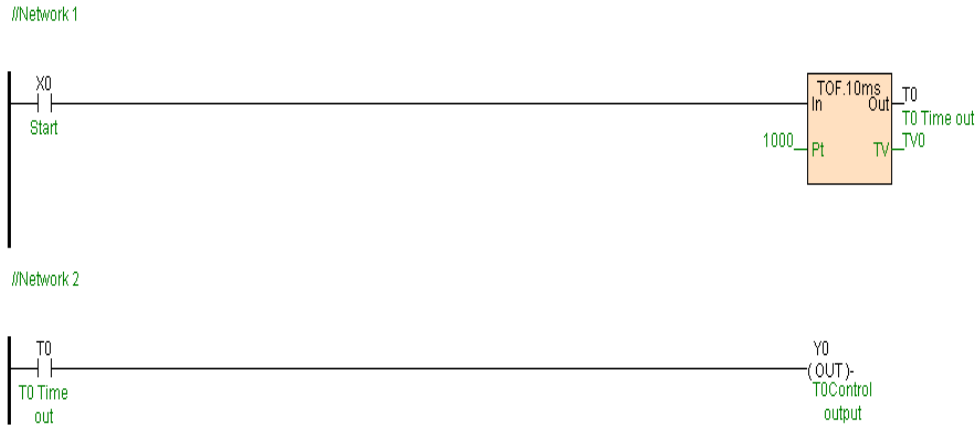
Parameter	Parameter define	Input	Output	Declare
Pt	Setting time	√		
Out	Timer coil Tx		√	
TV	Current time		√	

[Instruction function and effect declare]

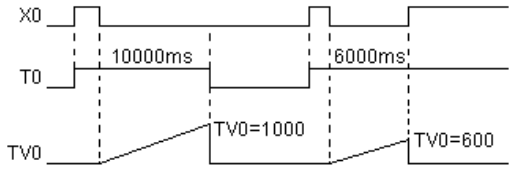
1. TOF is delay OFF instruction, when In=ON,Out(Timer coil Tx)=ON, When In state from ON go to OFF, start timer timing,TV is the timer current value, when TV equal to setting time (time time to),Out(Timer coil Tx)=OFF, and stop timing,TV=0.In the timing process(time non-arrival),the state of In go to ON, then Out (Timer coil Tx)=ON, timer stop timing,TV=0.

2. Timing time=time base(ns)× setting time(Pt).e.g.:time base is 10ms,Setting time Pt=1000, then delay OFF time is 10ms × 1000 = 10s.

[Instruction example]



[Program sketch map]



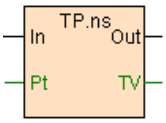
[Program description]

Timing time=time base (10ms)× setting time(Pt=1000)=10s.When X0=ON,T0=ON(Y0=ON), when X0=OFF, Timer T0 start timing, when TV0=1000,T0=OFF(Y0=OFF) moreover stop timing TV=0.if time non-arrival X0=ON, then T0=ON(Y0=ON), stop timing TV=0.

TP(Pulse timer)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TP.ns In, Pt, Tx	Download

Parameter	Parameter define	Input	Output	Declare
ns	Time base			T252~T255 time base is 1ms.Others can be set arbitrarily to 10ms. 100ms. 1s
In	Input	√		
Pt	Setting time	√		
Out	Timer coil Tx		√	
TV	Current time		√	

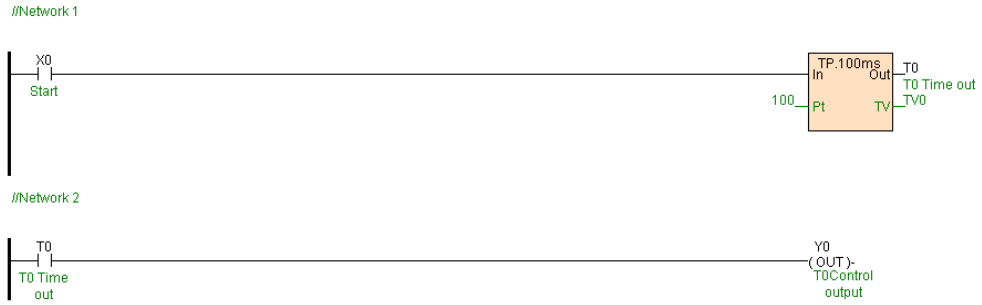
[Instruction function and effect declare]

1. TP is pulse timer, when In=ON,Out(Timer coil Tx)=ON, start timer timing,TV is timer current value, when TV equal to setting time (time arrival),Out(Timer coil

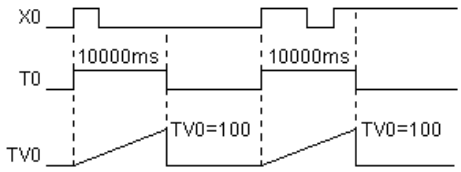
Tx)=OFF,stop timing moreover TV value reset to zero.In the timing process(time non-arrival),regardless of the state of In changed, timer keep timing,Out(Timer coil Tx) keep ON.

2. Timing time= time base(ns)× setting time(Pt).e.g.:time base is 100ms,setting timePt=100 , then the delay ON time is 100ms × 100 = 10s.

[Instruction example]



[Program sketch map]



[Program description]

Timing time= time base(100ms)× setting time(Pt=100)=10s.When X0=ON,T0=ON(Y0=ON),timer T0 start timing, when TV0=100,T0=OFF(Y0=OFF) moreover stop timing TV0=0.In the timing process,regardless of the state of X0 changed,timer keep timing,T0 keep ON until timing terminate.

Counter

Counter list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL

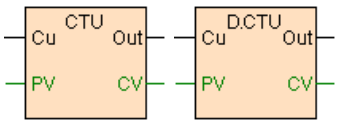
Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
CTU		D.CTU	Increase counter	√	√	√
CTD		D.CTD	Decrease counter	√	√	√
CTUD		D.CTUD	Increase and Decrease counter	√	√	√

Note:C48~C79 are 32 bit counter, others are 16 bit counter.

CTU. D.CTU(Increase counter)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			CTU Cu, PV, Cx D.CTU Cu, PV, Cx	Download

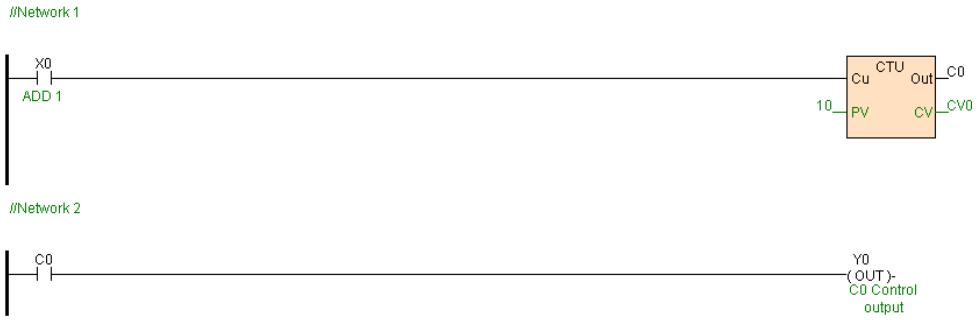
Parameter	Parameter define	Input	Output	Declare
Cu	Increase count input	√		
PV	Counter preset	√		
Out	Counter coil Cx		√	Among C48~C79 are 32 bit counter ,total 32 point
CV	Counter current value		√	

[Instruction function and effect declare]

CTU is 16 bit increase counter instruction (D.CTU is 32 bit), when increase count input Cu from OFF go to ON, counter add 1, when CV great than or equal to PV,

Out (counter coil Cx)=ON. Counting reached,if the counting pulse input again, counting will be continue ,CV value will be added 1 continue , until reach maximum value(16 bit counter maximum value is 32767,32 bit counter maximum value is 2147483647),counting will not be continue after reach maximum value.

[Instruction example]



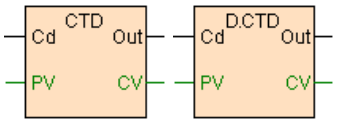
[Program description]

When X0 from OFF go to ON once,CV0 add 1, when CV0 ≧ 10 counting reach,C0=ON(Y0=ON).

CTD. D.CTD(Decrease counter)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			CTD Cd, PV, Cx D.CTD Cd, PV, Cx	Download

Parameter	Parameter define	Input	Output	Declare
Cd	Decrease count input	√		
PV	Counter preset	√		

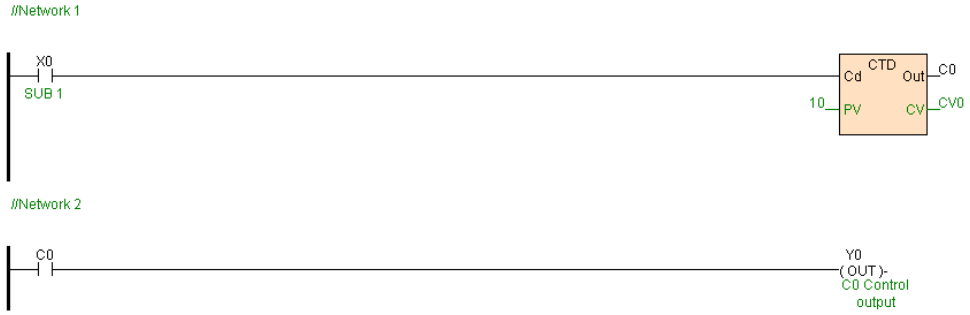
Parameter	Parameter define	Input	Output	Declare
Out	Counter coil Cx		√	Among C48~C79 are 32 bit counter ,total 32 point
CV	Counter current value		√	

[Instruction function and effect declare]

1. CTD is 16 bit increase counter instruction (D.CTU is 32 bit), When decrease count input Cu from OFF go to ON, counter subtract 1, when CV=0 then counting reached, Out (counter coil Cx)=ON.Counting reached ,if the counting pulse input again, counting will not be continue.

2. When CTD instruction loaded or reset ,CV = PV, that is CTD decrease from preset to 0.

[Instruction example]



[Program description]

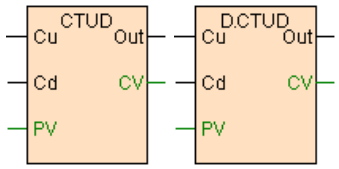
When program running CV=10, when X0 from OFF go to ON once,CV0 decrease 1, when CV0 = 0 counting reach,C0=ON(Y0=ON).

CTUD. D.CTUD(Increase and Decrease counter)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			CTUD Cu, Cd, PV, Cx D.CTUD Cu, Cd, PV, Cx	Download

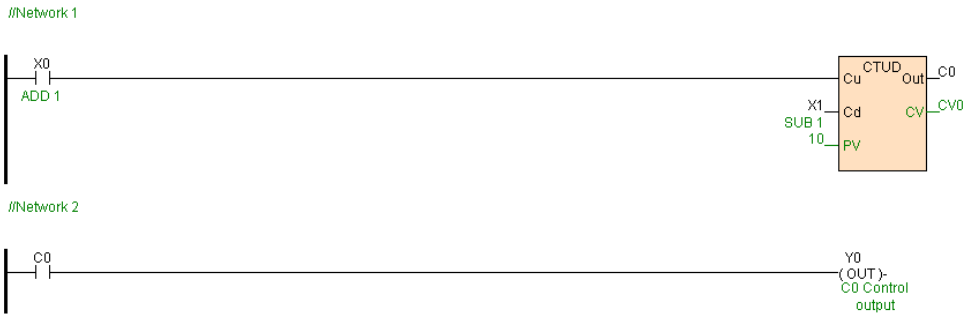
Parameter	Parameter define	Input	Output	Declare
Cu	Increase count Input	√		
Cd	Decrease input Input	√		
PV	Counter preset	√		
Out	Counter coil Cx		√	Among C48~C79 are 32 bit counter ,total 32 point
CV	Counter current value		√	

[Instruction function and effect declare]

1. CTUD is 16 bit increase and decrease counter instruction (D.CTUD is 32 bit), when increase count input Cu from OFF go to ON,counter add 1, when decrease count Input Cd from OFF go to ON,counter subtract 1, when CV great than or equal to PV, Out (counter coil Cx)=ON, When CV less than PV, Out (counter coil Cx)=OFF.

2. 16 bit counter maximum CV value is 32767,minimum CV value is -32768,32 bit counter maximum CV value is 2147483647,minimum CV value is -2147483648.

[Instruction example]



[Program description]

When X0 from OFF go to ON once,CV0 add 1, when X1 from OFF go to ON once,CV0 subtract 1, when CV0 ≧ 10,C0=ON(Y0=ON), When CV0 ＜ 10,C0=OFF(Y0=OFF).

High speed control instruction

High speed control instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
SHC			Single high speed counter	√	√	√
RESH			IO refresh	√	√	√
HHSC			High speed counter	√	√	√
HCWR			Write high speed counter	√	√	√
SPD			Speed detection	√	√	√
PWM			Pulse width modulation	√	√	√
PLSY		D.PLSY	Pulse output	√	√	√
PLSR		D.PLSR	Accelerate and decelerate pulse output	√	√	√
ZRN			Origin point return	√	√	√
SETZ			Set electric origin point	√	√	√
PPMR			Linear interpolation	√	√	√
CIMR			Circular interpolation	√	√	√
SPLS			Single pulse output	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
SYNP			Synchronization pulse output	√	√	√
PSTOP			Stop pulse output	√	√	√

[High speed control instruction declare]

1. PLC high speed counter and high speed pulse output defined by channel ,related the hardware of the PLC ,Configured by "PLC hardware configure". High speed counter channel signify by HSCx ,each channel use 2 high speed pulse input point .High speed pulse output channel signify by PLSx , each channel use 2 high speed pulse output point .
2. High speed counter support :pulse/direction . positive/negative pulse . A/B phase pulse input model, support 1. 2. 4 frequency multiplication counting model , refer to" HSC high speed counter parameter".
3. High speed pulse output support : single pulse. pulse/direction . positive/negative pulse . A/B phase pulse . synchronization pulse output,refer to"PLS high speed pulse output parameter".
4. Motion control support linear interpolation . circular interpolation . synchronization pulse output etc.; support absolute address . relative address ; support backlash compensation ;support electric origin point redefine etc..
5. SM system state bit of high speed counter as follows:

SM	Function declare	R/W	Preserve	Default
SM25	HSC0 study enable control,0 is normal state,1 is study state	R/W	No	0

SM26	HSC0 study confirm control	R/W	No	0
SM27	HSC0 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM30	HSC0 direction indication,0 is increase,1 is decrease	R	No	0
SM31	HSC0 error indication	R	No	0
SM33	HSC1 study enable control,0 is normal state,1 is study state	R/W	No	0
SM34	HSC1 study confirm control	R/W	No	0
SM35	HSC1 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM38	HSC1 direction indication,0 is increase,1 is decrease	R	No	0
SM39	HSC1 error indication	R	No	0
SM41	HSC2 study enable control,0 is normal state,1 is study state	R/W	No	0
SM42	HSC2 study confirm control	R/W	No	0
SM43	HSC2 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM46	HSC2 direction indication,0 is increase,1 is decrease	R	No	0

SM47	HSC2 error indication	R	No	0
SM49	HSC3 study enable control,0 is normal state,1 is study state	R/W	No	0
SM50	HSC3 study confirm control	R/W	No	0
SM51	HSC3 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM54	HSC3 direction indication,0 is increase,1 is decrease	R	No	0
SM55	HSC3 error indication	R	No	0
SM57	HSC4 study enable control,0 is normal state,1 is study state	R/W	No	0
SM58	HSC4 study confirm control	R/W	No	0
SM59	HSC4 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM62	HSC4 direction indication,0 is increase,1 is decrease	R	No	0
SM63	HSC4 error indication	R	No	0
SM65	HSC5 study enable control,0 is normal state,1 is study state	R/W	No	0
SM66	HSC5 study confirm control	R/W	No	0

SM67	HSC5 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM70	HSC5 direction indication,0 is increase,1 is decrease	R	No	0
SM71	HSC5 error indication	R	No	0
SM73	HSC6 study enable control,0 is normal state,1 is study state	R/W	No	0
SM74	HSC6 study confirm control	R/W	No	0
SM75	HSC6 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM78	HSC6 direction indication,0 is increase,1 is decrease	R	No	0
SM79	HSC6 error indication	R	No	0
SM81	HSC7 study enable control,0 is normal state,1 is study state	R/W	No	0
SM82	HSC7 study confirm control	R/W	No	0
SM83	HSC7 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM86	HSC7 direction indication,0 is increase,1 is decrease	R	No	0
SM87	HSC7 error indication	R	No	0

6. SV system register of high speed counter as follows :

SV	Function declare	R/W	Preserve	Default
SV60	HSC0 current segment number	R	Yes	0
SV61	HSC0 low word of current value	R	Yes	0
SV62	HSC0 high word of current value	R	Yes	0
SV63	HSC0 error code	R	Yes	0
SV801	HSC0 low word of frequency	R	Yes	0
SV802	HSC0 high word of frequency	R	Yes	0
SV64	HSC1 current segment number	R	Yes	0
SV65	HSC1 low word of current value	R	Yes	0
SV66	HSC1 high word of current value	R	Yes	0
SV67	HSC1 error code	R	Yes	0
SV803	HSC1 low word of frequency	R	Yes	0
SV804	HSC1 high word of frequency	R	Yes	0
SV68	HSC2 current segment number	R	Yes	0
SV69	HSC2 low word of current value	R	Yes	0

SV70	HSC2 high word of current value	R	Yes	0
SV71	HSC2 error code	R	Yes	0
SV805	HSC2 low word of frequency	R	Yes	0
SV806	HSC2 high word of frequency	R	Yes	0
SV72	HSC3 current segment number	R	Yes	0
SV73	HSC3 low word of current value	R	Yes	0
SV74	HSC3 high word of current value	R	Yes	0
SV75	HSC3 error code	R	Yes	0
SV807	HSC3 low word of frequency	R	Yes	0
SV808	HSC3 high word of frequency	R	Yes	0
SV76	HSC4 current segment number	R	Yes	0
SV77	HSC4 low word of current value	R	Yes	0
SV78	HSC4 high word of current value	R	Yes	0
SV79	HSC4 error code	R	Yes	0
SV809	HSC4 low word of frequency	R	Yes	0

SV810	HSC4 high word of frequency	R	Yes	0
SV80	HSC5 current segment number	R	Yes	0
SV81	HSC5 low word of current value	R	Yes	0
SV82	HSC5 high word of current value	R	Yes	0
SV83	HSC5 error code	R	Yes	0
SV811	HSC5 low word of frequency	R	Yes	0
SV812	HSC5 high word of frequency	R	Yes	0
SV84	HSC6 current segment number	R	Yes	0
SV85	HSC6 low word of current value	R	Yes	0
SV86	HSC6 high word of current value	R	Yes	0
SV87	HSC6 error code	R	Yes	0
SV813	HSC6 low word of frequency	R	Yes	0
SV814	HSC6 high word of frequency	R	Yes	0
SV88	HSC7 current segment number	R	Yes	0
SV89	HSC7 low word of current value	R	Yes	0
SV90	HSC7 high word of current value	R	Yes	0

SV91	HSC7 error code	R	Yes	0
SV815	HSC7 low word of frequency	R	Yes	0
SV816	HSC7 high word of frequency	R	Yes	0

7. SM system state bit of High speed pulse output as follows:

SM	Function declare	R/W	Preserve	Default
SM93	PLS0 prohibit the forward pulse	R/W	yes	0
SM94	PLS0 prohibit the reverse pulse	R/W	yes	0
SM95	PLS0 prohibit the brake function	R/W	yes	0
SM96	PLS0 pulse output indication	R	yes	0
SM97	PLS0 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM98	PLS0 error flag	R	yes	0
SM99	PLS0 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM100	PLS0 pulse output complete	R	yes	0
SM109	PLS1 prohibit the forward pulse	R/W	yes	0
SM110	PLS1 prohibit the reverse pulse	R/W	yes	0

SM111	PLS1 prohibit the brake function	R/W	yes	0
SM112	PLS1 pulse output indication	R	yes	0
SM113	PLS1 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM114	PLS1 error flag	R	yes	0
SM115	PLS1 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM116	PLS1 pulse output complete	R	yes	0
SM125	PLS2 prohibit the forward pulse	R/W	yes	0
SM126	PLS2 prohibit the reverse pulse	R/W	yes	0
SM127	PLS2 prohibit the brake function	R/W	yes	0
SM128	PLS2 pulse output indication	R	yes	0
SM129	PLS2 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM130	PLS2 error flag	R	yes	0
SM131	PLS2 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM132	PLS2 pulse output complete	R	yes	0

SM141	PLS3 prohibit the forward pulse	R/W	yes	0
SM142	PLS3 prohibit the reverse pulse	R/W	yes	0
SM143	PLS3 prohibit the brake function	R/W	yes	0
SM144	PLS3 pulse output indication	R	yes	0
SM145	PLS3 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM146	PLS3 error flag	R	yes	0
SM147	PLS3 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM148	PLS3 pulse output complete	R	yes	0
SM157	PLS4 prohibit the forward pulse	R/W	yes	0
SM158	PLS4 prohibit the reverse pulse	R/W	yes	0
SM159	PLS4 prohibit the brake function	R/W	yes	0
SM160	PLS4 pulse output indication	R	yes	0
SM161	PLS4 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM162	PLS4 error flag	R	yes	0
SM163	PLS4 position model ,0 is relative address	R/W	yes	0

	,1 is absolute address			
SM164	PLS4 pulse output complete	R	yes	0
SM173	PLS5 prohibit the forward pulse	R/W	yes	0
SM174	PLS5 prohibit the reverse pulse	R/W	yes	0
SM175	PLS5 prohibit the brake function	R/W	yes	0
SM176	PLS5 pulse output indication	R	yes	0
SM177	PLS5 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM178	PLS5 error flag	R	yes	0
SM179	PLS5 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM180	PLS5 pulse output complete	R	yes	0
SM189	PLS6 prohibit the forward pulse	R/W	yes	0
SM190	PLS6 prohibit the reverse pulse	R/W	yes	0
SM191	PLS6 prohibit the brake function	R/W	yes	0
SM192	PLS6 pulse output indication	R	yes	0
SM193	PLS6 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0

SM194	PLS6 error flag	R	yes	0
SM195	PLS6 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM196	PLS6 pulse output complete	R	yes	0
SM205	PLS7 prohibit the forward pulse	R/W	yes	0
SM206	PLS7 prohibit the reverse pulse	R/W	yes	0
SM207	PLS7 prohibit the brake function	R/W	yes	0
SM208	PLS7 pulse output indication	R	yes	0
SM209	PLS7 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM210	PLS7 error flag	R	yes	0
SM211	PLS7 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM212	PLS7 pulse output complete	R	yes	0

8. SV system register of high speed pulse output as follows:

SV	Function declare	R/W	Preserve	Default
SV92	PLS0 current segment number	R	Yes	0

SV93	PLS0 low word of pulse output number	R	Yes	0
SV94	PLS0 high word of pulse output number	R	Yes	0
SV95	PLS0 low word of current position	R/W	Yes	0
SV96	PLS0 high word of current position	R/W	Yes	0
SV97	PLS0 error code	R	Yes	0
SV156	PLS0 low word of mechanical original point	R/W	Yes	0
SV157	PLS0 high word of mechanical original point	R/W	Yes	0
SV158	PLS0 number of pulses to compensate the reverse interval	R/W	Yes	0
SV159	PLS0 follow performance parameters, range: 1~100	R/W	Yes	50
SV98	PLS1 current segment number	R	Yes	0
SV99	PLS1 low word of pulse output number	R	Yes	0
SV100	PLS1 high word of pulse output number	R	Yes	0
SV101	PLS1 low word of current position	R/W	Yes	0
SV102	PLS1 high word of current position	R/W	Yes	0
SV103	PLS1 error code	R	Yes	0

SV160	PLS1 low word of mechanical original point	R/W	Yes	0
SV161	PLS1 high word of mechanical original point	R/W	Yes	0
SV162	PLS1 number of pulses to compensate the reverse interval	R/W	Yes	0
SV163	PLS1 follow performance parameters, range: 1~100	R/W	Yes	50
SV104	PLS2 current segment number	R	Yes	0
SV105	PLS2 low word of pulse output number	R	Yes	0
SV106	PLS2 high word of pulse output number	R	Yes	0
SV107	PLS2 low word of current position	R/W	Yes	0
SV108	PLS2 high word of current position	R/W	Yes	0
SV109	PLS2 error code	R	Yes	0
SV164	PLS2 low word of mechanical original point	R/W	Yes	0
SV165	PLS2 high word of mechanical original point	R/W	Yes	0
SV166	PLS2 number of pulses to compensate the reverse interval	R/W	Yes	0
SV167	PLS2 follow performance parameters,	R/W	Yes	50

	range: 1~100			
SV110	PLS3 current segment number	R	Yes	0
SV111	PLS3 low word of pulse output number	R	Yes	0
SV112	PLS3 high word of pulse output number	R	Yes	0
SV113	PLS3 low word of current position	R/W	Yes	0
SV114	PLS3 high word of current position	R/W	Yes	0
SV115	PLS3 error code	R	Yes	0
SV168	PLS3 low word of mechanical original point	R/W	Yes	0
SV169	PLS3 high word of mechanical original point	R/W	Yes	0
SV170	PLS3 number of pulses to compensate the reverse interval	R/W	Yes	0
SV171	PLS3 follow performance parameters, range: 1~100	R/W	Yes	50
SV116	PLS4 current segment number	R	Yes	0
SV117	PLS4 low word of pulse output number	R	Yes	0
SV118	PLS4 high word of pulse output number	R	Yes	0
SV119	PLS4 low word of current position	R/W	Yes	0

SV120	PLS4 high word of current position	R/W	Yes	0
SV121	PLS4 error code	R	Yes	0
SV172	PLS4 low word of mechanical original point	R/W	Yes	0
SV173	PLS4 high word of mechanical original point	R/W	Yes	0
SV174	PLS4 number of pulses to compensate the reverse interval	R/W	Yes	0
SV175	PLS4 follow performance parameters, range: 1~100	R/W	Yes	50
SV122	PLS5 current segment number	R	Yes	0
SV123	PLS5 low word of pulse output number	R	Yes	0
SV124	PLS5 high word of pulse output number	R	Yes	0
SV125	PLS5 low word of current position	R/W	Yes	0
SV126	PLS5 high word of current position	R/W	Yes	0
SV127	PLS5 error code	R	Yes	0
SV176	PLS5 low word of mechanical original point	R/W	Yes	0
SV177	PLS5 high word of mechanical original point	R/W	Yes	0

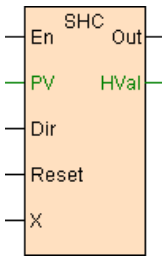
SV178	PLS5 number of pulses to compensate the reverse interval	R/W	Yes	0
SV179	PLS5 follow performance parameters, range: 1~100	R/W	Yes	50
SV128	PLS6 current segment number	R	Yes	0
SV129	PLS6 low word of pulse output number	R	Yes	0
SV130	PLS6 high word of pulse output number	R	Yes	0
SV131	PLS6 low word of current position	R/W	Yes	0
SV132	PLS6 high word of current position	R/W	Yes	0
SV133	PLS6 error code	R	Yes	0
SV180	PLS6 low word of mechanical original point	R/W	Yes	0
SV181	PLS6 high word of mechanical original point	R/W	Yes	0
SV182	PLS6 number of pulses to compensate the reverse interval	R/W	Yes	0
SV183	PLS6 follow performance parameters, range: 1~100	R/W	Yes	50
SV134	PLS7 current segment number	R	Yes	0
SV135	PLS7 low word of pulse output number	R	Yes	0

SV136	PLS7 high word of pulse output number	R	Yes	0
SV137	PLS7 low word of current position	R/W	Yes	0
SV138	PLS7 high word of current position	R/W	Yes	0
SV139	PLS7 error code	R	Yes	0
SV184	PLS7 low word of mechanical original point	R/W	Yes	0
SV185	PLS7 high word of mechanical original point	R/W	Yes	0
SV186	PLS7 number of pulses to compensate the reverse interval	R/W	Yes	0
SV187	PLS7 follow performance parameters, range: 1~100	R/W	Yes	50

SHC(Single high counter)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SHC En, PV, Dir, Reset, X, Out, HVal	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PV	Preset value	√		Occupy 2 continuous component
Dir	Counter direction	√		
Reset	Reset	√		
X	Pulse input	√		
Out	Comparison results		√	
HVal	High speed counter current value		√	Occupy 2 continuous component

[Instruction function and effect declare]

1. SHC instruction counts the high-speed impulse input of the Xn input point , it does not use the HSCx high-speed pulse input channel, one high-speed pulse input channel has two high-speed pulse input points. Therefore the host with eight pulse input channels can achieve the fucntion of 16 ways high-speed pulse input.

2. SHC instruction is a 32-bit high-speed pulse counter, it does not generate high-speed counter interrupt, and it doesn't use the SM, SV.

3. Dir terminal control the counting direction, when Dir = OFF it adds counting;while when Dir = ON,it reduces counting.

4. Reset terminal controls the reset of the counter,when Reset = ON , it resets the counter.

5. When "HVal" is greater than or equal to "PV" , "Out" is equal to "ON", when "HVal" is less than "PV", then "Out" is equal to "OFF".

[Instruction example]

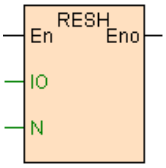
[Program description]

1. When M0=ON , HSC high-speed counter works, it begings counting the high-speed pulse input of the X0 input.
2. When M8=OFF, it adds counting, when M8=ON,it reduces counting.
3. When V50>=V1000, M100=ON, while when V50<V1000, M100=OFF .
4. When M9=ON, it resets the counter, M100=OFF, V50=0.

RESH(IO refresh)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			RESH En, IO, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
IO	IO start address be refreshed	√		Occupy N continuous component

Parameter	Parameter define	Input	Output	Declare
N	Number of component be refreshed	√		1~256
Eno	Enable output		√	

[Instruction function and effect declare]

1. RESH instruction use for refresh the state of external digital Input . Output (X. Y), in order to improve the response speed of the external signal.

2. En is the enable item of the instruction , when the state of En ON, the N continuous component (X. Y) of IO assigned will be updated immediately , without having to wait until the program scan complete

,the IO state update independence to the program scan cycle.

Note: the instruction used for real-time or high accuracy control circumstance ,as high speed control . interrupt processing etc. .Without RESH instruction in the program ,PLC external Input . Output state will be updated after the total program scaned finish.

[Instruction example]

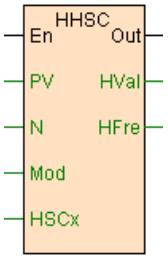
[Program description]

When X0=ON,Y0~Y7 Output state updated immediately , without having to wait until the program scan complete.

HHSC(High speed counter)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			HHSC En, PV, N, Mod, HSCx, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PV	Start address of preset value	√		Each segmentOccupy 2 register
N	Number of compare segments	√		1-48
Mod	Comp are model	√		0-2:0 is single segment compare , 1 is absoulte compare , 2 is relatively compare
HSCx	High speed counter number	√		
Out	Start address of compare result		√	Each segmentOccupy 1 component
HVal	High speed counter current value		√	Occupy 2 System register
HFre	High speed counter current frequency value		√	Occupy 2 System register

[Instruction function and effect declare]

1. HHSC instruction is deal with the high speed pulse input , it can deal with input pulse counting and measure the pulse frequency at the same time.

2. High speed counter support :pulse/direction . positive/negative pulse . A/B phase pulse input model, support 1. 2. 4 frequency multiplication counting model , refer to"HSC high speed counter parameter"

3. HHSC instruction relate to SM system state bit . SV System register , when count value=preset value generate " HSCx current value=preset value (the preset value of each segment will be generated) " interrupt , when the direction of input pulse changed will be generate "HSCx input direction changed " interrupt .

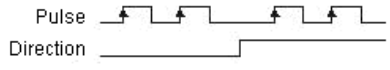
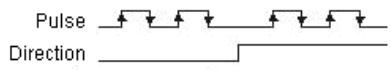
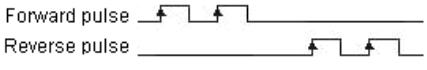
4. Support multi-segment compare ,support 3 type compare model :single segment compare . absolute model compare . relatively model compare .

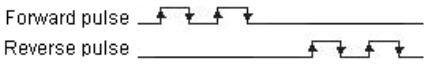
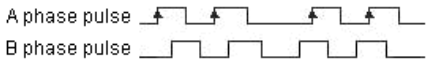
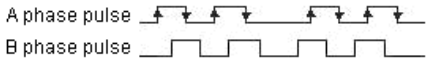
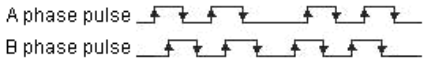
5. HHSC instruction have self-learning function ,in the self-learning can record the current value to the preset value , can continuous multi-segment self-learning .The high speed counter HSCx will be reset while enter into or quit the self-learning state .

6. Reset high speed counter . modify the preset value real time . modify the current value . modify the current segment number ,use HCWR instruction.

7. When En=ON,HHSC instruction executing, when En=OFF ,stop counting.

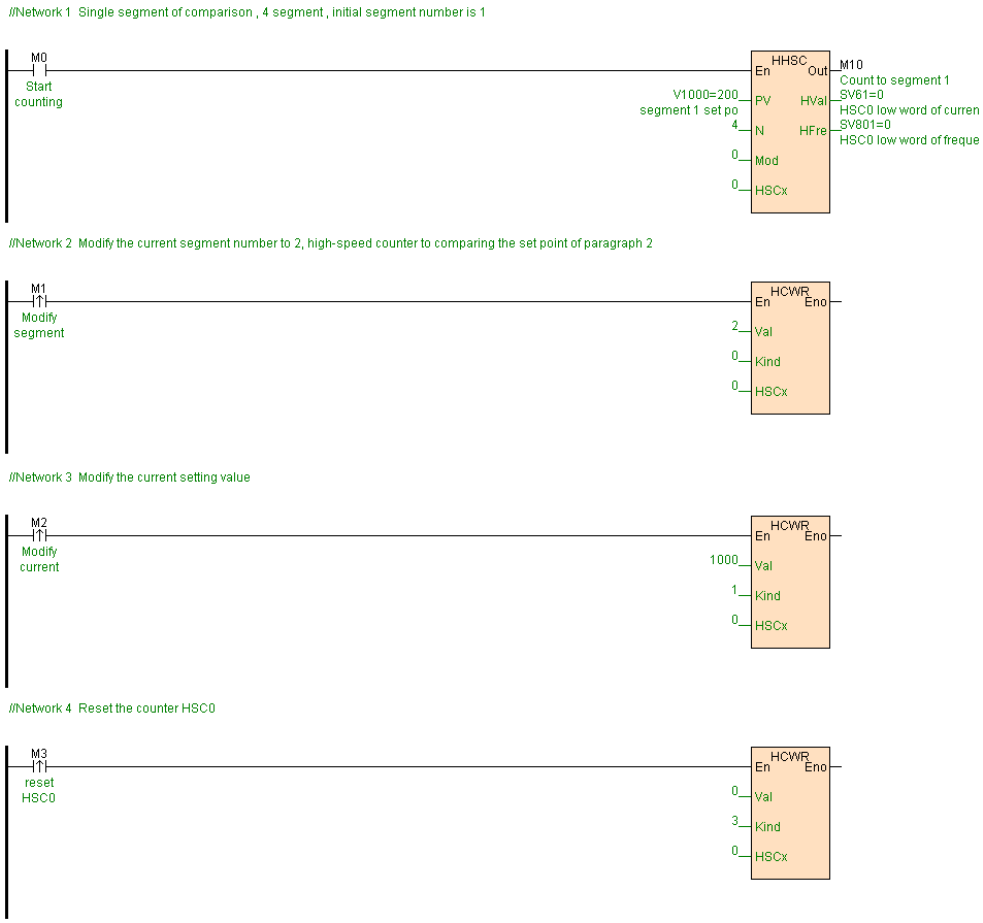
[High speed counter model and pulse waveform]

Count model		Pulse waveform	
Model	**Times frequency**	**Increase count**	**Decrease count**
0 --pulse/direction	1		
1 --pulse/direction	2		
2 -- foreward/reversal	1		

Count model		Pulse waveform	
Model	**Times frequency**	**Increase count**	**Decrease count**
3 -- foreward/reversal	2		
4 -- A/B phase	1		
5 -- A/B phase	2		
6 -- A/B phase	4		

[Instruction example 1]

Download



[Program 1 declare]

PV component	Value	Declare
V1000V1001	200	First segment preset
V1002V1003	500	Second segment preset
V1004V1005	1200	Third segment preset
V1006V1007	1500	Fourth segment preset

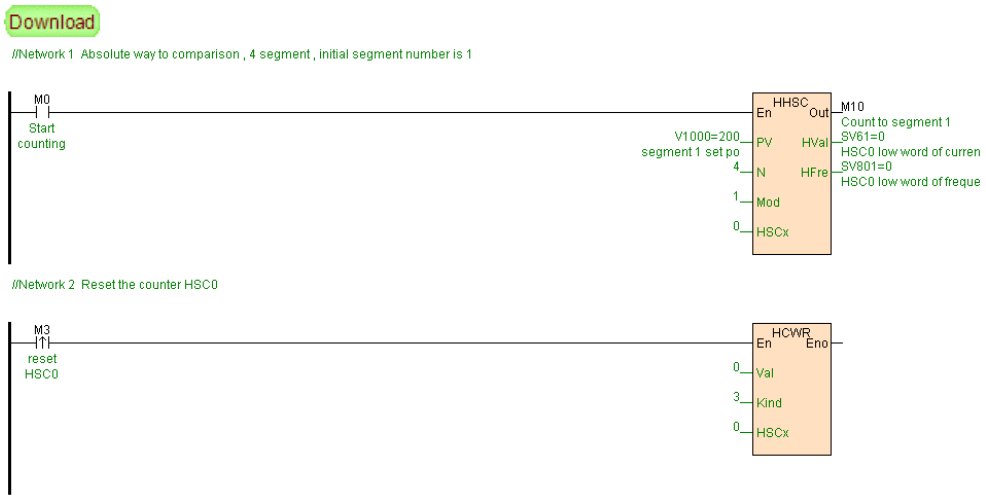
1. When M0=ON ,HHSC instruction executing ,high speed counter HSC0 set to single segment compare model ,number of compare segments are 4.Initial segment number is 1,HSC0 first segment preset value=200(V1000V1001) for compare.

2. When M1=ON ,instead HSC0 current segment number to 2,then HSC0 second segment preset value =500(V1002V1003) for compare .

3. When M2=ON ,instead HSC0 preset of current segment to 1000,that is second segment preset value=1000(V1002V1003).

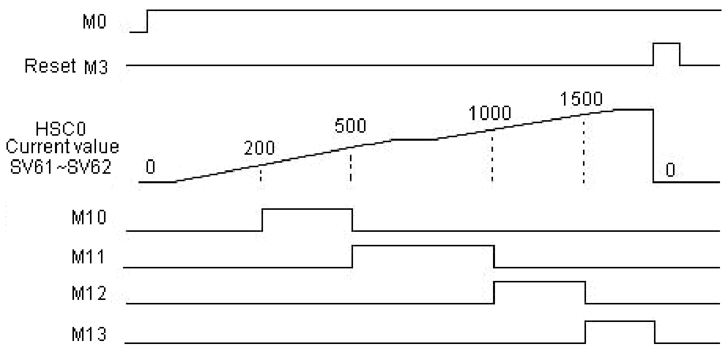
4. When M3=ON ,reset HSC0 ,HSC0 current value =0,HSC0 current segment number=1

[Instruction example 2]



[Program 2 schematic diagram]

PV component	Value	Declare
V1000V1001	200	First segment preset
V1002V1003	500	Second segment preset
V1004V1005	1000	Third segment preset
V1006V1007	1500	Fourth segment preset



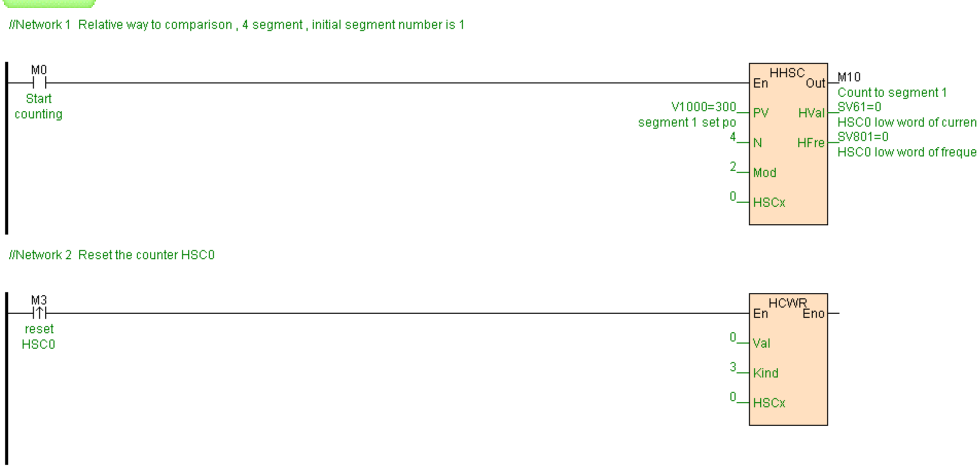
[program 2 declare]

1. When M0=ON ,HHSC instruction executing , high speed counter HSC0 set as absolute compare model,number of compare segments are 4.

2. When M3=ON ,reset HSC0 ,HSC0 current value =0,HSC0 current segment number=1

[Instruction example 3]

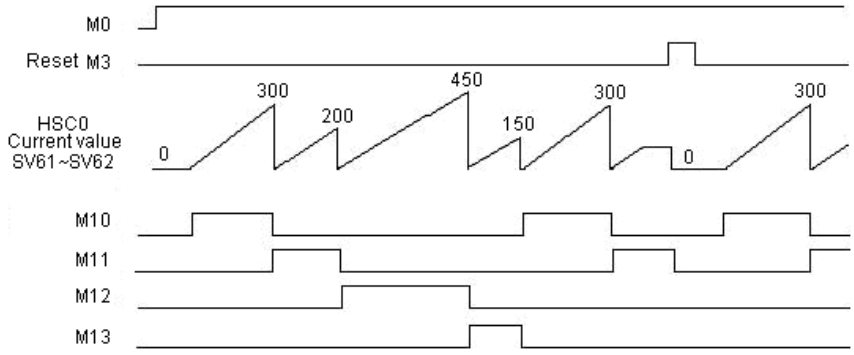
Download



[Program 3 schematic diagram]

PV component	Value	Declare
V1000V1001	300	First segment preset
V1002V1003	200	Second segment preset
V1004V1005	450	Third segment preset

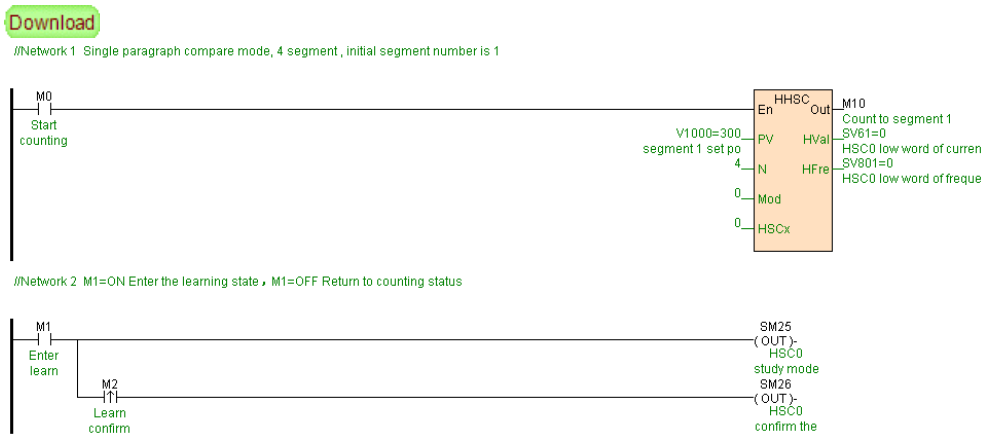
V1006V1007	150	Fourth segment preset



[Program 3 declare]

1. When M0=ON,HHSC instruction executing ,high speed counter HSC0 set as relatively compare model ,number of compare segments are 4.
2. When M3=ON ,reset HSC0 ,HSC0 current value=0,HSC0 current segment number=1

[Instruction example 4]



[Program 4 declare]

1. When M0=ON ,HHSC instruction executing, high speed counter HSC0 set as single segment compare Model ,number of compare segments 4.The initial segment of number 1.

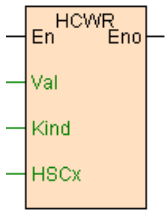
2. When M1=ON ,SM25=ON,HSC0 enter into study state, reset HSC0 at the same time.

3. When M2=ON,SM26=ON,HSC0 learn confirm that is write HSC0 current value to current segment preset value, segment number add 1 automatic go to next segment ,if segment number great than number of segments (this example N=4) then segment number =1.each SM26=ON once record one segment preset value .

4. At study finish ,reset M1=OFF,SM25=OFF,HSC0 quit study state , reset HSC0 at the same time.

HCWR(Write high speed counter)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			HCWR En, Val, Kind, HSCx	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Val	Value be wrote	√		

Parameter	Parameter define	Input	Output	Declare
Kind	Value type be wrote	√		0-3:0-write current segment,1-write current preset value,2-write high speed counter current value,3- reset high speed counter
HSCx	High speed counter number	√		
Eno	Enable output		√	

[Instruction function and effect declare]

1. HCWR instruction use for high speed counter assist control ,use cooperate HHSC instruction ,accomplish reset high speed counter . modify the preset value real time . modify high speed counter current value . modify current segment number.

2. If segment number be wrote exceed HSCx number of segment setting (HHSC instruction item N define) range ,HSCx report no.1 parameter error .

3. HCWR must be executed by edge .

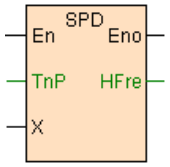
[Instruction example]

[Program description]

When M3=ON, reset high speed counter HSC0,HSC0 current value =0,current segment number =1.

SPD(Speed detection)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SPD En, TnP, X, HFre	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
TnP	Detection time or number of pulse	√		TnP＞0 is detection time (unit 0.1ms),TnP＜0 is detection number of pulse
X	pulse input	√		
Eno	Enable output		√	
HFre	Frequency value		√	Occupy 2 continuous component

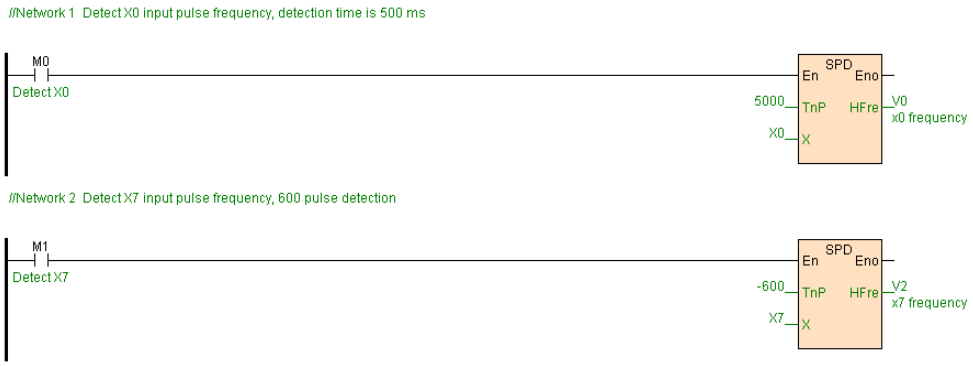
[Instruction function and effect declare]

1. SPD instruction detection the pulse frequency of the high speed input point XnInput of MPU .It nonuse HSCx high speed pulse input channel ,1 high speed pulse input channel have 2 high speed pulse input point .That is the MPU with 8 channels high speed pulse can realize detection 16 channels high speed input pulse .

2. SPD instruction support use time or number of pulse model to detection frequency , when TnP＞0 then use time model to detection frequency (unit 0.1ms), when TnP＜0 then use number of pulse model to detection frequency .TnP=0 then HFre=0.

3. For detection frequency ensure , when input pulse frequency great than 19KHz please use time model to detection (Suggestion detection time great than 500ms,TnP＞5000) , when input pulse frequency less than 19KHz please use number of pulse to detection frequency .

[Instruction example]



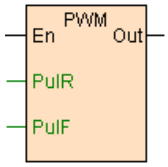
[Program description]

1. When M0=ON ,detect X0 pulse input frequency value ,detection time 5000*0.1ms=500ms, that is 500ms refresh once .

2. When M1=ON ,detect X7 pulse input frequency value, number of detection pulses are 600, that is 600 pulses detected refresh once .

PWM(Pulse width modulation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			PWM En, PulR, PulF, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PulR	Pulse duty factor	√		Unit 0.1%,range 0~1000
PulF	Pulse output frequency	√		Occupy 2 continuous
Out	Pulse output		√	

[Instruction function and effect declare]

1. PWM instruction output assigned frequency . duty factor pulse via high speed pulse output point Yn of MPU .It nonuse PLSx high speed pulse output channel ,1 high speed pulse output channel have 2 high speed pulse output point .That is the MPU with 8 channels high speed pulse can realize detection 16 channels high speed output pulse .

2. When PulF≤0 no pulse output , when PulF＜minimum frequency (10Hz) then use minimum frequency , when PulF＞ maximum frequency then use maximum frequency .

3. When PulF＞0,if PulR＞0 moreover PulR＜1000 ,Out output pulse of duty factor PulR . frequency Pul ,if PulR=0 then Out output low level ,if PulR≥1000 then Out output high level signal.

4. PulR. PulF value can modified real time.

[Instruction example]

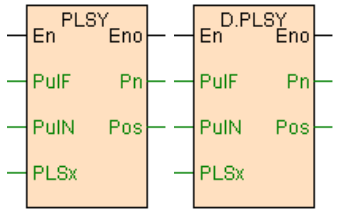
[Program description]

1. When M0=ON, from Y3 output pulse which duty factor is 30%. frequency is 50KHz pulse signal .

2. When M0=OFF, pulse output stop.

PLSY. D.PLSY(Pulse output)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			PLSY En, PulF, PulN, PLSx D.PLSY En, PulF, PulN, PLSx	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PulF	Pulse output frequency	√		
PulN	Number of pulse output	√		
PLSx	The channel of pulse output	√		
Eno	Enable output		√	
Pn	Number of pulse already output		√	Occupy 2 system register

Parameter	Parameter define	Input	Output	Declare
Pos	Current position		√	Occupy 2 system register

[Instruction function and effect declare]

1. PLSY is single segment pulse output instruction.

2. PulN is number of output pulse,PulN＞0 express output forward pulse ;PulN＜0 express output reverse pulse ;when PulN＝0 and relative address mode ,express output continuous pulse not take into account numbers.

3. PulF is pulse output frequency .PulN＝0 (output continuous pulse not take into account numbers), if PulF=0 then no output,PulF＞0 express output forward pulse,PulF＜0 express output reverse pulse. PulN≠0 ,if PulF≤0 then no output, PulF great than 0 however less than minimum frequency (10Hz) then according to minimum frequency output, PulF greater than maximum frequency then according to maximum frequency output

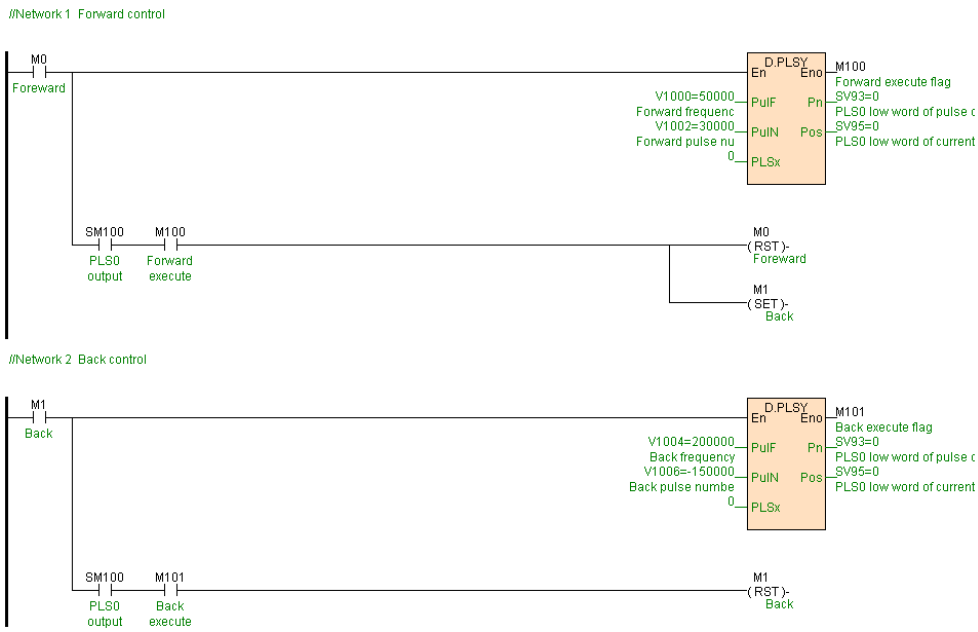
4. PLSY instruction relate to SM system state bit. SV system register, moreover generate pulse output interrupt , instruction start executing pulse output general "PLSx start output pulse " interrupt , instruction executed complete stop pulse output general " PLSx output complete" interrupt .

5. En=ON instruction executing ,Eno=ON; When pulse output process En go to OFF, then stop pulse output,Eno=OFF.

6. PLSY instruction there is not number of branches ,can coexist with others pulse output instruction ,but each pulse output channel only one instruction at the same time .

7. At instruction executing pulse output process, PulF pulse output frequency can be modified real time ,PulN can not be modified real time .

[Instruction example]

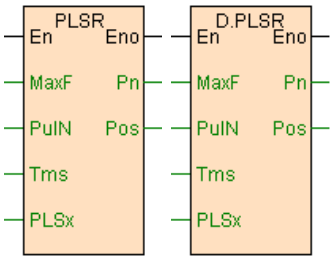


[Program description]

1. When M0=ON ,pulse output channel PLS0 use 50KHz frequency output 30000 forward pulse ,M100=ON ,output finish SM100=ON, reset M0,set M1.
2. When M1=ON ,pulse output channel PLS0 use 200KHz frequency output 150000 reverse pulse,M101=ON ,output finish SM100=ON ,reset M1.

PLSR. D.PLSR(Accelerate and decelerate pulse output)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			PLSR En, MaxF, PulN, Tms, PLSx D.PLSR En, MaxF, PulN, Tms, PLSx	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
MaxF	Pulse output maximum frequency	√		
PulN	Total output pulses	√		
Tms	Accelerate and decelerate time	√		(5~5000ms)
PLSx	Pulse output channel	√		
Eno	Enable output		√	
Pn	Number of pulse already output		√	Occupy 2 system register
Pos	Current position		√	Occupy 2 system register

[Instruction function and effect declare]

1. PLSR is have accelerate and decelerate single segment pulse output instruction .Tms accelerate and decelerate time , when Tms=0 express have no accelerate and decelerate , when Tms＜ minimum accelerate and decelerate time then according to minimum accelerate and decelerate time , when Tms＞ maximum accelerate and decelerate time then according to maximum accelerate and decelerate time.

2. MaxF is pulse output maximum frequency ,MaxFmust greater than 0,MaxF greater than 0 however less than minimum frequency (10Hz) then according minimum frequency output, MaxF greater than maximum frequency then

according to maximum frequency output,MaxF≤0 then no output, report no.3 parameter error.

3. PulN is total output pulses ,greater than 0 express output forward pulse ,less than 0express output reverse pulse, equal to 0 then no output, report no.3 parameter error .

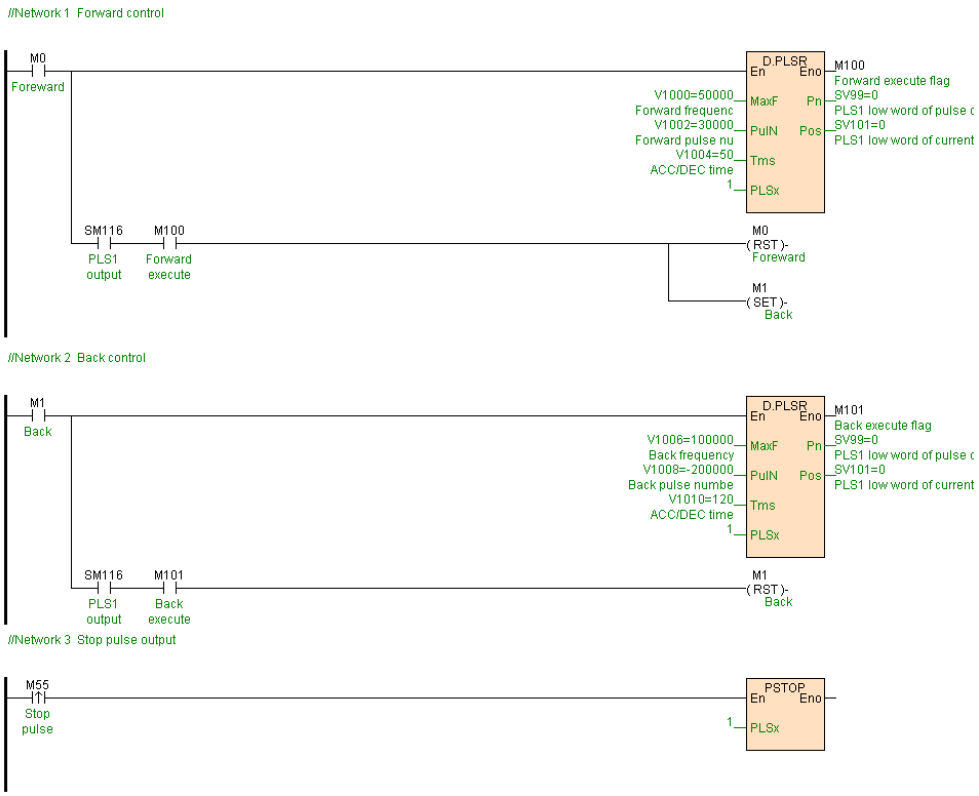
4. PLSR instruction relate to SM system state bit. SV system register , moreover general pulse output interrupt, instruction start executing pulse output general "PLSx start output pulse " interrupt , instruction executed complete stop pulse output general " PLSx output complete" interrupt .

5. En=ON instruction executing ,Eno=ON; When pulse output process En go to OFF, instruction according to Tms slow down brake stop (no setup brake stop),Eno=OFF.

6. PLSR instruction there is not number of branches ,can coexist with others pulse output instruction ,but each pulse output channel only one instruction at the same time .

7. At instruction executing pulse output process, MaxF. PulN can not be modified (not take effect real time be modified, need rerun instruction).

[Instruction example]



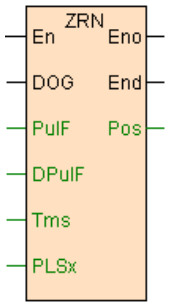
[Program description]

1. When M0=ON, pulse output channel PLS1 use 50KHz frequency output 30000 forward pulse ,accelerate and decelerate time 50ms,M100=ON, output finish SM116=ON, reset M0,set M1.
2. When M1=ON ,pulse output channel PLS1 use100KHz frequency output 200000 reverse pulse ,accelerate and decelerate time 120ms,M101=ON, output finish SM116=ON, reset M1.
3. When M55=ON ,pulse output channel PLS1 Stop pulse output.

ZRN(Origin point return)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

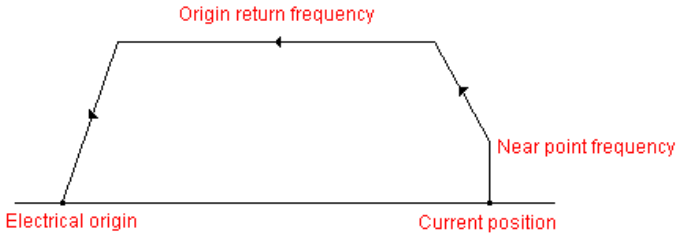
Language	LD	FBD	IL	Program example
Instruction format			ZRN En, DOG, PulF, DPulF, Tms, PLSx, End	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
DOG	Near point signal	√		
PulF	Origin return frequency	√		Occupy 2 continuous component
DPulF	Near point frequency	√		Occupy 2 continuous component
Tms	Accelerate and decelerate time	√		(5~5000ms)
PLSx	Pulse output channel	√		
Eno	Enable output		√	
End	The origin return to complete		√	
Pos	Current position		√	Occupy 2 system register

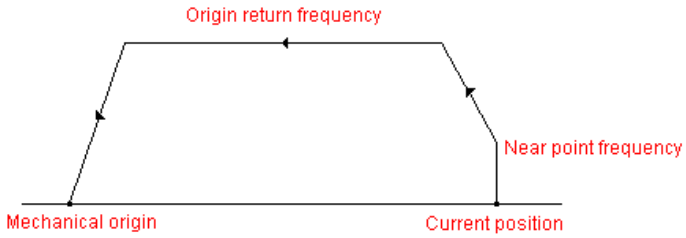
[Instruction function and effect declare]

1. When ZRN executing , according to near point frequency as the initial frequency ,accelerate to origin return frequency start moving,Slow down while approaching the origin to the near point frequency.

2. DOG = 0, said back to electrical origin.The current position > electrical origin, in order to reduce pulse direction,The current location < electric origin, in order to add pulse direction.After they back to the origin of the electric current position is set to 0.If the current location = electrical origin, the instruction is not action.



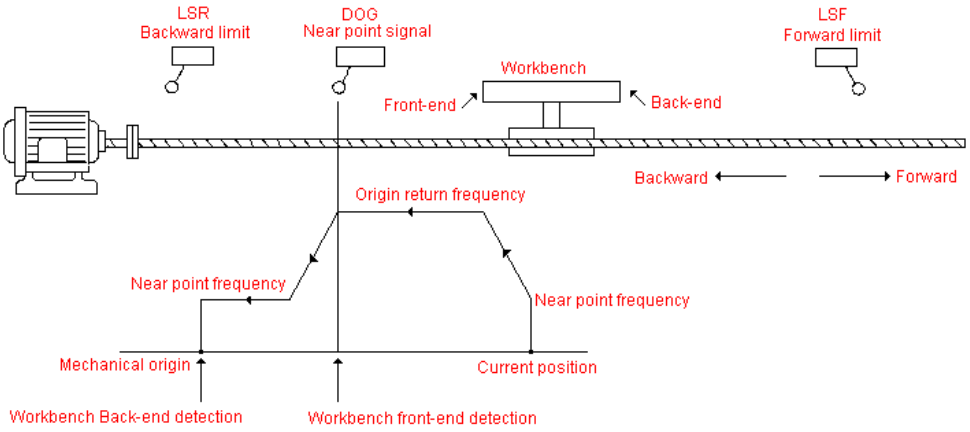
3. DOG = 1, said back to the mechanical origin, no near point signal.The current position > mechanical origin, in order to reduce pulse direction,The current location < mechanical origin, in order to add pulse direction.After they back to the origin of the mechanical ,Set the current position = mechanical origin.If the current location = mechanical origin, the instruction is not action.



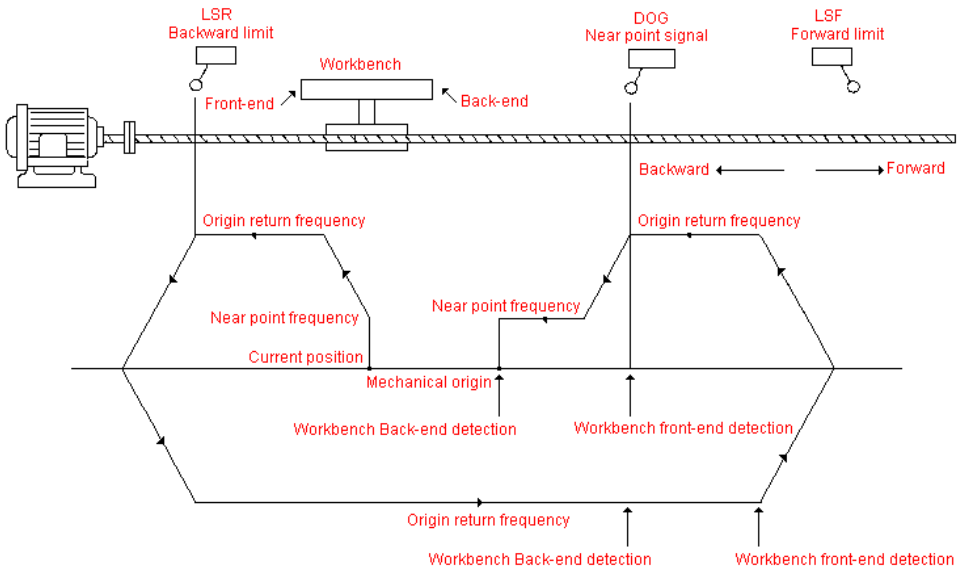
4. DOG terminal specified external input point X,Said use near point signal back to the mechanical origin.When using the DOG near point signal,ZRN instruction with the function of search DOG,ZRN fixed to reduce pulse direction.

a). The current position when the outside of the DOG,ZRN instruction from near point frequency acceleration origin return frequency ,when the near point signal (DOG) from OFF to ON,started to slow down to near point frequency (Crawl speed),when the near point signal (DOG) from ON to OFF,pulse output to stop,the origin return to

complete.After back to the mechanical origin, set the current position = mechanical origin.



b). The current location on the interior of the DOG or the DOG,ZRN instruction from near point frequency acceleration origin return frequency,until met LSR back limit switch to slow down to zero,then reverse forward;When the near point signal (DOG) from ON to OFF,slowed to zero and then back again;When the near point signal (DOG) from OFF to ON,started to slow down to near point frequency (Crawl speed),when the near point signal (DOG) from ON to OFF,pulse output to stop,the origin return to complete.After back to the mechanical origin, set the current position = mechanical origin.



5. When design the near point signal (DOG) , please think about DOG front and back have enough length , thus after sensed the DOG signal from OFF go to ON have enough time to decelerate to crawl frequency , no then result in position offset .

6. Near point signal (DOG) should connect to PLC MPU X input point , no then be influenced by scanning cycle result in position offset .

7. ZRN instruction relate to SM system state bit. SV system register .

8. When PulF=0,report no.3 parameter error no output pulse , no then when PulF ＜minimum frequency (10Hz) then use minimum frequency , when PulF＞ maximum frequency then use maximum frequency .

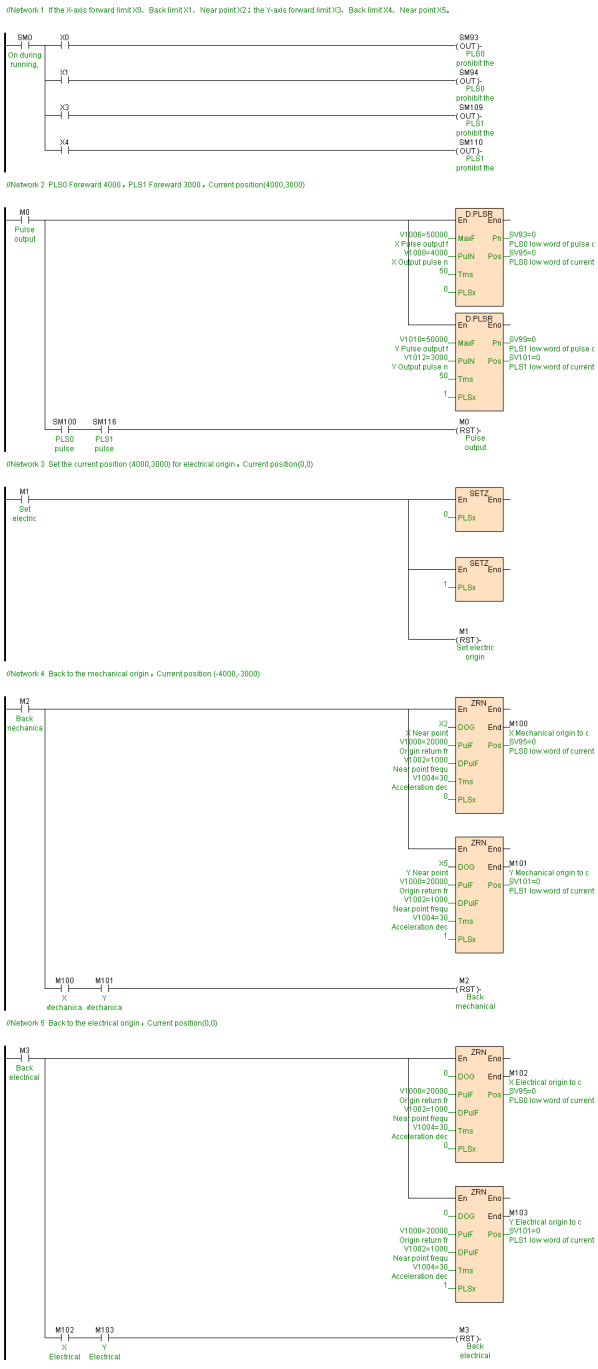
9. When DPulF＜minimum frequency (10Hz)or DPulF＞maximum frequency , that is near point output frequency overrange use minimum frequency .

10. When Tms＜minimum accelerate and decelerate time then use minimum accelerate and decelerate time, When Tms＞maximum accelerate and decelerate time then use maximum accelerate and decelerate time.

11. After the instruction started, all of parameter can not modified ,until instruction turn off .

12. When En=OFF ,all output actions will be stop immediately .

[Instruction example]



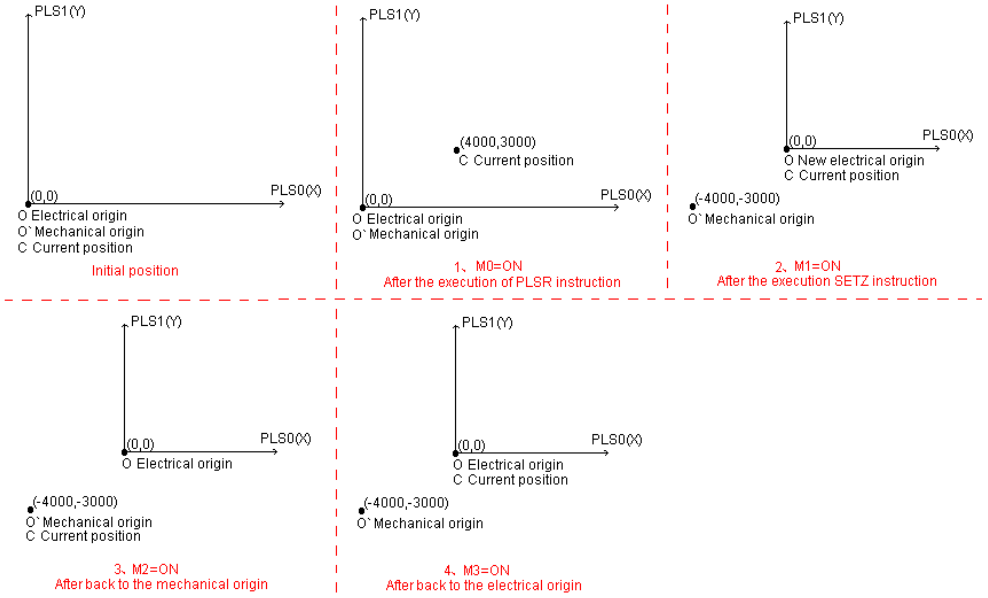
[Program description]

1. When M0=ON,PLS0 use 50KHz frequency output 4000 forward pulse ,output complete SM100=ON,PLS1 use 50KHz frequency output 3000 forward pulse ,output complete SM116=ON.reset M0

2. When M1=ON ,set (4000,3000) to electrical origin point, mechanical origin point changed to (-4000,-3000).

3. When M2=ON,PLS0 and PLS1 to back to the origin of 20 KHZ frequency, frequency of 1 KHZ near point back to the mechanical origin,current location (4000, 3000).

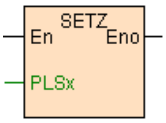
4. When M3=ON,PLS0 and PLS1 to back to the origin of 20 KHZ frequency, frequency of 1 KHZ near point back to the electrical origin,current location (0, 0).



SETZ(Set electric origin point)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

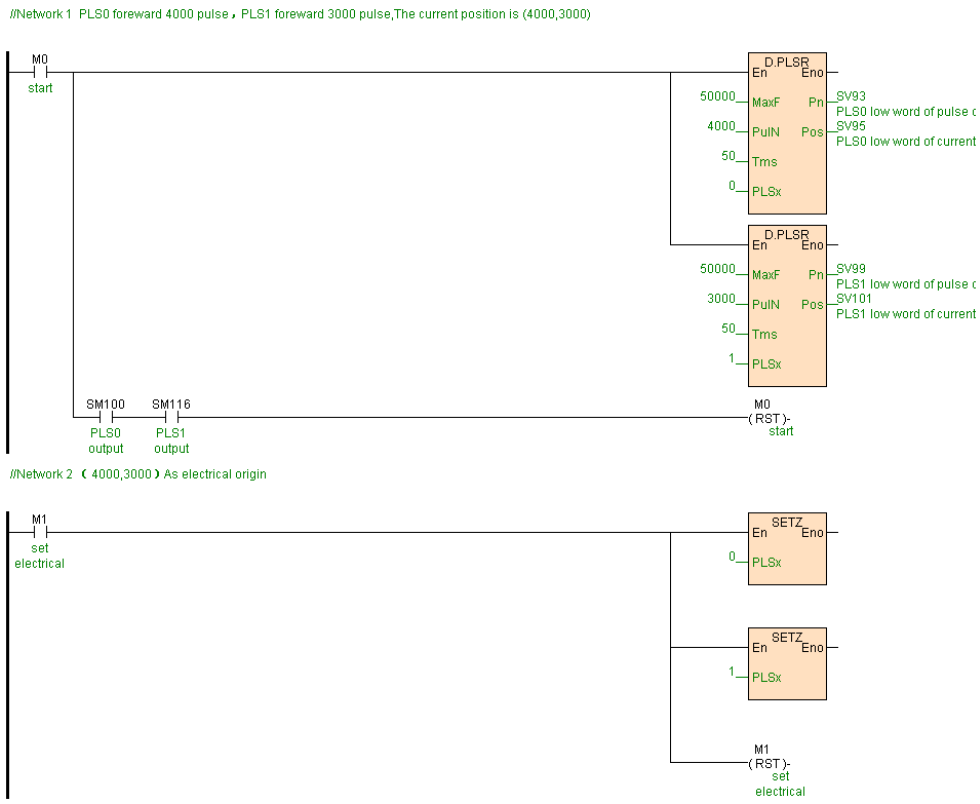
Language	LD	FBD	IL	Program example
Instruction format			SETZ En, PLSx	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PLSx	Pulse output channel	√		
Eno	Enable output		√	

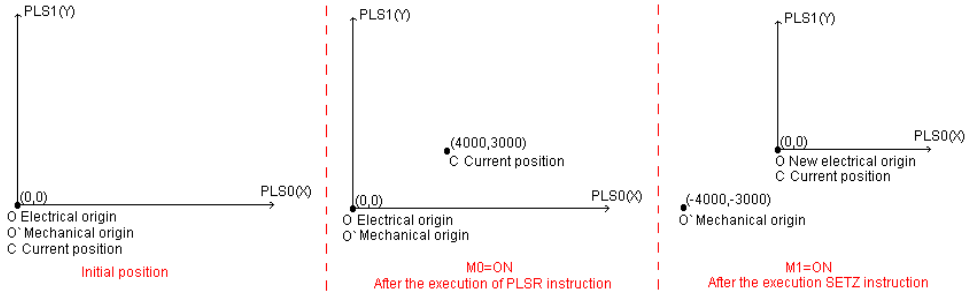
[Instruction function and effect declare]

When system initial ,electrical origin point is mechanical origin point .after executed SETZ instruction ,set current position to electrical origin point, current position clear zero.

[Instruction example]



[Program sketch map]

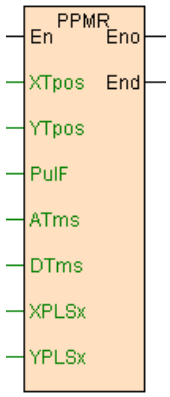


[Program description]

1. When M0=ON,PLS0 use 50KHz frequency output 4000 forward pulse ,output complete SM100=ON,PLS1 use 50KHz frequency output 3000 forward pulse ,output complete SM116=ON.reset M0
2. When M1=ON ,set (4000,3000) to electrical origin point, mechanical origin point changed to (-4000,-3000).

PPMR(Linear interpolation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			PPMR En, XTpos, YTpos, PulF, ATms, DTms, XPLSx, YPLSx, End	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
XTpos	X target location	√		Occupy 2 continuous component
YTpos	Y target location	√		Occupy 2 continuous component
PulF	Pulse output frequency	√		Occupy 2 continuous component
ATms	Accelerate time	√		(5~5000ms)
DTms	Decelerate time	√		(5~5000ms)
XPLSx	X pulse output channel	√		
YPLSx	Y pulse o utput channel	√		

Parameter	Parameter define	Input	Output	Declare
Eno	Enable output		√	
End	Li near interpolation complete		√	

[Instruction function and effect declare]

1. PPMR instruction use current position for starting point, use (XTpos,YTpos) end point , make linear interpolation output.

2. Starting point and end point can not the same point , no then report no.3 parameter error.

3. PPMR instruction relate to SM system state bit . SV system register.

4. When PulF=0,report no.3 parameter error no pulse output , no then When PulF ＜minimum frequency (10Hz) then use minimum frequency , when PulF＞ maximum then use maximum frequency.

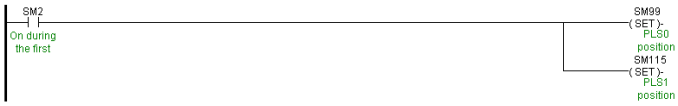
5. When ATms=0, it means there is no acceleration function. When ATms<0, it reports the error of the No.3 parameter. When ATms>maximum acceleration time or maximum deceleration time, then use the maximum acceleration time or the maximum deceleration time correspondingly. When DTms=0, it means no deceleration function. DTms<0 reports the error of the No.3 parameter. When DTms>maximum acceleration time or maximum deceleration time, then use the maximum acceleration time or the maximum deceleration time correspondingly.

6. After the instruction started, all of parameter can not modified ,until instruction turn off.

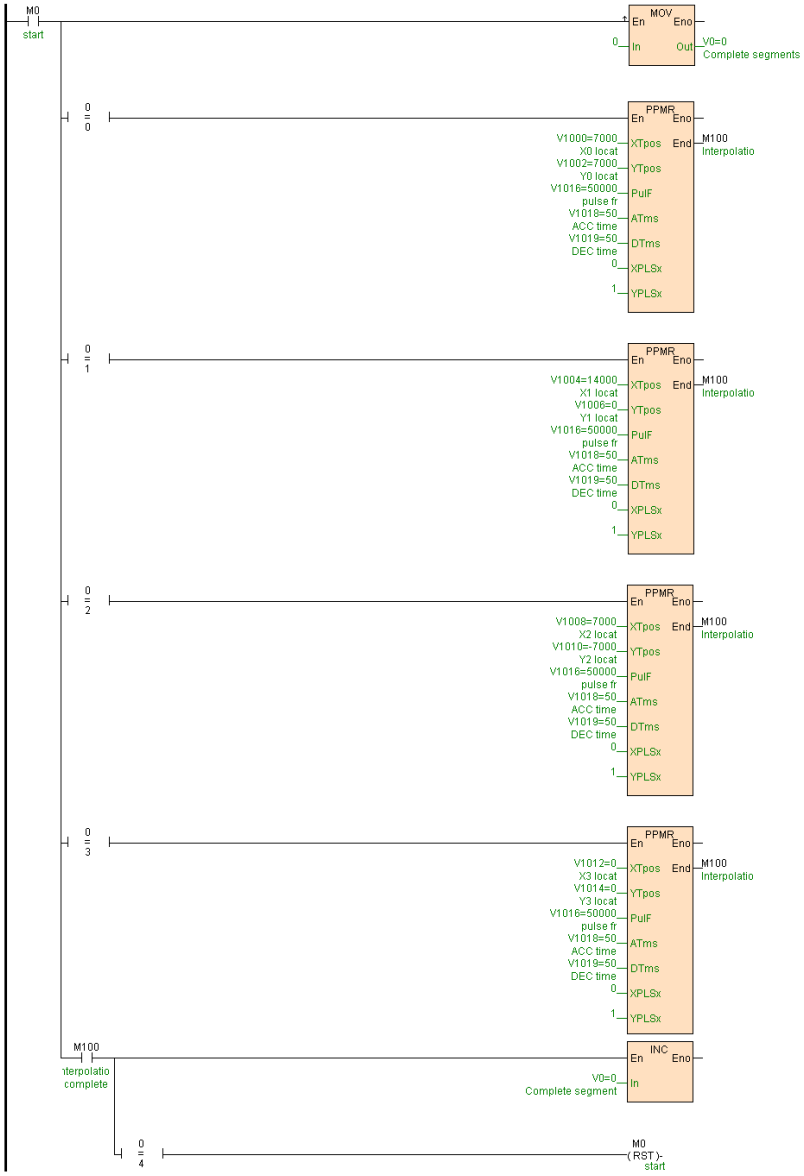
7. When En=OFF ,all output will be stop immediately.

[Instruction example]

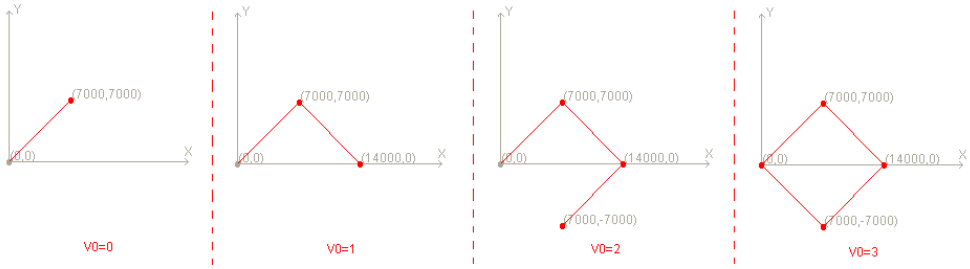
//Network 1 Set the PLS0, PLS1 positioning model for an absolute address



//Network 2 X axis is PLS0, Y axis is PLS1, linear interpolation control



[Program sketch map]



[Program description]

1. Via initial register table " linear interpolation parameter" setup 4 segment linear coordinate value .

Register component	Value	Declare
V1000V1001	7000	X target location0
V1002V1003	7000	Y target location0
V1004V1005	14000	X target location1
V1006V1007	0	Y target location1
V1008V1009	7000	X target location2
V1010V1011	-7000	Y target location2
V1012V1013	0	X target location3
V1014V1015	0	Y target location3
V1016V1017	50000	Pulse frequency
V1018	50	Accelerate time
V1019	50	Decelerate time

2. The program first scan cycle SM2=ON, setup PLS0. PLS1 position model to absolute address model(SM99=ON,SM115=ON).

3. X axis is PLS0,Y axis is PLS1, when M0=ON,V0=0, execute from current position (0,0) to (7000,7000) linear interpolation .

4. 1 segment interpolated complete M100=ON,V0=1, execute from (7000,7000) to (14000,0) linear interpolation .

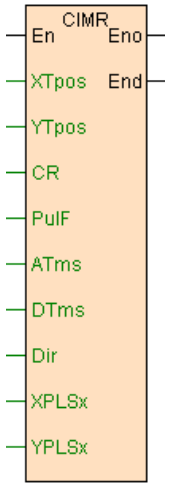
5. 2 segment interpolated complete M100=ON,V0=2, execute from(14000,0) to (7000,-7000) linear interpolation .

6. 3 segment interpolated complete M100=ON,V0=3, execute from(7000,-7000) to (0,0) linear interpolation .

7. 4 segment interpolated complete M100=ON,V0=4,reset M0,M0=OFF.

CIMR(Circular interpolation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			CIMR En, XTpos, YTpos, CR, PulF, ATms, DTms, Dir, XPLSx, YPLSx, End	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
XTpos	X target location	√		Occupy 2 continuous component
YTpos	Y target location	√		Occupy 2 continuous component
CR	Circle radius	√		Occupy 2 continuous component
PulF	Pulse output frequency	√		Occupy 2 continuous component

Parameter	Parameter define	Input	Output	Declare
ATms	Accelerate time	√		(5~5000ms)
DTms	Decelerate time	√		(5~5000ms)
Dir	Direction of motion	√		
XPLSx	X pulse output channel	√		
YPLSx	Y pulse o utput channel	√		
Eno	Enable output		√	
End	Circular interpolation complete		√	

[Instruction function and effect declare]

1. CIMR instruction use current position as starting point ,use (XTpos,YTpos) as end point , use CR as radius, make circular interpolation output.

2. Starting point and end point can not the same point , if radius less than half distance from staring point to end point (CR＜L / 2,L= distance from staring point to end point), then report no.3 parameter error.

3. CR＞0 express minor arc(the arc which less than semi-circle),CR＜0 express major arc (the arc which greater than semi-circle),Dir

is direction of motion

(0-clockwise,1-anticlockwise)

4. CIMR instruction relate to SM system state bit. SV system register.

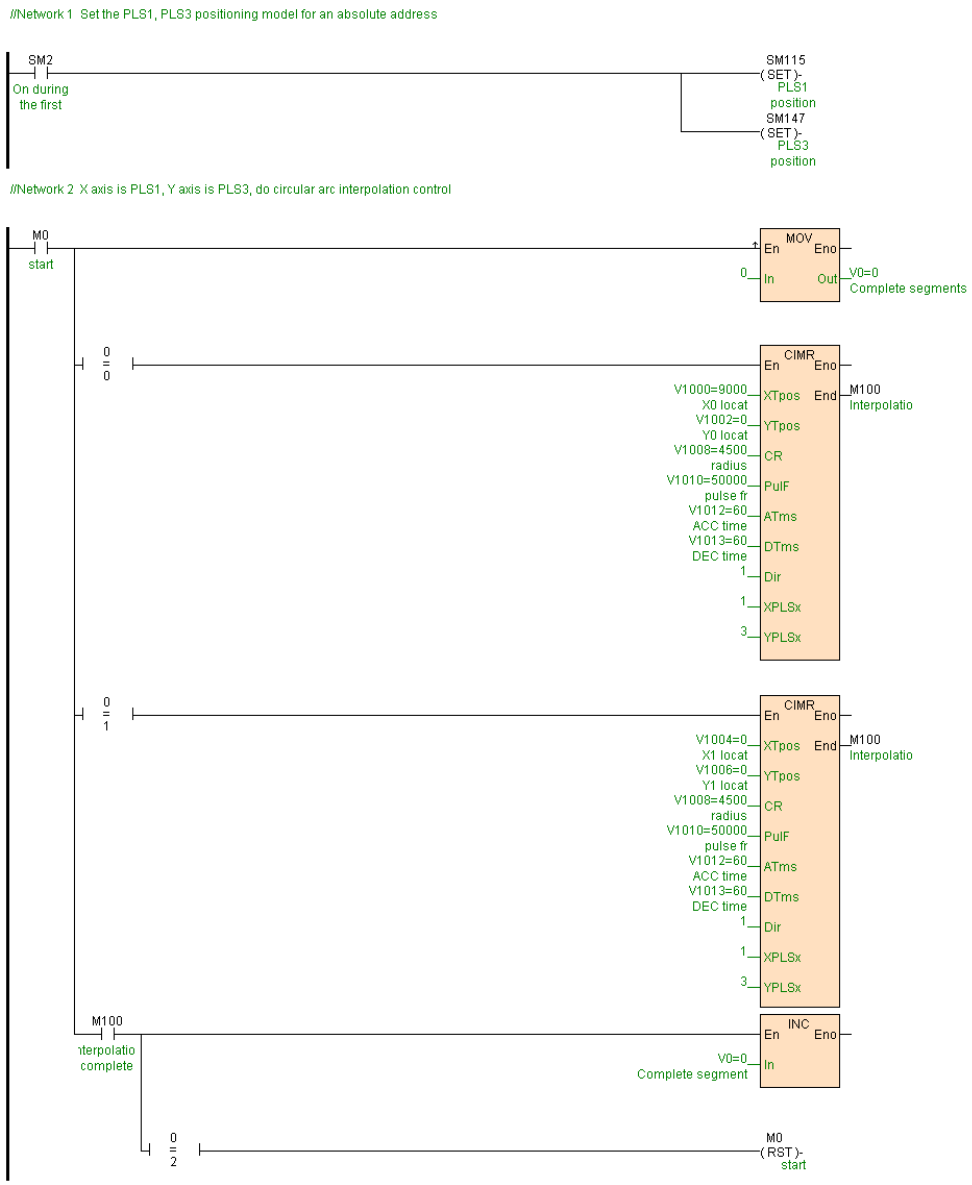
5. When PulF=0,report no.3 parameter error no pulse output, no then when PulF＜ minimum frequency (10Hz) then use minimum frequency , when PulF＞ maximum frequency then use maximum frequency .

6. When ATms=0, it means there is no acceleration function. When ATms<0, it reports the error of the No.3 parameter. When ATms>maximum acceleration time or maximum deceleration time, then use the maximum acceleration time or the maximum deceleration time correspondingly. When DTms=0, it means no deceleration function. DTms<0 reports the error of the No.3 parameter. When DTms>maximum acceleration time or maximum deceleration time, then use the maximum acceleration time or the maximum deceleration time correspondingly.

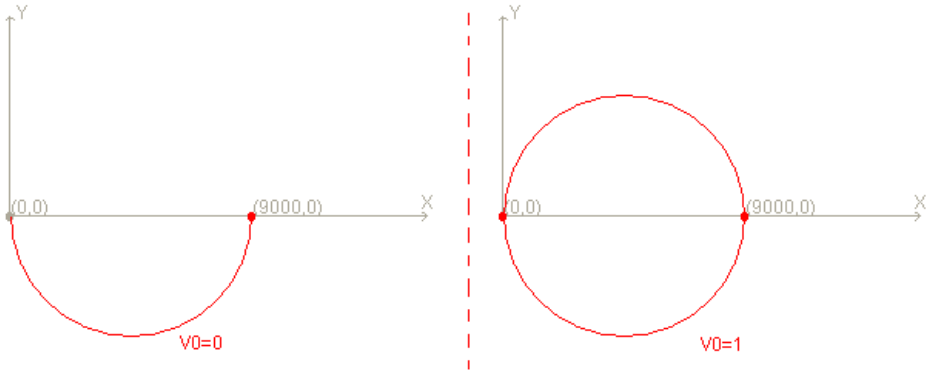
7. After the instruction started, all of parameter can not modified ,until instruction turn off.

8. When En=OFF ,all output will be stop immediately.

[Instruction example]



[Program sketch map]



[Program description]

1. Via initial register table " circular interpolation parameter" setup 2 segment arc coordinate value.

Register component	Value	Declare
V1000V1001	9000	X target location0
V1002V1003	0	Y target location0
V1004V1005	0	X target location1
V1006V1007	0	Y target location1
V1008V1009	4500	Circle radius
V1010V1011	50000	Pulse frequency
V1012	60	Accelerate time
V1013	60	Decelerate time

2. The program first scan cycle SM2=ON, set PLS1. PLS3 position model absolute addressModel(SM115=ON,SM147=ON).

3. X axis is PLS1,Y axis is PLS3, when M0=ON,V0=0,executing anticlockwise circular interpolation according to current position(0,0) for starting point (9000,0) radius is 4500.

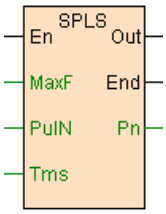
4. 1 segment circular interpolation completeM100=ON,V0=1,executing anticlockwise circular interpolation from (9000,0) to (0,0) radius is 4500 .

5. 2 segment circular interpolation completeM100=ON,V0=2,reset M0,M0=OFF.

SPLS(Single pulse output)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			SPLS En, MaxF, PulN, Tms, Out, End, Pn	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
MaxF	Pulse output maximum frequency	√		Occupy 2 continuous component
PulN	Total number of pulse output	√		Occupy 2 continuous component
Tms	Accelerate and decelerate time	√		(5~5000ms)
Out	Pulse output		√	
End	Pulse output complete		√	
Pn	Number of pulse already output		√	Occupy 2 continuous component

[Instruction function and effect declare]

1. SPLS instruction via MPU high speed pulse output point Yn output high speed pulse, it not use PLSx high speed pulse output channel,1 high speed pulse output channel have 2 high speed output points. Thus MPU with 8 pulse output channel can carry out 16 high speed pulse outputs.

2. SPLS instruction only output pulse, direction need controlled by Y output point in the program ,it will not generate pulse output interruption ,not use SM. SV.

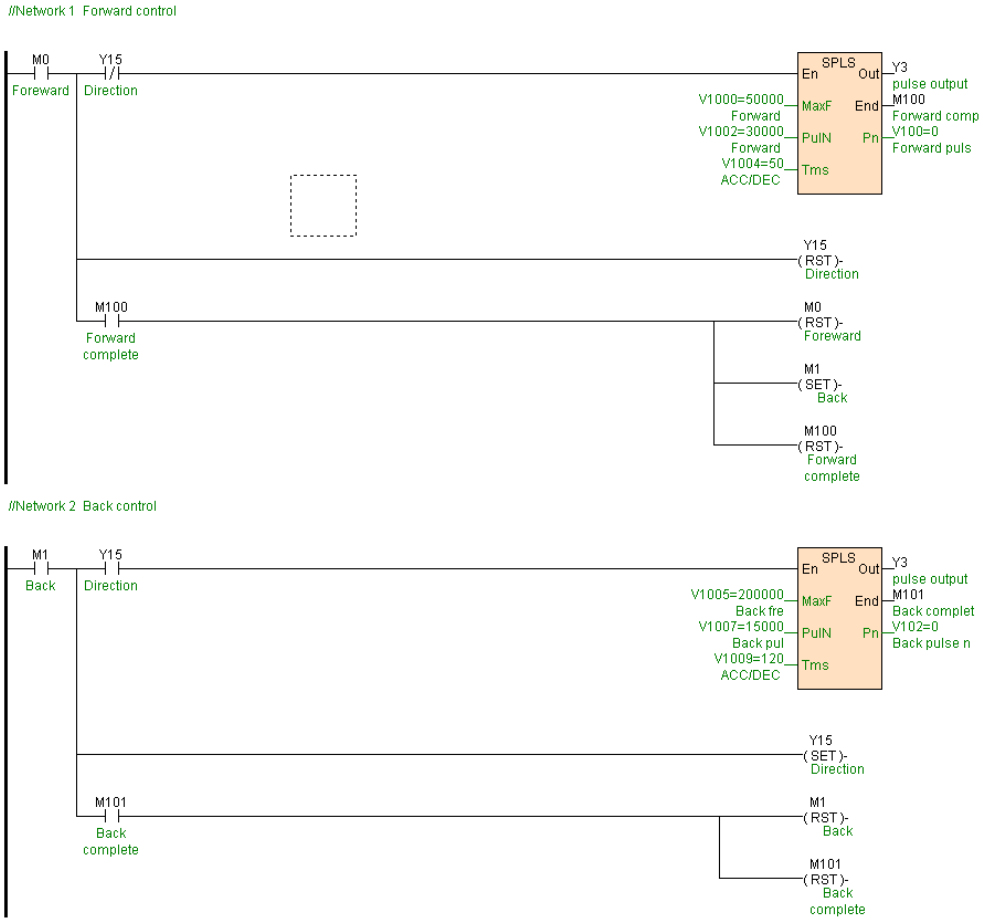
3. When Tms=0 express no accelerate and decelerat, no then when Tms＜ minimum accelerate and decelerate time then use minimum accelerate and decelerate time, when Tms＞maximum accelerate and decelerate time then use maximum accelerate and decelerate time.

4. When MaxF≤0 no pulse output , no then when MaxF＜ minimum frequency (10Hz) then use minimum frequency , when MaxF＞ maximum frequency then use maximum frequency .

5. When PulN=0 express not take into account pulse output ;when PulN＞0 then according to PulN pulses output, pulse output complete End set; when PulN＜0 no pulse output .

6. MaxF can be modified real time ,PulN can not be modified real time .

[Instruction example]

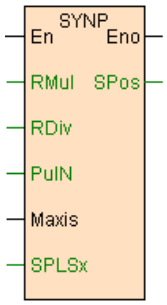


[Program description]

1. When M0=ON ,Y3 use 50KHz frequency output 30000 pulses ,Y15=OFF(OFF express forward direction),M100=OFF
2. When pulse output complete,M100=ON, reset M0, set M1.
3. When M1=ON ,Y3 use 200KHz frequency output 15000 pulses ,Y15=ON (ON express reverse direction),M101=OFF
4. When pulse output complete ,M101=ON, reset M1.

SYNP(Synchronization pulse output)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SYNP En, RMul, RDiv, PulN, Maixs, SPLSx	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
RMul	Multiplying factor	√		Occupy 2 continuous component
RDiv	D ivide factor	√		Occupy 2 continuous component
PulN	Number of delay pulses	√		
Maixs	Main axis	√		
PLSx	Pulse output channel	√		
Eno	Enable output		√	
SPos	Slave current position		√	Occupy 2 system register

[Instruction function and effect declare]

1.SYNP instruction realizes electronic gear function, specifing that the slave axis SPLSx pulse output follows the change of the pulse of master axis Maxis, the ratio

between them coefficient K = RMul / RDiv. SPLSx direction consistent with the direction of the spindle Maxis, RMul and RDiv must be greater than zero.

2.Maxis is defined as the master axis. When Maxis are specified as X0, X2, X4, X6, X8, X10, X12 and X14, they indicate the high-speed pulse input channels HSC0~ HSC7.

3.PulN specifies the number of delay pulses. After Maxis inputs "pulN" pulses, SPLSx starts to follow Maxis. If PulN ≤0,it indicates no delay.

4.When multiple SYNP instructions execute at the same time , Maxis can be repeated, SPLSx can not be repeated, that is to say a master axis can drive multiple slave axises.

5.SYNP instruction relate to SM system state bit . SV system register , If Maxis is high-speed pulse input channel, it will generate high-speed counter direction change interrupt.

6.SYNP instruction does not have the limitation on the number, it can coexist with other pulse output instructions, but each pulse output channel can only execute one instruction at the same time.

7.After this instruction is started, the parameters RMul and RDiv can't be modified in real time.

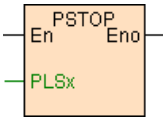
[Instruction example]

[Program description]

1. X10=ON, then M0=ON,SYNP instruction executing ,PLS1 after delay 15 pulses ,use 1:1 HSC0 pulse input to output.
2. X11=ON, then M0=OFF, SYNP instruction executing stop.

PSTOP(Stop pulse output)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			PSTOP En, PLSx	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PLSx	Pulse output channel	√		
Eno	Enable output		√	

[Instruction function and effect declare]

PSTOP instruction refer to PLSx Stop pulse output .

[Instruction example]

Refer to PLSR instruction example.

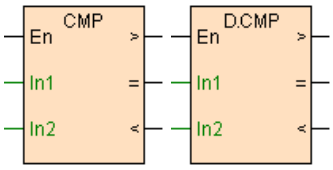
Compare instruction

Compare instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
CMP		D.CMP	Compare instruction	√	√	√
ZCP		D.ZCP	Regional comparison	√	√	√
MATC		D.MATC	Numerical value match	√	√	√
ABSC		D.ABSC	Absolute cam comparison	√	√	√
BON			ON bit determine	√	√	√
BONC		D.BONC	ON bit numbers	√	√	√
MAX		D.MAX	Maximum	√	√	√
MIN		D.MIN	Minimum	√	√	√
SEL		D.SEL	Selection of conditions	√	√	√
MUX		D.MUX	Multi-choice	√	√	√

CMP. D.CMP(Compare)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			CMP En, In1, In2, Out D.CMP En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Input1	√		
In2	Input2	√		
Out	Status output		√	Occupy 3 continuous component

[Instruction function and effect declare]

CMP is 16 bit integer compare instruction (D.CMP is 32 bit integer compare), at the same time output＞. =. ＜ these 3 result.

[Instruction example]

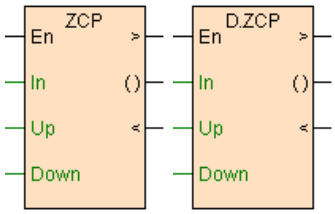
[Program description]

1. 16 bit compare CMP instruction, when AI1>500 then M10=ON, when AI1=500 then M11=ON, when AI1<500 then M12=ON.

2. 32 bit compare D.CMP instruction, when V10V11>V0V1 then M20=ON, when V10V11=V0V1 then M21=ON, when V10V11<V0V1 then M22=ON.

ZCP. D.ZCP(Regional comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ZCP En, In, Up, Down, Out D.ZCP En, In, Up, Down, Out	Download

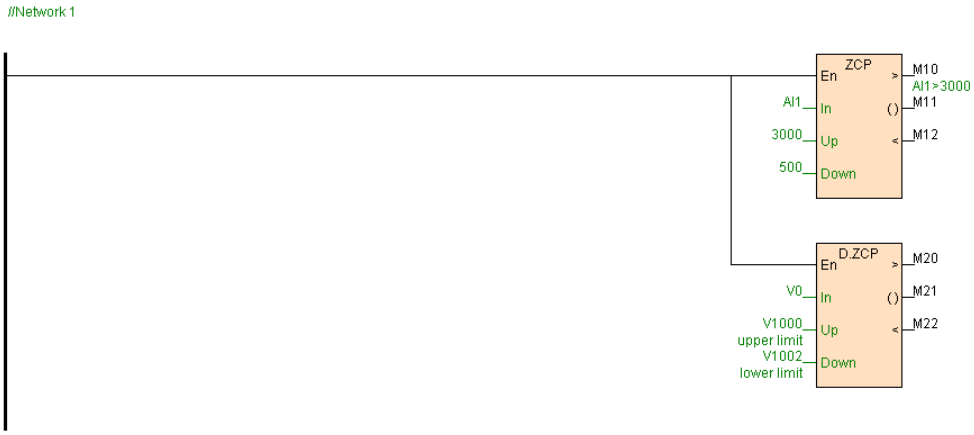
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Up	Regional upper limit	√		
Down	Regional lower limit	√		
Out	Status output		√	Occupy 3 continuous component

[Instruction function and effect declare]

1. ZCP is 16 bit integer regional compare instruction(D.CMP is 32 bit integer regional compare),at the same time output ＞. =. ＜ these 3 result.

2. If regional upper limit Up < regional lower limit Down, then instruction will swap them automatic .

[Instruction example]



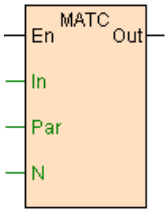
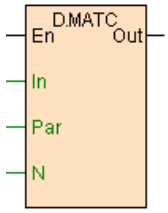
[Program description]

1. 16 bit compare ZCP instruction , when AI1>3000 then M10=ON, when AI1≤3000 moreover AI1≥500 then M11=ON, when AI1<500 then M12=ON.

2. 32 bit compare D.ZCP instruction, when V0V1>V1000V1001 then M20=ON, when V0V1≤V1000V1001 moreover V0V1≥V1002V1003 then M21=ON, when V0V1＜V1002V1003 then M22=ON.

MATC. D.MATC(Numerical match)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MATC En, In, Par, N, Out D.MATC En, In, Par, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Par	Numerical value match start component	√		MATC: occupy N continuous component, D.MATC: occupy 2N continuous component
N	Number to compare	√		1~256
Out	Status output		√	

[Instruction function and effect declare]

MATC instruction compare In input and N data start from Par ,if In equal to one of then express match right Out=ON, no then Out=OFF.

[Instruction example]

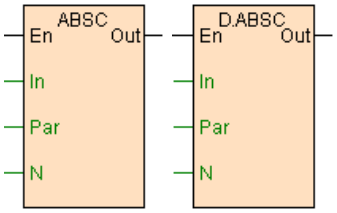
[Program description]

1. MATC instruction get electricity from busbar and always execute .

2. As long as V0 equal to one of V1000. V1001. V1002. V1003. V1004. V1005 then match right M0=ON, no then M0=OFF.

ABSC. D.ABSC(Absolute cam comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ABSC En, In, Par, N, Out D.ABSC En, In, Par, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Par	Multi-segment compare start component	√		ABSC: occupy 2N c ontinuous component ,D.ABSC: occupy 4 continuous component
N	Number of compare segments	√		1- 64
Out	Start address of compare result		√	Occupy N continuous component

[Instruction function and effect declare]

1. ABSC instruction region compare In input N segment data start from Par ,compare results output to start from out N continuous component.

2. To the segment which lower limit ≤ upper limit , if lower limit≤In≤ upper limit, then compare result of the segment Output=ON, if In＜lower limit or In＞upper limit, then compare result of the segment Output =OFF.

3. To the segment which lower limit＞upper limit,if upper limit≤In≤ lower limit, then compare result of the segment Output=OFF, if In＞lower limit or In＜upper limit, then compare result of the segment Output=ON.

4. Parameter Par and N relation declare:

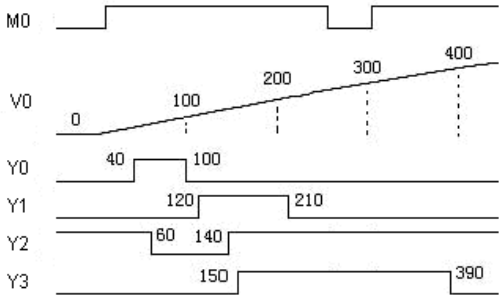
Number	Number of compare segments N	Par component meaning	Out component meaning
1	1	1 segment lower limit	1 segment compare result output
2		1 segment upper limit	
3	2	2 segment lower limit	2 segment compare result output
4		2 segment upper limit	
…	…	…	…
15	8	8 segment lower limit	8 segment compare result output
16		8 segment upper limit	

[Instruction example]

[Program sketch map]

Par component	Value	Declare
V1000	40	1 segment lower limit
V1001	100	1 segment upper limit
V1002	120	2 segment lower limit
V1003	210	2 segment upper limit
V1004	140	3 segment lower limit
V1005	60	3 segment upper limit

V1006	150	4 segment lower limit
V1007	390	4segment upper limit

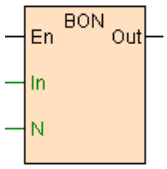


[Program description]

1. M0=ON,region compare V0 and start from V1000 4 segment.
2. When V0 is 40~100,Y0=ON, no then Y0=OFF; when V0 is 120~210 ,Y1=ON, no then Y1=OFF; when V0 is 60~140 ,Y2=OFF, wo then Y2=ON; when V0 is 150~390,Y3=ON, no then Y3=OFF.
3. When M0=OFF, instruction stop execute ,Y0~Y3 remain unchanged.

BON(ON bit determine)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BON En, In, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable			

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
N	Which bit	√		1~16
Out	Status output		√	

[Instruction function and effect declare]

BON instruction use to determine the bit of the register whether or not 1,result output to Out.

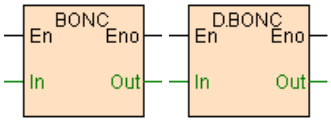
[Instruction example]

[Program description]

1. BON instruction get electricity from busbar and always execute
2. If V0=8(binary 00000000 00001000, fourth bit is 1), then M0=ON.

BONC. D.BONC(ON bit numbers)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			BONC En, In, Out D.BONC En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

BONC instruction get the number which the bit is 1 of the register, result output to Out.

[Instruction example]

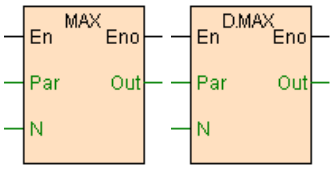
[Program description]

1. BONC instruction get electricity from busbar and always execute
2. If V0=1234(binary 00000100 11010010, total 5 bits are 1),then V100=5.

MAX. D.MAX(Maximum)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MAX En, Par, N, Out D.MAX En, Par, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Par	Compare value start component	√		MAX: occupy N continuous component, D.MAX: occupy 2N continuous component
N	Number data for compare	√		2~256
Eno	Enable output		√	
Out	Compare result output		√	

[Instruction function and effect declare]

MAX instruction compare N data start from Par ,maximum output to Out.

[Instruction example]

[Program description]

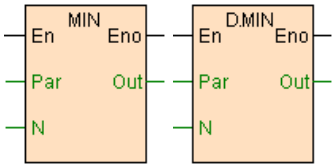
1. M0=ON, if V1000=30. V1001=-150. V1002=25. V1003=8. V1004=95. V1005=-20, then V0=95.

2. M0=ON, if V1100V1101=30000. V1102V1103=-50000. V1104V1105=23000. V1106V1107=600. V1108V1109=1500, then V10V11=30000.

3. When M0=OFF, instruction stop execute ,Out remain unchanged.

MIN. D.MIN(Minimum)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MIN En, Par, N, Out D.MIN En, Par, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Par	Compare value start component	√		MIN: occupy N continuous component, D.MIN: occupy 2N continuous component
N	Number data for compare	√		2~256
Eno	Enable output		√	
Out	Compare result output		√	

[Instruction function and effect declare]

MIN instruction compare N data start from Par ,minimum output to Out.

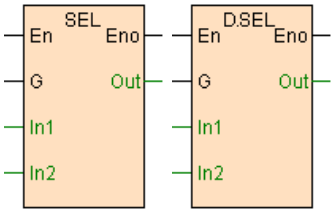
[Instruction example]

[Program description]

1. M0=ON, if V1000=30. V1001=-150. V1002=25. V1003=8. V1004=95. V1005=-20, then V0=-150.
2. M0=ON, if V1100V1101=30000. V1102V1103=-50000. V1104V1105=23000. V1106V1107=600. V1108V1109=1500, then V10V11=-50000.
3. When M0=OFF, instruction stop execute ,Out remain unchanged.

SEL. D.SEL(Selection of conditions)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			SEL En, G, In1, In2, Out D.SEL En, G, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
G	selection condition	√		

Parameter	Parameter define	Input	Output	Declare
In1	select data 1	√		
In2	select data 2	√		
Eno	Enable output		√	
Out	Select result output		√	

[Instruction function and effect declare]

SEL instruction is either-or instruction, G=OFF then Out=In1,G=ON then Out=In2.

[Instruction example]

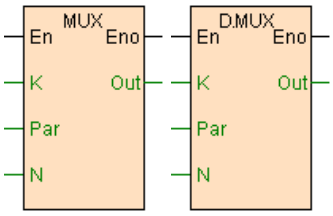
[Program description]

1. SEL instruction get electricity from busbar and always execute.
2. If AI0=230. AI1=512, M0=OFF then V0=230,M0=ON then V0=512.

MUX. D.MUX(Multi-choice)

Instruction format and parameter specification

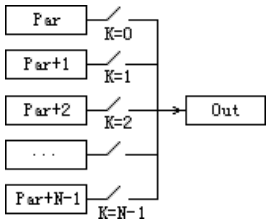
Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MUX En, K, Par, N, Out D.MUX En, K, Par, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
K	Select channel	√		
Par	Select data start component	√		MUX: occupy N continuous component ,D.MUX: occupy 2N continuous component
N	Number of data be selected	√		1~256
Eno	Enable output		√	
Out	Select result o utput		√	

[Instruction function and effect declare]

MUX instruction select one of data according to the value of select channel K(K=0~N-1) from N address continuous register output to Out (multi select one). schematic diagram as follows:



[Instruction example]

[Program description]

1. MUX instruction get electricity from busbar and always execute.
2. If V1000=30. V1001=-150. V1002=25. V1003=8. V1004=95. V1005=-20, when V00=3 express select fourth value output,V10=8.

Shift instruction

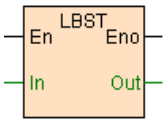
Shift instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
LBST			Low byte evaluation	√	√	√
HBST			High byte evaluation	√	√	√
MOV		D.MOV	Move	√	√	√
BMOV			Block move	√	√	√
FILL			Fill	√	√	√
XCH			Byte swap	√	√	√
BXCH			Block swap	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
SHL			Bit left shift	√	√	√
SHR			Bit right shift	√	√	√
WSHL			Word left shift	√	√	√
WSHR			Word right shift	√	√	√
ROL			Bit rotate left shift	√	√	√
ROR			Bit rotate right shift	√	√	√
WROL			Word rotate left shift	√	√	√
WROR			Word rotate right shift	√	√	√
BSHL			Byte left shift	√	√	√
BSHR			Byte right shift	√	√	√
ATBL			Append to array	√	√	√
FIFO			First in first out	√	√	√
LIFO			Last in first out	√	√	√
SORT			Data sort	√	√	√

LBST(Low byte evaluation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			LBST En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Data output		√	

[Instruction function and effect declare]

LBST use for specified assignment to the low byte of output register Out , high byte remain unchanged.

[Instruction example]

[Program description]

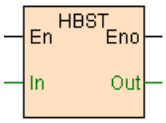
If initial V1000=0x1E34,then M0=ON V1000=0x1E0C.

HBST(High byte evaluation)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			HBST En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Data output		√	

[Instruction function and effect declare]

HBST use for specified assignment to the high byte of output register Out , low byte remain unchanged.

[Instruction example]

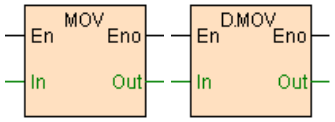
[Program description]

If initial V1001=0xFF6A,then M1=ON V1001=0x856A

MOV. D.MOV(Move)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MOV En, In, Out D.MOV En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	DataOutput		√	

[Instruction function and effect declare]

Move instruction MOV also call assign instruction, use for assign the specified data to output register Out.

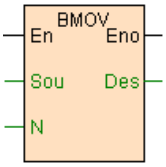
[Instruction example]

[Program description]

Assign the initial value will the program first scan cycle,V0=80,V10V11=-50.

BMOV(Block move)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BMOV En, Sou, N, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Block move start address component	√		Occupy N continuous component
N	Number to be moved	√		1~256
Eno	Enable output		√	
Des	Target block move start address component		√	Occupy N continuous component

[Instruction function and effect declare]

BMOV block move instruction move N components start from Sou to N components start Des .As follows:

[Instruction example]

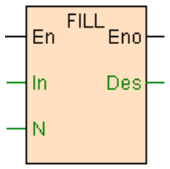
[Program description]

When M0=ON, Move V1000~V1005 to V0~V5, Move X0~X4 to Y2~Y6.

Sou component	Initial value	Move result
V1000	30	V0=30
V1001	-150	V1=-150
V1002	25	V2=25
V1003	8	V3=8
V1004	95	V4=95
V1005	-20	V5=-20
X0	ON	Y2=ON
X1	OFF	Y3=OFF
X2	ON	Y4=ON
X3	ON	Y5=ON
X4	OFF	Y6=OFF

FILL(FiIl)

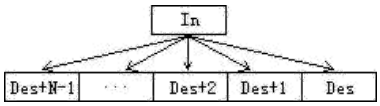
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FILL En, In, N, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Data to fill	√		
N	Number of fill data	√		1~256
Eno	Enable output		√	
Des	Move target block start component		√	Occupy N continuous component

[Instruction function and effect declare]

Fill instruction FILL use for fill the In value into Des start N component.Can use for batch reset . set register component and bit component.As follows:



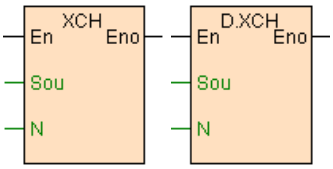
[Instruction example]

[Program description]

When M0=ON, Reset V100~104 5 registers to 0, reset Y0~Y11 to OFF, set M100~M105 to ON.

XCH(Byte swap). D.XCH(Register swap)

Instruction format and parameter specification

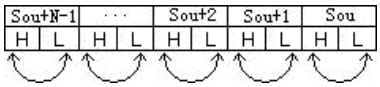
Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			XCH En, Sou, N D.XCH En, Sou, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Start register to swap	√		Occupy N continuous component
N	Number of swap registers	√		1~256

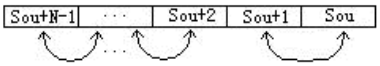
Parameter	Parameter define	Input	Output	Declare
Eno	Enable output		√	

[Instruction function and effect declare]

1. 16 bit instruction XCH is byte swap , use for start from Sou N registers swap the high low byte .As follows:



2. 32 bit instruction D.XCH is register swap ,use for start from Sou N register component each border upon 2 register swap, if N is odd then the last one register unchanged.As follows:



3. XCH

instruction general executed by edge.

[Instruction example]

[Program description]

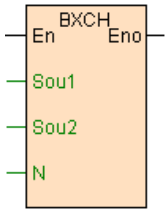
When M0=ON, Swap high low byte of V1000. V1001 these 2 registers ,swap V1002. V1003 these 2 registers ,swap V1004. V1005 these 2 registers .

Sou c	Initial value	Sw

omponent		ap result
V1000	30 (0x001E)	V1000=7680 (0x1E00)
V1001	-150 (0xFF6A)	V1001=27391(0x6AFF)
V1002	25	V1002=8
V1003	8	V1003=25
V1004	95	V1004=-20
V1005	-20	V1005=95

BXCH(Block swap)

Instruction format and parameter specification

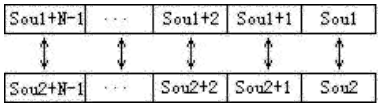
Language	LD	FBD	IL	Program example
Instruction format			BXCH En, Sou1, Sou2, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou1	Source 1 start component	√		Occupy N continuous component
Sou2	Source 2 start component	√		Occupy N continuous component

Parameter	Parameter define	Input	Output	Declare
N	Number of component	√		1~256
Eno	Enable output		√	

[Instruction function and effect declare]

1. Block swap instruction BXCH use for start from Sou1 N components and start from Sou2 N components.As follows:



2. BXCH

instruction general executed by edge.

[Instruction example]

[Program description]

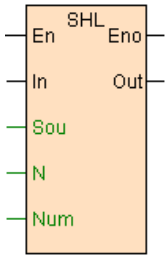
When M0=ON, Swap V1000~V1002 these 3registers and V1003~V1005 these 3 register.

component	Initial value	Swap result
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20

V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25

SHL(Bit left shift)

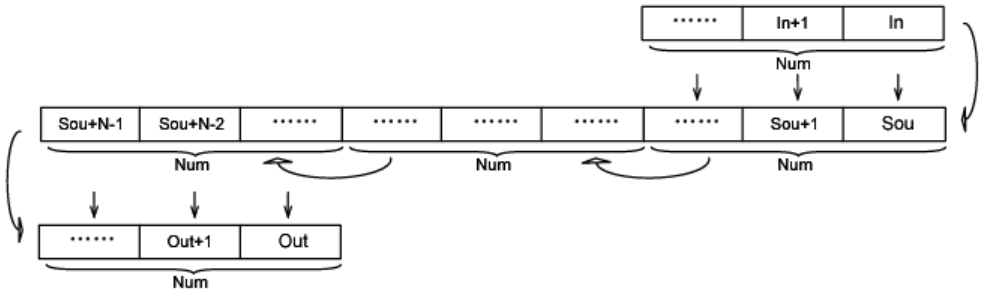
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SHL En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Bit shift start component	√		Occupy Num continuous component
Sou	Source start component	√		Occupy Num continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. According to Num a group,SHL instruction use start from Sou N components left shift Num bit, shift start from In Num components ,shift out start from Out Num components.As follows:



2. If Sou is register component, then use start from Sou N registers left shift bitwise Num bit.

3. 1≤Num≤N, no then instruction not execute.

4. SHL instruction general executed by edge.

[Instruction example]

[Program description]

1. When M0=ON, M100~M105 left shift 3 bit, shift in X0~X2,shift out put Y0~Y2.

Sou component	Initial value	Left shift result

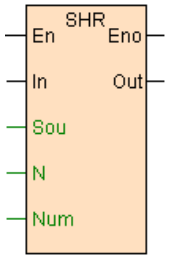
M100	O N	M100=OFF
M101	OFF	M101=ON
M102	ON	M102=ON
M103	ON	M103=ON
M104	ON	M104=OFF
M105	OFF	M105=ON
X0	OFF	
X1	ON	
X2	ON	
Y0		Y0=ON
Y1		Y1=ON
Y2		Y2=OFF

2. When M1=ON,V1000~V1005 bitwise left shift 3 bits, shift in X0~X2,shift out to M200~M202.

Sou c omponent	Initial value	Left shift result
V1000	00000000 00011110	V1000=00000000 11110110
V1001	11111111 01101010	V1001=11111011 01010000
V1002	00000000 00011001	V1002=00000000 11001111
V1003	00000000 00001000	V1003=00000000 01000000
V1004	00000000 01011111	V1004=00000010 11111000
V1005	11111111 11101100	V1005=11111111 01100000
X0	OFF	
X1	ON	
X2	ON	
M200		M200=ON
M201		M201=ON
M202		M202=ON

SHR(Bit right shift)

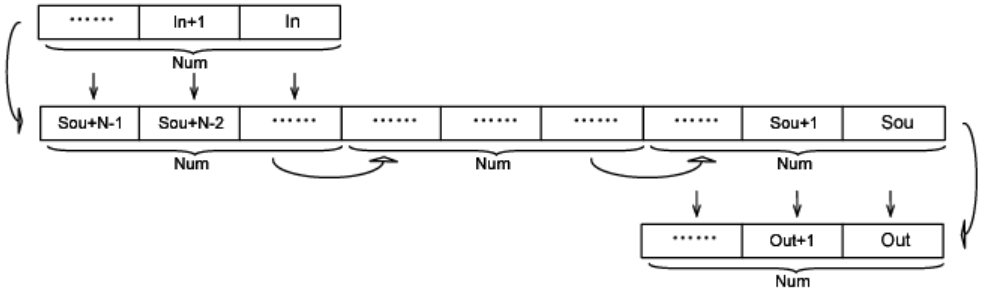
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SHR En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Bit shift start component	√		Occupy Num continuous component
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. According to Num a group ,SHR instruction use start from Sou N components right shift Num bit, shift start from In Num components ,shift out start from Out Num components.As follows:



2. If Sou are register component, then use start from Sou N registers right shift bitwise Num bit .

3. 1≤Num≤N, no then instruction not executed.

4. SHR instruction general executed by edge.

[Instruction example]

[Program description]

1. When M0=ON,M100~M105 right shift 3 bits, shift in X0~X2,shift out to Y0~Y2.

Sou c omponent	Initial value	Right shift result
M100	O N	M100=ON
M101	OFF	M101=ON
M102	OFF	M102=OFF
M103	ON	M103=OFF

M104	ON	M104=ON
M105	OFF	M105=ON
X0	OFF	
X1	ON	
X2	ON	
Y0		Y0=ON
Y1		Y1=OFF
Y2		Y2=OFF

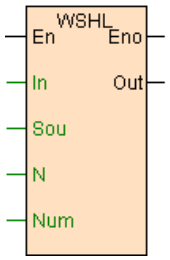
2. When M1=ON,V1000~V1005 right shift 3 bits bitwise, shift in X0~X2,shift out to 在M200~M202.

Sou c omponent	Initial value	Right shift result
V1000	00000000 00011110	V1000=01000000 00000011
V1001	11111111 01101010	V1001=00111111 11101101
V1002	00000000 00011001	V1002=00000000 00000011
V1003	00000000 00001000	V1003=11100000 00000001
V1004	00000000 01011111	V1004=10000000 00001011
V1005	11111111 11101100	V1005=11011111 11111101
X0	OFF	
X1	ON	
X2	ON	
M200		M200=OFF
M201		M201=ON
M202		M202=ON

WSHL(Word left shift)

Instruction format and parameter specification

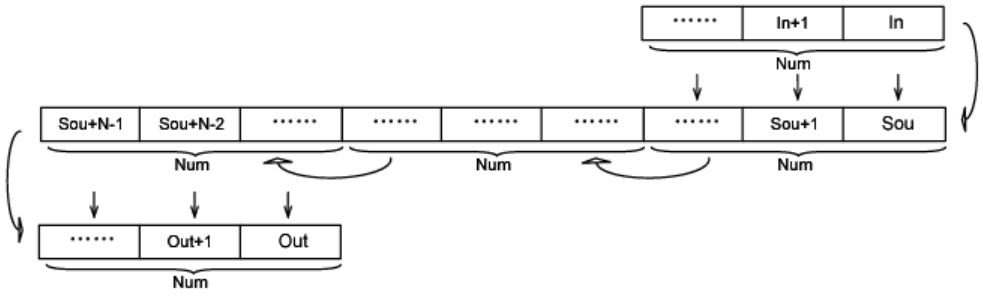
Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			WSHL En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Word shift start component	√		Occupy Num continuous component
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. WSHL instruction use start from Sou N components left shift Num word, shift start from In Num components ,shift out start from Out Num components.As follows:



2. 1≤Num≤N, no then instruction not execute.

3. WSHL instruction general executed by edge.

[Instruction example]

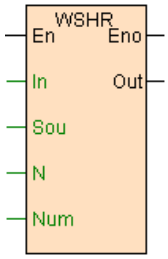
[Program description]

When M0=ON,V1000~V1005 left shift 3 word ,shift in V0~V2,shift out to V100~V102.

Sou c omponent	Initial value	Le ft shift result
V1000	30	V1000=100
V1001	-150	V1001=200
V1002	25	V1002=300
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V0	100	
V1	200	
V2	300	
V100		V100=8
V101		V101=95
V102		V102=-20

WSHR(Word right shift)

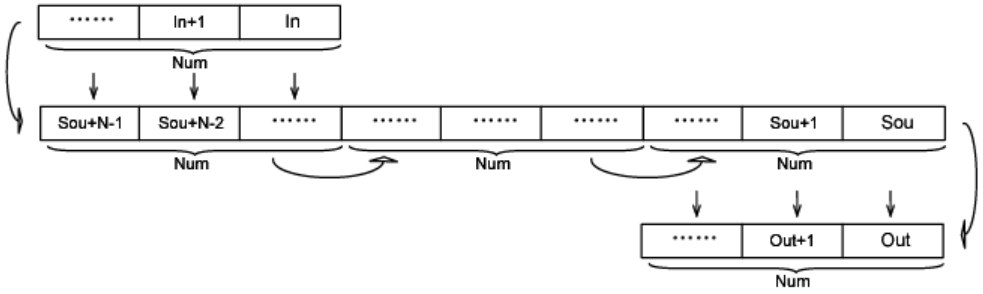
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WSHR En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Word shift start component	√		Occupy Num continuous component
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. WSHR instruction use start from Sou N components right shift Num word, shift start from In Num components ,shift out start from Out Num components.As follows:



2. 1≤Num≤N, no then instruction not execute.

3. WSHR instruction general executed by edge.

[Instruction example]

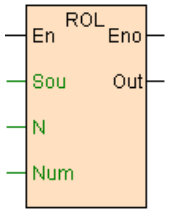
[Program description]

When M0=ON,V1000~V1005 right 3 word, shift in V0~V2, shift out to V100~V102.

Sou c omponent	Initial value	Ri ght shift result
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=100
V1004	95	V1004=200
V1005	-20	V1005=300
V0	100	
V1	200	
V2	300	
V100		V100=30
V101		V101=-150
V102		V102=25

ROL(Bit rotate left shift)

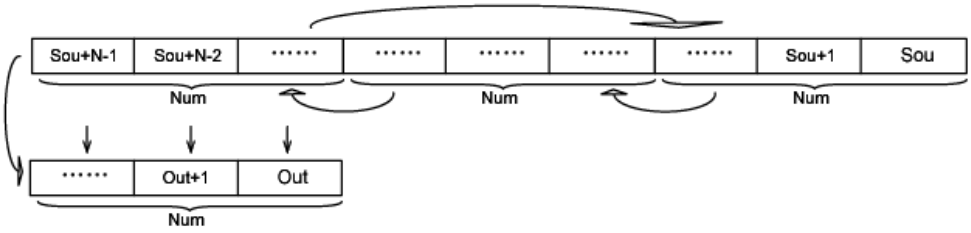
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ROL En, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. According to Num a group ,ROL instruction use start from Sou N components left shift Num bit, shift start from Sou Num components ,shift out start from Out Num components.As follows:



2. If Sou are register component, then use start from Sou N registers left shift bitwise .

3. 1≤Num≤N, no then instruction not execute .

4. ROL instruction general executed by edge.

[Instruction example]

[Program description]

1. When M0=ON, M100~M105 rotate left shift 3 bits, shift out to Y0~Y2.

Sou c omponent	Initial value	Le ft shift result
M100	O N	M100=ON
M101	OFF	M101=ON
M102	ON	M102=OFF
M103	ON	M103=ON
M104	ON	M104=OFF

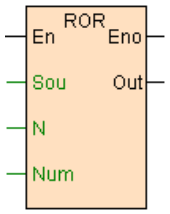
M105	OFF	M105=ON
Y0		Y0=ON
Y1		Y1=ON
Y2		Y2=OFF

2. When M1=ON, V1000~V1005 rotate left shift 3 bits, shift out to M200~M202.

Sou c omponent	Initial value	Left shift result
V1000	00000000 00011110	V1000=00000000 11110111
V1001	11111111 01101010	V1001=11111011 01010000
V1002	00000000 00011001	V1002=00000000 11001111
V1003	00000000 00001000	V1003=00000000 01000000
V1004	00000000 01011111	V1004=00000010 11111000
V1005	11111111 11101100	V1005=11111111 01100000
M200		M200=ON
M201		M201=ON
M202		M202=ON

ROR(Bit rotate right shift)

Instruction format and parameter specification

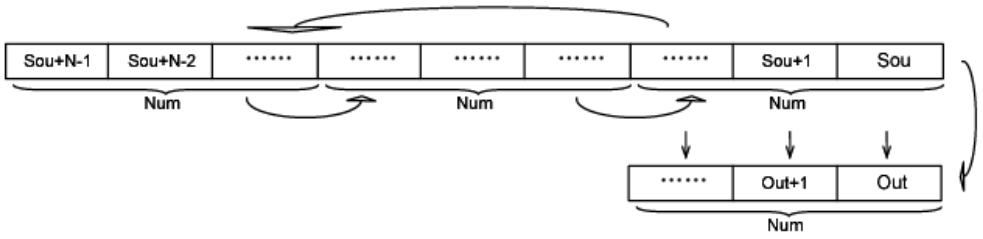
Language	LD	FBD	IL	Program example
Instruction format			ROR En, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component

Parameter	Parameter define	Input	Output	Declare
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. According to Num a group ,ROR instruction use start from Sou N components right shift Num bit, shift start from Sou Num components ,shift out start from Out Num components.As follows:



2. If Sou are register component, then use start from Sou N registers right shift bitwise .

3. 1≤Num≤N, no then instruction not execute.

4. ROR instruction general executed by edge.

[Instruction example]

[Program description]

1. When M0=ON, M100~M105 rotate right shift 3 bits , shift out to Y0~Y2.

Sou c omponent	Initial value	Ri ght shift result
M100	O N	M100=ON
M101	OFF	M101=ON
M102	OFF	M102=OFF
M103	ON	M103=ON
M104	ON	M104=OFF
M105	OFF	M105=OFF
Y0		Y0=ON
Y1		Y1=OFF
Y2		Y2=OFF

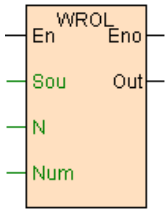
2. When M1=ON, V1000~V1005 rotate right shift 3 bits, shift out to M200~M202.

Sou c omponent	Initial value	Right shift result
V1000	00000000 00011110	V1000=01000000 00000011
V1001	11111111 01101010	V1001=00111111 11101101
V1002	00000000 00011001	V1002=00000000 00000011
V1003	00000000 00001000	V1003=11100000 00000001
V1004	00000000 01011111	V1004=10000000 00001011

V1005	11111111 11101100	V1005=11011111 11111101
M200		M200=OFF
M201		M201=ON
M202		M202=ON

WROL(Word rotate left shift)

Instruction format and parameter specification

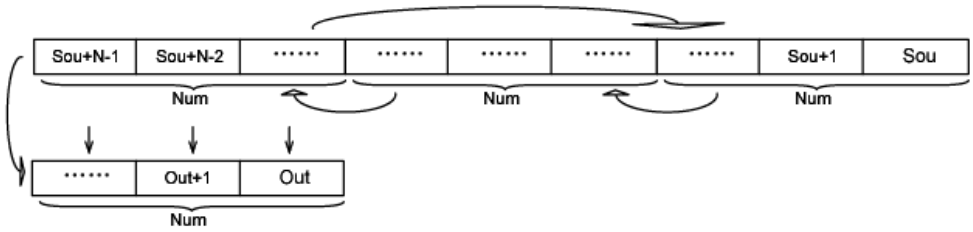
Language	LD	FBD	IL	Program example
Instruction format			WROL En, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. WROL instruction use start from Sou N components left shift Num word, shift start from Sou Num components ,shift out start from Out Num components.As

follows:



2. 1≤Num≤N, no then instruction not execute .

3. WROL instruction general executed by edge.

[Instruction example]

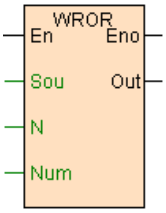
[Program description]

When M0=ON,V1000~V1005 rotate left shift 3 words, shift out to V100~V102.

Sou c omponent	Initial value	Le ft shift result
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V100		V100=8
V101		V101=95
V102		V102=-20

WROR(Word rotate right shift)

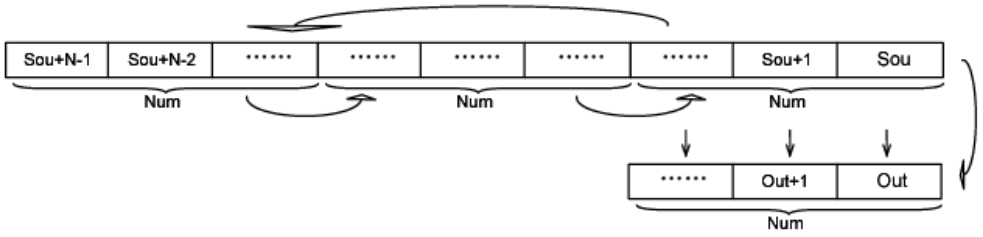
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WROR En, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy Num continuous component

[Instruction function and effect declare]

1. WROR instruction use start from Sou N components right shift Num word, shift start from Sou Num components ,shift out start from Out Num components.As follows:



2. 1≤Num≤N, no then instruction not execute.

3. WROR instruction general executed by edge.

[Instruction example]

[Program description]

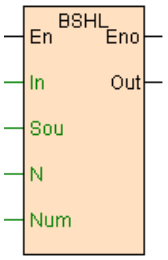
When M0=ON,V1000~V1005 rotate right shift 3 words, shift out to V100~V102.

Sou c omponent	Initial value	Right shift result
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V100		V100=30
V101		V101=-150
V102		V102=25

BSHL(Byte left shift)

Instruction format and parameter specification

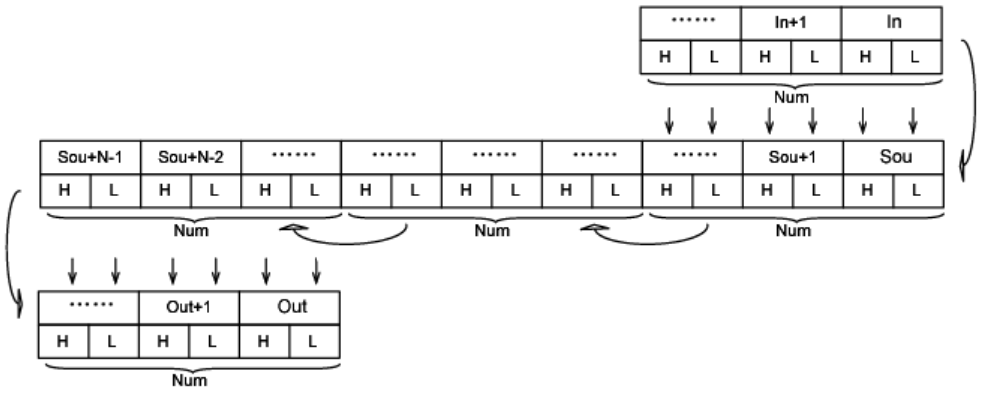
Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			BSHL En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Shift in byte start component	√		Occupy (Num-1)\2+1 continuous component
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy (Num-1)\2+1 continuous component

[Instruction function and effect declare]

1. BSHL instruction use start from Sou N components left shift Num byte, shift in start from In Num bytes, shift out to start from Out Num bytes.As follows:



2. 1≤Num≤2*N, no then instruction not execute.

3. BSHL instruction general executed by edge.

[Instruction example]

[Program description]

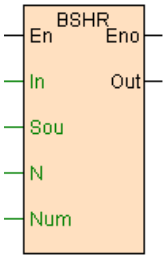
When M0=ON,V1000~V1005 left shift 3 bytes, shift in V0 and V1 low byte, shift out to V100 and V101 low byte .

Sou c omponent	Initial value	Le ft shift result
V1000	30(0x001E)	V1000=100(0x1234)
V1001	-150(0xFF6A)	V1001=200(0x1E78)
V1002	25 (0x0019)	V1002=300 (0x6A00)
V1003	8(0x0008)	V1003=30(0x19FF)
V1004	95 (0x005F)	V1004=-150(0x0800)
V1005	-20(0xFFEC)	V1005=25(0x5F00)
V0	4660 (0x1234)	

V1	22136 (0x5678)	
V100		V100=8(0xEC00)
V101		V101=95(0x00FF)

BSHR(Byte right shift)

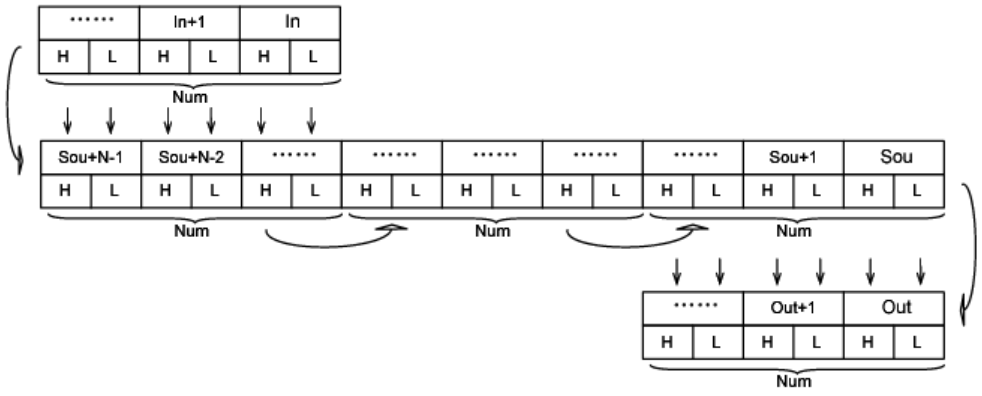
Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BSHR En, In, Sou, N, Num, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Shift in byte start component	√		Occupy (Num-1)\2+1 continuous component
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Num	Shift times	√		
Eno	Enable output		√	
Out	Shift out component		√	Occupy (Num-1)\2+1 continuous component

[Instruction function and effect declare]

1. BSHR instruction use start from Sou N components right shift Num byte, shift in start from In Num bytes, shift out to start from Out Num bytes.As follows:



2. 1≤Num≤2*N, no then instruction not execute.

3. BSHR instruction general executed by edge.

[Instruction example]

[Program description]

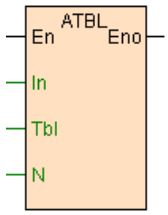
When M0=ON,V1000~V1005 right shift 3 bytes, shift in V0 and V1 low byte, shift out to V100 and V101 low byte.

Sou c omponent	Initial value	Ri ght shift result
V1000	30(0x001E)	V1000=100(0x19FF)
V1001	-150(0xFF6A)	V1001=200(0x0800)
V1002	25 (0x0019)	V1002=300 (0x5F00)
V1003	8(0x0008)	V1003=30(0xEC00)
V1004	95 (0x005F)	V1004=-150(0x34FF)

V1005	-20(0xFFEC)	V1005=25(0x7812)
V0	4660 (0x1234)	
V1	22136 (0x5678)	
V100		V100=8(0x001E)
V101		V101=95(0x006A)

ATBL(Append to array)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ATBL En, In, Tbl, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input data	√		
Tbl	Array start component	√		Occupy N+1 continuous component
N	Array length	√		1~256
Eno	Enable output		√	

[Instruction function and effect declare]

1. Append to array instruction ATBL will append bit state or register value to the array specified by Tbl in sequence.

2. En is instruction Enable item, general use edge type (rising edge or faillling edge) signal.When En state hop ON once , append In value to array specified by Tbl, array element number add 1(Tbl register content value add 1).

3. Tbl define store data array start component ,N is array length , among the first register (Tbl)of the array be the numbers of the array element ,Tbl+1 to Tbl+N total N registers use for store array data.so, if In is bit component ,the maximum array element can be stored are N*16; If In is regieter ,the maximum array element can be stored are N. When number of array element exceed the maximum value of array elements , data con not append into the array .

4. Store queue of the array as follows :

A. Bit component data: if append to array is bit, store queue of the array as follows:

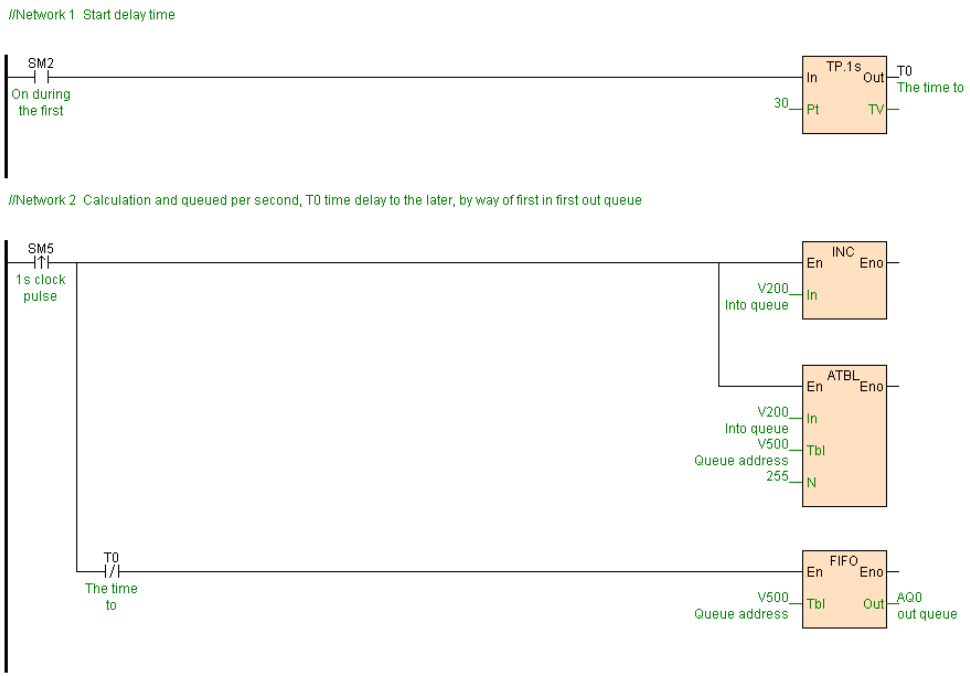
Array component		**Array content**
Tbl		Number of array element
Tbl+1	b0	First array element
	b1	Second array element
	.....	
	b15	Sixteen array element

Tbl+2	b0	seventeen array element
	b1	eighteen array element
	.....	.....
.....	b0	.....
	.....	.....

B. Register component data: if appended to array is 16 bit register, store queue of the array as follows:

Array component	**Array content**
Tbl	Number of array element
Tbl+1	First array element
Tbl+2	Second array element
Tbl+3	Third array element
.....	.....

[Instruction example]

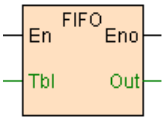


[Program description]

1. First scan SM2=ON, start T0 timer(30s).
2. SM5 is clock pulse per second , cycles per second, INC instruction V200 add 1(for simulated data),ATBL instruction append V200 into the array start from V500 , array length is 255.
3. After 30s ,T0=OFF, per second FIFO instruction will be first in first out get data from array, output to AQ0.

FIFO(First in first out)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FIFO En, Tbl, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Tbl	Array start component	√		
Eno	Enable output		√	
Out	Data output		√	

[Instruction function and effect declare]

1. FIFO instruction according to first in first out model get out data from array ATBL , each data be get out array element subtract 1.

2. If number of array elements≤0, then instruction not execute.

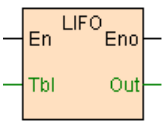
3. FIFO instruction cooperate ATBL instruction ready-made first in first out array,FIFO instruction general executed by edge.

[Instruction example]

Refer to ATBL instruction.

LIFO(Last in first out)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			LIFO En, Tbl, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Tbl	Array start component	√		
Eno	Enable output		√	
Out	Data output		√	

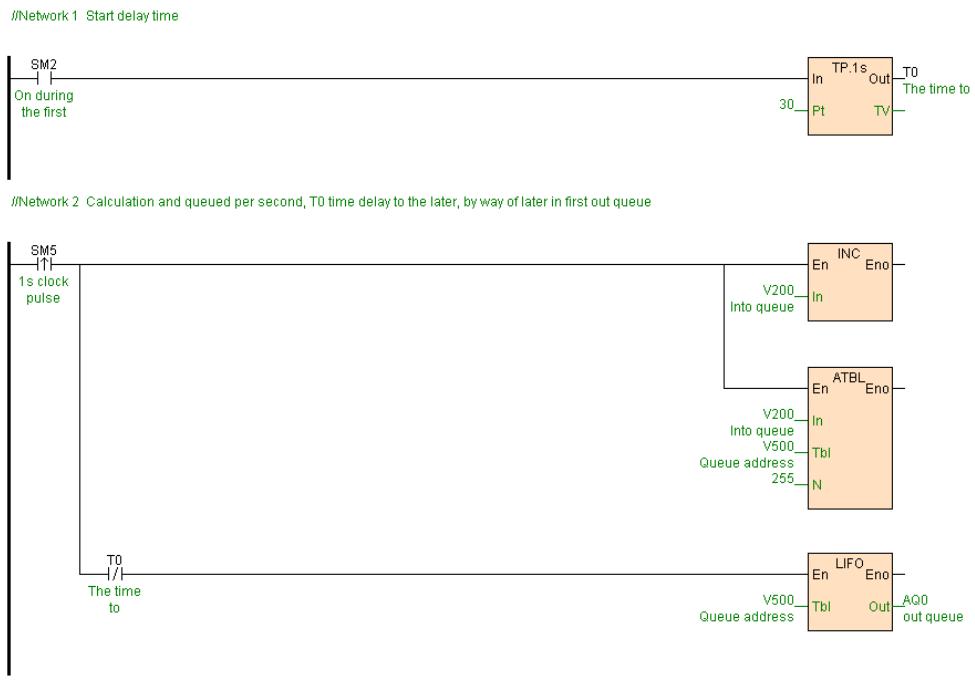
[Instruction function and effect declare]

1. LIFO instruction according to last in first out model get out data from array ATBL , each data be get out array element subtract 1.

2. If number of array elements≤0, then instruction not execute.

3. LIFO instruction cooperate ATBL instruction ready-made last in first out array(stack),LIFO instruction general executed by edge.

[Instruction example]



[Program description]

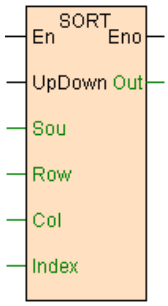
1. First scan SM2=ON, start T0 timer(30s).

2. SM5 clock pulse per second , cycles per second, INC instruction V200 add 1(for simulated data), ATBL instruction append V200 into the array start from V500 , array length is 255.

3. After 30s,T0=OFF, per second LIFO instruction get out data from array according to last in first out ,output to AQ0.

SORT(Data sort)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SORT En, UpDown, Sou, Row, Col, Index, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
UpDown	Ascending or descending order control	√		
Sou	Source start component	√		Occupy Row * Col continuous component
Row	Row	√		1~64

Parameter	Parameter define	Input	Output	Declare
Col	Line	√		1~64
Index	Arrange sequence	√		
Eno	Enable output		√	
Out	Output		√	Occupy Row * Col continuous component

[Instruction function and effect declare]

1. The data from start Sou Row row Col line total Row×Col elements will be sorted ,sort refer to Index specified line ,sort direction controlled by UpDown, UpDown is OFF then ascending sort , UpDown is ON then descending sort, data be sorted store into the first Out total Row×Col elements.

2. SORT instruction be executed edge ,if modified the Sou data after sorted, then must be retrigger.

3. Must meet Row≥1. Col≥1. Index≤Col, no then instruction not execute.

[Instruction example]

[Program description]

If initial data (performance) as follows:

Name	Chinese	Math	English
Wu	V1000=98	V1001=65	V1002=81
Chen	V1003=78	V1004=89	V1005=65
Wang	V1006=87	V1007=99	V1008=68

Li	V1009=60	V1010=92	V1011=83
Zhang	V1012=72	V1013=90	V1014=56

If M100=OFF, when M0=ON, start from V1000 5 row 3 line array ,according to line 2(mathematic performance) use ascending sort , result store into start from V0 5x3 elements.

Name	Chinese	Math	English
Wu	V0=98	V1=65	V2=81
Chen	V3=78	V4=89	V5=65
Zhang	V6=72	V7=90	V8=56
Li	V9=60	V10=92	V11=83
Wang	V12=87	V13=99	V14=68

If M100=ON, when M0=ON,start from V1000 5 row 3 line array ,according to line 2(mathematic performance) use descending sort , result store into start from V0 5x3 elements.

Name	Chinese	Math	English
Wang	V0=87	V1=99	V2=68
Li	V3=60	V4=92	V5=83
Zhang	V6=72	V7=90	V8=56
Chen	V9=78	V10=89	V11=65
Wu	V12=98	V13=65	V14=81

Data conversion instruction

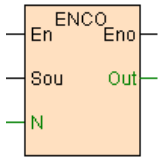
Data conversion instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
ENCO			Encoder	√	√	√
DECO			Decoder	√	√	√
BTOW			Bit convert to word	√	√	√
WTOB			Word convert to bit	√	√	√
HEX	HEX.LB		ASCII convert to hexadecimal	√	√	√
ASCI	ASCI.LB		Hexadecimal convert to ASCII	√	√	√
BUNB			Discrete bit combination to continuous bit	√	√	√
BUNW			Discrete bit combination to continuous word	√	√	√
WUNW			Discrete word combination to continuous word	√	√	√
BDIB			Continuous bit disperse to discrete bit	√	√	√
WDIB			Continuous word disperse to discrete bit	√	√	√
WDIW			Continuous word disperse to discrete word	√	√	√
BCD		D.BCD	BIN convert to BCD	√	√	√
BIN		D.BIN	BCD convert to BIN	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
ITOL			Integer convert to long integer	√	√	√
GRAY			BIN convert to GRAY code	√	√	√
GBIN			GRAY code convert to BIN	√	√	√

ENCO(Encoder)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ENCO En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
N	Number of bits be encoded	√		0~8
Eno	Enable output		√	
Out	Encode output		√	

[Instruction function and effect declare]

1. ENCO instruction use for obtain the position of the maximum ON bit in Sou data .
2. 0≤N≤8,maximum may encode 2^8=256 bits.

3.

If the source data have many bits are 1(ON), then only deal with the highest bit

.

[Instruction example]

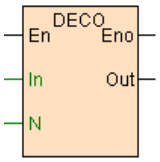
[Program description]

1. M0=ON,X0~X7(2^3=8) proceed encode, if X6=ON,X7=OFF other not to matter, then V0=7.

2. M0=ON,V100 low 8 bits (2^3=8,high 8 bits not use) proceed encode ,if V100=2345(00001001 00101001),then V10=6.

DECO(Decoder)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			DECO En, In, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In	Decode input	√		
N	Decode digitals	√		0~8
Eno	Enable output		√	
Out	Decode o utput		√	

[Instruction function and effect declare]

1. DECO instruction decode N digitals of the In data, result output to Out.

2. 0≤N≤8,maximum may decode output 2^8=256 bits.

[Instruction example]

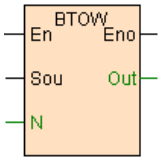
[Program description]

M0=ON,V0 decoded Output to Y0~Y7(2^3=8) and low 8 bits of V100 (2^3=8,high 8 bits total are 0). If V0=7, then among Y0~Y7 only Y6=ON others OFF,V100=64(00000000 01000000).

BTOW(Bit convert to word)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			BTOW En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component
N	Convert digitals	√		1~256
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

BTOW instruction convert N bit components start from Sou, convert to integer result output toOut.

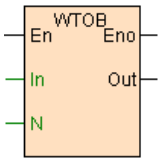
[Instruction example]

[Program description]

When M0=ON,X0~X5 convert to integer ,if X1=ON. X2=ON. X5=ON others are OFF,then V0=38(00000000 00100110).

WTOB (Word convert to bit)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WTOB En, In, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
N	Convert digitals	√		1~256
Eno	Enable output		√	
Out	Convert result start c omponent		√	Occupy N continuous component

[Instruction function and effect declare]

WTOB instruction convert N bits of In to output to Out.

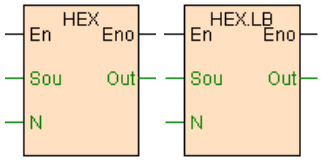
[Instruction example]

[Program description]

When M0=ON,low 7 bits of V0 convert to Y0~Y6,if V0=38(00000000 00100110),then Y1=ON. Y2=ON. Y5=ON others are OFF.

HEX. HEX.LB(ASCII convert to hexadecimal)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			HEX En, Sou, N, Out HEX.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		HEX occupy (N-1)\2+1 continuous component,HEX.LB occupy N continuous component
N	Number of characters converted	√		1~256
Eno	Enable output		√	
Out	Output		√	Occupy (N-1)\4+1 continuous component

[Instruction function and effect declare]

1. HEX instruction convert start from Sou ASCII code to HEX value, number of N characters will be onverted.

2. 8 bit model instruction HEX.LB only convert low byte of Sou , high byte not use .

3. ASCII code characters only be 0~9 and A. B. C. D. E. F these 6 characters, if there have illegality characters in Sou then instruction not execute.

[Instruction example]

[Program description]

If initial data (ASCII code data) as follows:

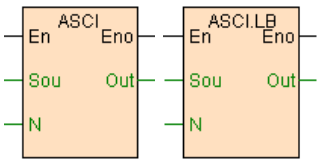
Component	Register value (ASCII code)	High low byte value (ASCII code)	ASCII character
V1000	0x3938	Low byte 0x38	"8"
		Hight byte 0x39	"9"
V1001	0x4241	Low byte 0x41	"A"
		Hight byte 0x42	"B"
V1002	0x3534	Low byte 0x34	"4"
		Hight byte 0x35	"5"
V1003	0x3332	Low byte 0x32	"2"
		Hight byte 0x33	"3"
V1004	0x4645	Low byte 0x45	"E"
		Hight byte 0x46	"F"
V1005	0x3039	Low byte 0x39	"9"
		Hight byte 0x30	"0"
V1006	0x3831	Low byte 0x31	"1"
		Hight byte 0x38	"8"
V1007	0x4443	Low byte 0x43	"C"
		Hight byte 0x44	"D"

When M0=ON, start from V1000 ASCII code convert to data ,HEX convert result to components start form V0 ,HEX.LB only convert to low byte ASCII code of V1000 , convert result to start from V10, N=1~8 converted result as follows.

N	HEX		HEX.LB	
	V1	V0	V11	V10
1		0x8		0x8
2		0x89		0x8A
3		0x89A		0x8A4
4		0x89AB		0x8A42
5	0x8	0x9AB4	0x8	0xA42E
6	0x89	0xAB45	0x8A	0x42E9
7	0x89A	0xB452	0x8A4	0x2E91
8	0x89AB	0x4523	0x8A42	0xE91C

ASCI. ASCI.LB(Hexadecimal convert to ASCII)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			ASCI En, Sou, N, Out ASCI.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy (N-1)\4+1 continuous component
N	number of character be converted	√		1~256
Eno	Enable output		√	
Out	Output		√	ASCI occupy (N-1)\2+1 continuous component,ASCI.LB occupy N continuous component

[Instruction function and effect declare]

1. ASCI instruction convert start from Sou value to ASCII code character , number of N characters will be converted, convert result stored to start from Out component .

2. 8 bit model instruction ASCI.LB only store convert low byte to low byte of Out ,hight byte is 0.

[Instruction example]

[Program description]

If initial data (convert to ASCII code)as follows:

Component	Register value
V1100	0x4523
V1101	0x89AB

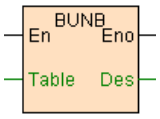
When M1=ON,ASCI instruction convert the value of start from V1100 to ASCII code, result to start from V100 component,ASCI.LB instruction put the result to start from V2000 components (Only low byte, hight byte is0),N=1~8 convert ed result as follows

.

Instruction format	Out component	N	1	2	3	4	5	6	7	8
ASCI	V100	Low byte	"3"	"2"	"5"	"4"	"B"	"A"	"9"	"8"
		Hight byte		"3"	"2"	"5"	"4"	"B"	"A"	"9"
	V101	Low byte			"3"	"2"	"5"	"4"	"B"	"A"
		Hight byte				"3"	"2"	"5"	"4"	"B"
	V102	Low byte					"3"	"2"	"5"	"4"
		Hight byte						"3"	"2"	"5"
	V103	Low byte							"3"	"2"
		Hight byte								"3"
ASCI.LB	V200	Low byte	"3"	"2"	"5"	"4"	"B"	"A"	"9"	"8"
	V201	Low byte		"3"	"2"	"5"	"4"	"B"	"A"	"9"
	V202	Low byte			"3"	"2"	"5"	"4"	"B"	"A"
	V203	Low byte				"3"	"2"	"5"	"4"	"B"
	V204	Low byte					"3"	"2"	"5"	"4"
	V205	Low byte						"3"	"2"	"5"
	V206	Low byte							"3"	"2"
	V207	Low byte								"3"

BUNB(Discrete bit combination to continuous bit)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BUNB En, Table, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Table	Discrete bit table	√		
Eno	Enable output		√	
Des	Target start component		√	

[Instruction function and effect declare]

BUNB instruction use for combination the discrete bit which Table define to continuous bit components .

Discrete bit table: may be called by BUNB. BUNW. BDIB. WDIB instruction Table item.

how to define the discrete bit table please refer to "instruction use table" section.

[Instruction example]

[Program description]

If "read discrete bit table" defined as follows:

Sequence number	Bit

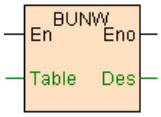
	component
1	X3
2	M10
3	M301
4	S21
5	M77
6	M100
7	X1
8	Y6

When M0=ON,BUNB instruction combine the "read discrete bit table" defined bit to start from M500 continuous component.

Sequence number	Executed result
1	M500 = X3
2	M501 = M10
3	M502 = M301
4	M503 = S21
5	M504 = M77
6	M505 = M100
7	M506 = X1
8	M507 = Y6

BUNW(Discrete bit combination to continuous word)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BUNW En, Table, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Table	Discrete bit table	√		
Eno	Enable output		√	

Parameter	Parameter define	Input	Output	Declare
Des	Target start component		√	

[Instruction function and effect declare]

BUNW instruction use for combination the discrete bit which Table define to continuous word components.

Discrete bit table:May be called by BUNB. BUNW. BDIB. WDIB instruction Table item.

How to define the discrete bit table please refer to " instruction use table" section.

[Instruction example]

[Program description]

If "read discrete bit table" defineas follows:

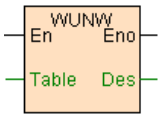
Sequence number	bit component
1	X3
2	M10
3	M301
4	S21
5	M77
6	M100
7	X1
8	Y6

When M1=ON,BUNW instruction combination bitwise "read discrete bit table" defined bit to start from V0 registers.

Sequence number	Executed result
1	V0.b0 = X3
2	V0.b1 = M10
3	V0.b2 = M301
4	V0.b3 = S21
5	V0.b4 = M77
6	V0.b5 = M100
7	V0.b6 = X1
8	V0.b7 = Y6

WUNW(Discrete word combination to continuous word)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WUNW En, Table, Des	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Table	Discrete register component table	√		
Eno	Enable output		√	
Des	Target start component		√	

[Instruction function and effect declare]

WUNW instruction use for combination the discrete register which Table define to continuous register components .

Discrete register component table:May be called by WUNW. WDIW instruction Table item. How to define discrete register component table please refer to " instruction use table " section.

[Instruction example]

[Program description]

If "read discrete register table " defineas follows:

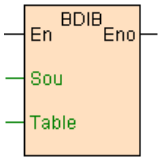
Sequence number	bit component
1	AI1
2	V10
3	V106
4	AQ0

When M0=ON, WUNW instruction combination move the "read discrete register table " defined register to start from V200 continuous component.

Sequence number	Executed result
1	V200 = AI1
2	V201 = V10
3	V202 = V106
4	V203 = AQ0

BDIB(Continuous bit disperse to discrete bit)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			BDIB En, Sou, Table	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
Table	Discrete bit component table	√		
Eno	Enable output		√	

[Instruction function and effect declare]

BDIB instruction use for disperse the bit components start from Sou to discrete bit components defined by Table.

Discrete bit table: may be called by BUNB. BUNW. BDIB. WDIB instruction Table item.

How to define the discrete bit table please refer to " instruction use table" section.

[Instruction example]

[Program description]

If "write discrete bit table " define as follows:

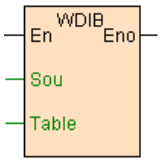
Sequence number	Bit component
1	Y4
2	M10
3	M301
4	S21
5	M77
6	M100
7	Y0
8	Y6

When M0=ON,BDIB instruction combination move the start from M500 continuous bit to "write discrete bit table " .

Sequence number	Executed result
1	Y4 = M500
2	M10 = M501
3	M301 = M502
4	S21= M503
5	M77= M504
6	M100 = M505
7	Y0 = M506
8	Y6 = M507

WDIB(Continuous word disperse to discrete bit)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WDIB En, Sou, Table	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
Table	Discrete bit component table	√		
Eno	Enable output		√	

[Instruction function and effect declare]

WDIB instruction use for disperse the bit components start bitwise from Sou register to discrete bit components defined by Table.

Discrete bit table: may be called by BUNB. BUNW. BDIB. WDIB instruction Table item.

How to define the discrete bit table please refer to " instruction use table" section.

[Instruction example]

[Program description]

If "write discrete bit table " define as follows:

Sequence number	Bit component
1	Y4
2	M10

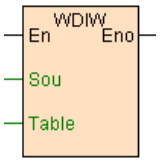
3	M301
4	S21
5	M77
6	M100
7	Y0
8	Y6

When M1=ON, WDIB instruction discrete bitwise the start from V0 register to "write discrete bit table" defined bit components .

Sequence number	Executed result
1	Y4 = V0.b0
2	M10 = V0.b1
3	M301 = V0.b2
4	S21= V0.b3
5	M77= V0.b4
6	M100 = V0.b5
7	Y0 = V0.b6
8	Y6 = V0.b7

WDIW(Continuous word disperse to discrete word)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WDIW En, Sou, Table	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
Table	Discrete register component table	√		
Eno	Enable output		√	

[Instruction function and effect declare]

WDIW instruction use for disperse the registers start from Sou register to discrete registers defined by Table.

Discrete register component table: may be called by WUNW. WDIW instruction Table item. How to defined discrete register component table please refer to " instruction use table" section.

[Instruction example]

[Program description]

If " write discrete register table" define as follows:

Sequence number	Bit component
1	V90
2	V10
3	V106
4	AQ0

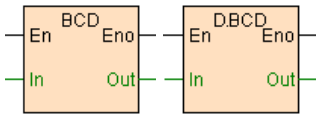
When M0=ON, WDIW instruction discrete the register start from V200 to "read discrete register table " defined discrete register.

Sequence number	Executed result
1	V90 = V200
2	V10 = V201
3	V106 = V202
4	AQ0 = V203

BCD. D.BCD(BIN convert to BCD)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16,32 bit Instruction format			BCD En, In, Out D.BCD En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

1. BCD instruction (D.BCD is 32 bit instruction) use for convert the value to BCD code.

2. 16 bit instruction value input range is 0~9999,32 bit instruction value input range is 0~99999999,In exceed range then instruction not execute.

[Instruction example]

[Program description]

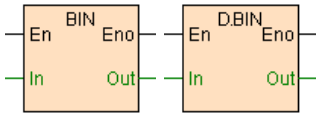
When M0=ON,BCD instruction convert V1000 value to BCD code result to V0,D.BCD instruction convert V1001V1002 value to result to V10V11,as follows table.

In component	Initial value	BCD convert result	D.BCD convert result
V1000	2345	V0 = 0x2345	
V1001V1002	48702861		V10V11 =

			0x48702861

BIN. D.BIN(BCD convert to BIN)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			BIN En, In, Out D.BIN En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

1. BIN instruction (D.BIN is 32 bit instruction) use for convert BCD code to value.
2. BCD code (Binary-coded Decimal) also name binary code decimal numbers. binary-decimal code or 8421 code, that is expand the decimal number to binary number according to 8421 model .BCD code is four digit binary code, that is convert decimal number to binary number , but different from the general convert , each decimal number 0-9 corresponding a four digit binary code,corresponding relation as follows: decimal 0 corresponding 0000; decimal 1 corresponding 0001 ..... decimal 9 corresponding 1001, following 10 express in above-mentioned 2 code, decimal 10 express is 00010000, that is BCD code come across 1001 generate carry bit , unlike general binary code reach 1111 general carry bit 10000.

3. If In input contain not BCD code (contain 0xA. 0xB. 0xC. 0xD. 0xE. 0xF) then instruction not execute.

[Instruction example]

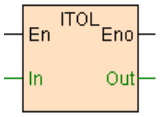
[Program description]

When M1=ON,BIN instruction convert the BCD code of V1100 to value result to V20,D.BIN instruction convert the BCD code of V1101V1102 to value result to V30V31,as follows table .

In component	Initial value	BIN convert result	D.BIN convert result
V1100	0x3938	V20 = 3938	
V1101V1102	0x35344241		V30V31 = 35344241

ITOL(Integer convert to long integer)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ITOL En, In, Out	Download

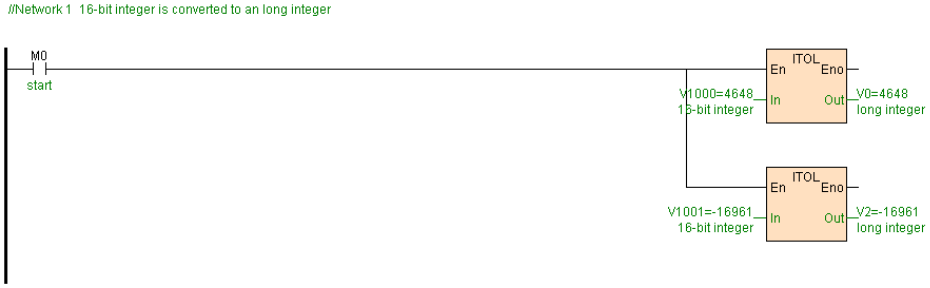
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Eno	Enable output		√	
Out	Output		√	Occupy 2 continuous component

[Instruction function and effect declare]

ITOL instruction use for convert 16 bit integer to 32 bit long integer .

[Instruction example]



[Program description]

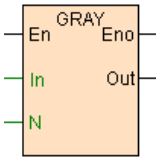
When M0=ON,ITOL instruction convert V1000 to long integer result to V0V1, convert V1001 to long integer result to V2V3,as follows table .

In component	Initial value	convert result
V1000	4648	V0V1 = 4648
V1001	-16961	V2V3 = -16961

GRAY(BIN convert to GRAY code)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			GRAY En, In, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		In is bit component occupy N continuous component, is register component occupy (N-1)\16+1 continuous component
N	GRAY code length	√		1-32
Eno	Enable output		√	
Out	Output		√	Out is bit component occupy N continuous component, is register component occupy (N-1)\16+1 continuous component

[Instruction function and effect declare]

1. GRAY instruction use for convert value to gray code.

2. In must ＞0 then instruction not execute.

[Instruction example]

[Program description]

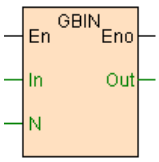
When M0=ON, convert the low 7 bits of V0 to gray code output to Y0~Y6,convert M10~M16 to gray code output to V10.

In component	Value	Convert to gray code result

V0	25	Y0=ON
		Y1=OFF
		Y2=ON
		Y3=OFF
		Y4=ON
		Y5=OFF
		Y6=OFF
M10	O N	V10=21
M11	OFF	
M12	OFF	
M13	ON	
M14	ON	
M15	OFF	
M16	OFF	

GBIN(GRAY code convert to BIN)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			GBIN En, In, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		In is bit component occupy N continuous component, is register component occupy (N-1)\16+1 continuous component
N	GRAY code length	√		1-32
Eno	Enable output		√	

Parameter	Parameter define	Input	Output	Declare
Out	Output		√	Out is bit component occupy N continuous component, is register component occupy (N-1)\16+1 continuous component

[Instruction function and effect declare]

1. GBIN instruction use for convert gray code to value.

2. In must ＞0 then instruction not execute.

[Instruction example]

[Program description]

When M1=ON, convert Y0~Y6 gray code to value output to V20, convert the gray code of the low 7 bits of V10 to value output to M100~M106.

In component	Value	Gray code convert to value
Y0	ON	V20=25
Y1	OFF	
Y2	ON	
Y3	OFF	
Y4	ON	
Y5	OFF	
Y6	OFF	
V10	21	M100=ON
		M101=OFF
		M102=OFF
		M103=ON
		M104=ON
		M105=OFF
		M106=OFF

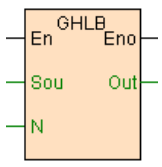
Character instruction

Character instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
GHLB			Obtain high low byte	√	√	√
GETB			Capture byte string	√	√	√
BCMP	BCMP.LB		Byte string comparison	√	√	√
ITOC		D.ITOC	Integer convert to character	√	√	√
CTOI			Character convert to integer	√	√	√
FTOC			Floating point convert to character	√	√	√
CTOF			Character convert to floating point	√	√	√

GHLB(Obtain high low byte)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			GHLB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy N continuous component
N	Number of component	√		1~256
Eno	Enable output		√	
Out	Output		√	Occupy 2N continuous component

[Instruction function and effect declare]

GHLB instruction separate the high low byte of start from Sou N registers low byte output to Out.

[Instruction example]

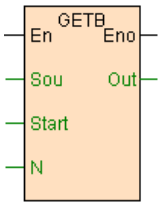
[Program description]

When M0=ON, separate high low byte of V1000~V1004 output to V0~V9.

Sou c omponent	Value	Output component	Output result
V1000	0x001E	V0	0x1E
		V1	0x00
V1001	0xFF6A	V2	0x6A
		V3	0xFF
V1002	0x0E19	V4	0x19
		V5	0x0E
V1003	0x1208	V6	0x08
		V7	0x12
V1004	0x0D5F	V8	0x5F
		V9	0x0D

GETB(Capture byte string)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			GETB En, Sou, Start, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy (Start+N)\2 continuous component
Start	The start byte sequence number	√		
N	Number of bytes	√		1~256
Eno	Enable output		√	
Out	Output		√	Occupy (N+1)\2 continuous component

[Instruction function and effect declare]

GETB instruction capture N bytes from start byte of the byte string start from Sou.

[Instruction example]

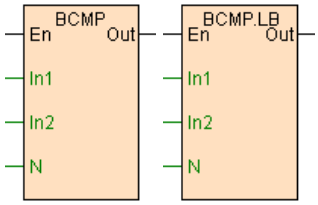
[Program description]

When M0=ON, capture 7 bytes from second byte of the byte string start from V1000 ,output to V0~V3.

Sou c omponent	Value	Output component	Output result
V1000	0x011E	V0	0x6A01
V1001	0xFF6A	V1	0x19FF
V1002	0x0E19	V2	0x080E
V1003	0x1208	V3	0x0012
V1004	0x0D5F		

BCMP. BCMP.LB(Byte string comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			BCMP En, In1, In2, N, Out BCMP.LB En, In1, In2, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Compare byte string 1	√		BCMP occupy (N+1)\2 continuous component, BCMP.LB occupy N continuous component
In2	Compare byte string 2	√		BCMP occupy (N+1)\2 continuous component, BCMP.LB occupy N continuous component
N	Compare number of bytes	√		1~256
Out	Compare result output		√	

[Instruction function and effect declare]

BCMP instruction compare the byte string of In1 and In2,compare n bytes, if equal to then Out=ON, no then Out=OFF. BCMP.LB is low byte model, only compare low byte

part.

[Instruction example]

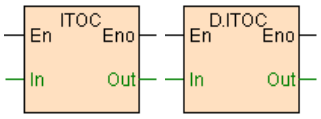
[Program description]

When M0=ON, compare byte string V1000~V1004 and byte string V1010~V1014 ,BCMP instruction compare 7 bytes,BCMP.LB instruction compare 5 low byte.

In1 component	Initial value	In2 component	Initial value	Compare result
V1000	0x011E	V1010	0x011E	M100=OFF M101=ON
V1001	0xFF6A	V1011	0x026A	
V1002	0x0E19	V1012	0x0019	
V1003	0x1208	V1013	0x1208	
V1004	0x0D5F	V1014	0x5D5F	

ITOC. D.ITOC(Integer convert to character)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ITOC En, In, Out D.ITOC En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	Occupy 6 continuous component

[Instruction function and effect declare]

ITOC instruction use for integer convert to character ,D.ITOC use for long integer convert to character. Output automatic occupy 6 continuous register , total can express12 characters, if convert result not enough 12 characters then the behind register filled by the blank space.

[Instruction example]

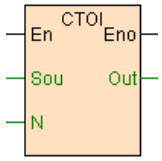
[Program description]

When M0=ON, ITOC convert V1000 integer to character output to V0~V5,D.ITOC convert V1001V1002 long integer to character output to V10~V15,as follows table .

In component	Initial value	convert result
V1000	286	V0~V5 ="286"
V1001V1002	-2584810	V10~V15 =" -2584810"

CTOI(Character convert to integer)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			CTOI En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy 6 continuous component
N	Convert character number	√		N range :1~11
Eno	Enable output		√	
Out	Output		√	Occupy 2 continuous component

[Instruction function and effect declare]

1. CTOI use for convert N character start Sou to long integer , if convert result exceed long integer range then not convert moreover Eno is 0(OFF),Out maintain original not changed .

2. N is will be converted character number, valid range 1~11,if exceed range then not convert moreover Eno=OFF, Out maintain original not changed .

3. If will be converted character contain illegal character(except 0 ~ 9. +. - character), replace space to ahead , lop back .For example: character '123'. '123dfg'. 'A123' convert result all re integer 123.

[Instruction example]

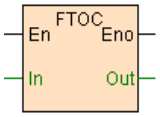
[Program description]

When M0=ON, CTOI convert character 7 characters of V1000~V1005 to integer output to V0V1, convert 9 characters of V1010~V1015 to integer output to V2V3,as follows table .

In component	Initial value	convert result
V1000~V1005	"1234567890"	V0V1 =1234567
V1010~V1015	"-987654321"	V2V3 =-98765432

FTOC(Floating point convert to character)

Instruction format and parameter specification

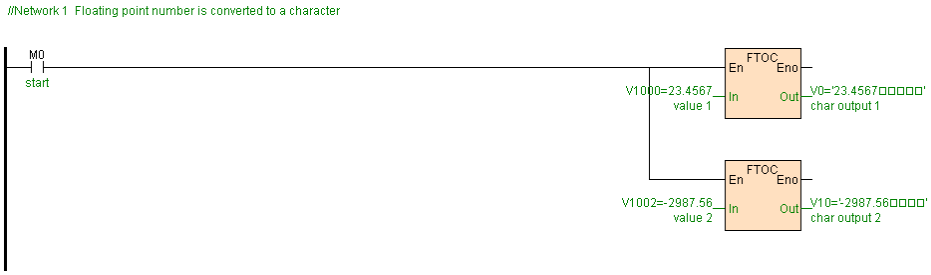
Language	LD	FBD	IL	Program example
Instruction format			FTOC En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		Occupy 2 continuous component
Eno	Enable output		√	
Out	Output		√	Occupy 6 continuous component

[Instruction function and effect declare]

FTOC instruction use for floating point convert to character.Output automatic occupy 6 continuous register ,total can express12 character, if convert result not enough 12 characters then the behind register filled by the blank space.

[Instruction example]



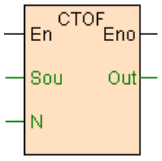
[Program description]

When M0=ON,FTOC use for convert floating point V1000V1001 to character output to V0~V5, convert floating point V1002V1003 to character output to V10~V15,as follows table .

In component	Initial value	convert result
V1000V1001	23.4567	V0~V5 ="23.4567"
V1002V1003	-2987.56	V10~V15 =" -2987.56"

CTOF(Character convert to floating point)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			CTOF En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		Occupy 6 continuous component
N	Convert character number	√		N range :1~11
Eno	Enable output		√	
Out	Output		√	Occupy 2 continuous component

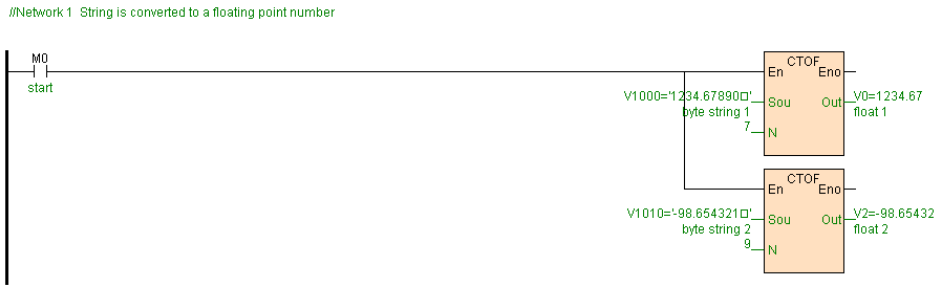
[Instruction function and effect declare]

1. CTOF convert N characters start from Sou to floating point, if convert result exceed floating point range then not convert moreover Eno is 0(OFF),Out maintain original not changed .

2. N is number of character be converted, valid range 1~11, if exceed range then not convert moreover Eno=OFF,Out maintain original not changed .

3. If the character be converted contain illegal character(except 0 ~ 9. .. +. - character), eplace space to ahead , lop back .For example: character '1.23'. '1.23dfg'. 'A1.23' convert result all are floating point 1.23.

[Instruction example]



[Program description]

When M0=ON,CTOF convert 7 characters of V1000~V1005 to floating point output to V0V1, convert 9 characters of V1010~V1015 to floating point output to V2V3,as follows table .

In component	Initial value	convert result
V1000~V1005	"1234.67890"	V0V1 =1234.67
V1010~V1015	"-98.654321"	V2V3 =-98.65432

Arithmetical instruction

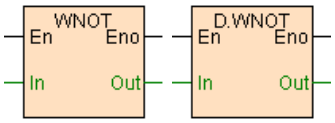
Arithmetical operation instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
WNOT		D.WNOT	Negation	√	√	√
WAND		D.WAND	AND operation	√	√	√
WOR		D.WOR	OR operation	√	√	√
WXOR		D.WXOR	XOR operation	√	√	√
ADD		D.ADD	Addition	√	√	√
SUB		D.SUB	Subtraction	√	√	√
INC		D.INC	Increase 1	√	√	√
DEC		D.DEC	Decrease 1	√	√	√
MUL		D.MUL	Multiplication	√	√	√
DIV		D.DIV	Division	√	√	√
ACCU		D.ACCU	Accumulation	√	√	√
AVG		D.AVG	Average	√	√	√
ABS		D.ABS	Absolute value	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
NEG		D.NEG	Two's complement	√	√	√

WNOT. D.WNOT(Negation)

Instruction format and parameter specification

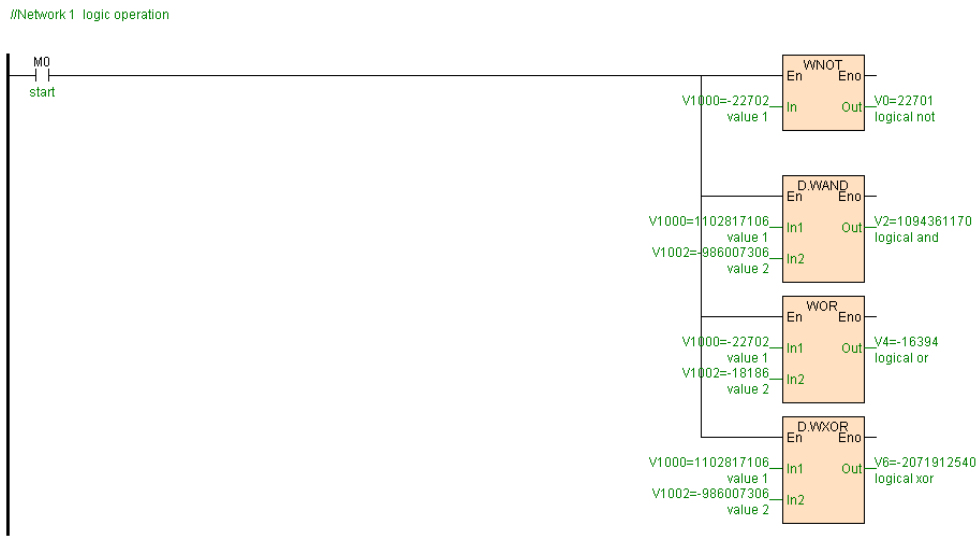
Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			WNOT En, In, Out D.WNOT En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

WNOT instruction bitwise negation output.

[Instruction example]



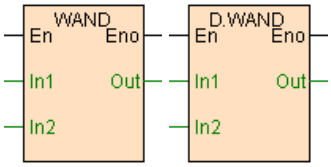
[Program description]

When M0=ON, execute as follows logical operation,V0= not V1000;V2V3=V1000V1001 and V1002V1003;V4=V1000 or V1002;V6V7=V1000V1001 xor V1002V1003,as follows table .

component	Initial value	Arithmetic result
V1000	10100111 01010010	V0 =01011000 10101101
V1000V1001	01000001 10111011 10100111 01010010	V2V3 =01000001 00111010 1010000001010010
V1002	10111000 11110110	V4 =10111111 11110110
V1002V1003	11000101 00111010 10111000 11110110	V6V7 =10000100 10000001 00011111 10100100

WAND. D.WAND(AND operation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			WAND En, In1, In2, Out D.WAND En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Input1	√		
In2	Input2	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

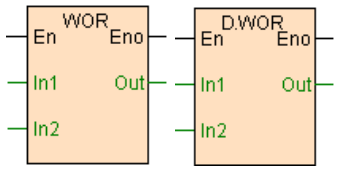
WAND instruction use for In1. In2 bitwise logical AND operation output.

[Instruction example]

Refer to WNOT instruction example.

WOR. D.WOR(OR operation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			WOR En, In1, In2, Out D.WOR En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Input1	√		

Parameter	Parameter define	Input	Output	Declare
In2	Input2	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

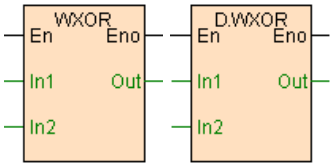
WOR instruction use for In1. In2 bitwise logical OR operation output.

[Instruction example]

Refer to WNOT instruction example.

WXOR. D.WXOR(XOR operation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			WXOR En, In1, In2, Out D.WXOR En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Input1	√		
In2	Input2	√		
Eno	Enable output		√	

Parameter	Parameter define	Input	Output	Declare
Out	Output		√	

[Instruction function and effect declare]

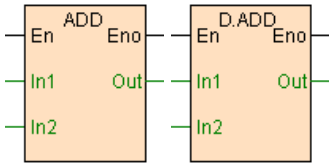
WXOR instruction use for In1. In2 bitwise logical XOR operation output.

[Instruction example]

Refer to WNOT instruction example.

ADD. D.ADD(Addition)

Instruction format and parameter specification

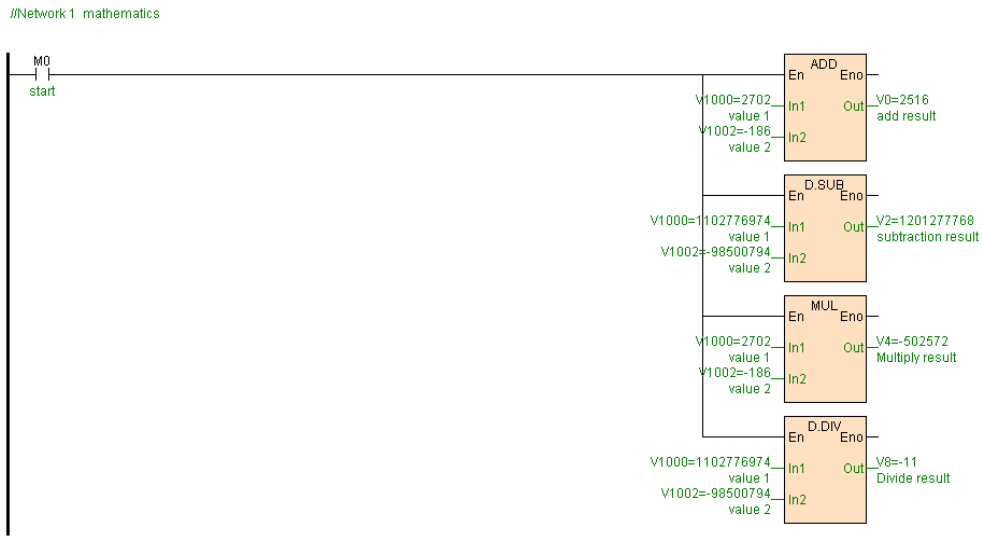
Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ADD En, In1, In2, Out D.ADD En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Augend	√		
In2	Addend	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

ADD instruction use for In1 add In2 output to Out.

[Instruction example]



[Program description]

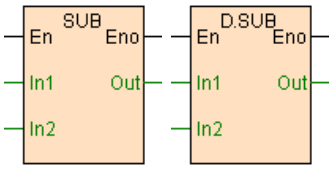
When M0=ON, execute as follows logical operation,V0= V1000+V1002;V2V3=V1000V1001-V1002V1003;V4V5=V1000* V1002;V8V9=V1000V1001\ V1002V1003,as follows table .

Component	**Initial value**	**Arithmetic result**
V1000	2702	V0 =2516
V1000V1001	1102776974	V2V3 =1201277768
V1002	-186	V4V5 =-502572
V1002V1003	-98500794	Quotient V8V9 =-11,remainder V10V11=19268240

SUB. D.SUB(Subtraction)

Instruction format and parameter specification

Language	**LD**	**FBD**	**IL**	**Program example**

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			SUB En, In1, In2, Out D.SUB En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Minuend	√		
In2	Subtracter	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

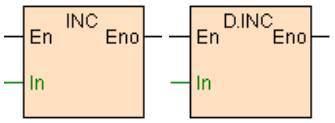
SUB instruction use for In1 subtract In2 output to out.

[Instruction example]

Refer to ADD instruction example.

INC. D.INC(Increase 1)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			INC En, In D.INC En, In	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Augend	√		
Eno	Enable output		√	

[Instruction function and effect declare]

1. INC instruction use for In increase 1 again store to In.

2. 16 bit instruction INC,if In=32767,Increase 1 change to -32768.

3. 32 bit instruction D.INC,if In=2147483647,Increase 1 change to -2147483648.

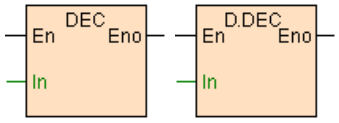
[Instruction example]

[Program description]

When X0=ON, then V0V1=V0V1+1

DEC. D.DEC(Decrease 1)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			DEC En, In D.DEC En, In	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Minuend	√		
Eno	Enable output		√	

[Instruction function and effect declare]

1. DEC instruction use for In decrease 1 again store to In.

2. 16 bit instruction DEC if In=-32768,Increase 1 change to 32767.

3. 32 bit instruction D.DEC if In=-2147483648,Increase 1 change to 2147483647.

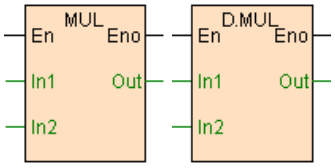
[Instruction example]

[Program description]

When X1=ON, then V2=V2-1

MUL. D.MUL(Multiplication)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			MUL En, In1, In2, Out D.MUL En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Multiplicand	√		
In2	Multiplier	√		
Eno	Enable output		√	
Out	Output		√	MUL: occupy 2 continuous component,D.MUL: occupy 4 continuous component

[Instruction function and effect declare]

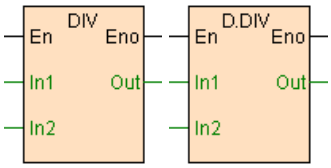
MUL instruction use for In1 multiply by In2 output to out. Out+1,32 bit instruction D.MUL use for (In1. In1+1) multiply by (In2. In2+1),arithmetic result output to (Out. Out+1. Out+2. Out+3).

[Instruction example]

Refer to ADD instruction example.

DIV. D.DIV(Division)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			DIV En, In1, In2, Out D.DIV En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In1	Dividend	√		
In2	Divisor	√		
Eno	Enable output		√	
Out	Output		√	DIV: occupy 2 continuous component,D.DIV: occupy 4 continuous component

[Instruction function and effect declare]

1. 16 bit instruction (DIV), use for In1 divide In2,arithmetic result quotient store to out ,remainder store to out+1.

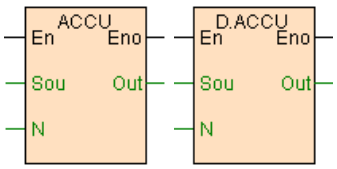
2. 32 bit instruction (D.DIV), use for (In1. In1+1) divide (In2. In2+1),arithmetic result quotient store to (Out. Out+1),remainder store to (Out+2. Out+3).

[Instruction example]

Refer to ADD instruction example.

ACCU. D.ACCU(Accumulation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ACCU En, Sou, N, Out D.ACCU En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
Sou	Source start component	√		ACCU: occupy N continuous component,D.ACCU: occupy 2N continuous component
N	Number of data	√		1~256
Eno	Enable output		√	
Out	Output		√	ACCU: occupy 2 continuous component,D.ACCU: occupy 4 continuous component

[Instruction function and effect declare]

1. ACCU instruction use for accumulation sum of N 16 bit integer start from Sou ,arithmetic result store to out. Out+1.

2. D.ACCU instruction use for accumulation sum of N 32 bit integer (216 bit register)start from Sou , arithmetic result store to out. Out+1. Out+2. Out+3.

[Instruction example]

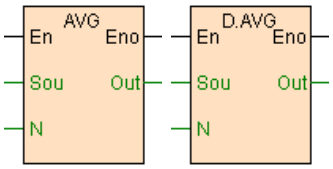
[Program description]

When M0=ON,V0V1= V1000+V1001+V1002+V1003,as follows table .

component	Initial value	Arithmetic result
V1000	2702	V0V1 =17839
V1001	16827	
V1002	-186	
V1003	-1504	

AVG. D.AVG(Average)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			AVG En, Sou, N, Out D.AVG En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		AVG: occupy N continuous component,D.AVG: occupy 2N continuous component
N	Number of data	√		1~256
Eno	Enable output		√	
Out	Output		√	AVG: occupy 2 continuous component,D.AVG: occupy 4 continuous component

[Instruction function and effect declare]

1. AVG instruction use for get average value N 16 bit integer start from Sou ,arithmetic result store to out, remainder store to Out+1.

2. D.AVG instruction use for get average value N 32 bit integer (two 16 bit register)start from Sou, arithmetic result store to out. Out+1, remainder store to Out+2. Out+3.

[Instruction example]

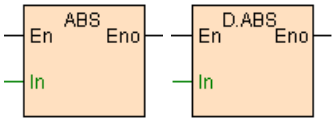
[Program description]

When M0=ON,V0= (V1000+V1001+V1002+V1003)\4,as follows table .

c omponent	Initial value	Arithmetic result
V1000	2702	V0=4459, remainder V1=3
V1001	16827	
V1002	-186	
V1003	-1504	

ABS. D.ABS(Absolute value)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ABS En, In D.ABS En, In	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	

[Instruction function and effect declare]

ABS instruction get In absolute value result again store to In.

[Instruction example]

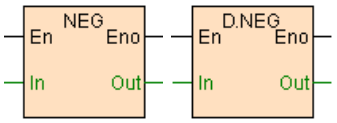
[Program description]

When M0=ON, get V1000. V1001V1002 absolute value, as follows table .

Component	Initial value	Absolute value result
V1000	2702	V1000=2702
V1001V1002	-12172869	V1001V1002=12172869

NEG. D.NEG(Two's complement)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			NEG En, In, Out D.NEG En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

NEG instruction get In two's complement.

[Instruction example]

[Program description]

When M0=ON, get V1000. V1001V1002 two's complement, as follows table .

Component	Initial value	Two's complement
V1000	2702	V0=-2702
V1001V1002	-12172869	V2V3=12172869

Floating point instruction

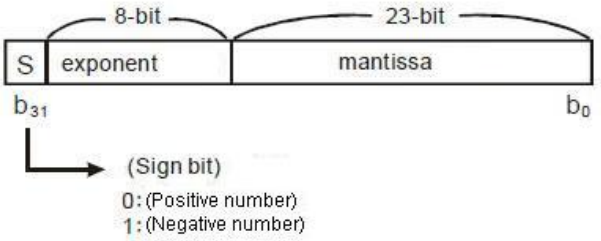
Floating point instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
FCMP			Floating point comparison	√	√	√
FZCP			Floating point regional comparison	√	√	√
FMOV			Floating point move instruction	√	√	√
FADD			Floating point addition	√	√	√
FSUB			Floating point subtraction	√	√	√
FMUL			Floating point multiplication	√	√	√
FDIV			Floating point division	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
FACCU			Floating point accumulation	√	√	√
FAVG			Floating point average	√	√	√
FMAX			Floating point maximum	√	√	√
FMIN			Floating point minimum	√	√	√
FTOI			Floating point convert to integer	√	√	√
ITOF		D.ITOF	Integer convert to floating point	√	√	√
FABS			Floating point absolute	√	√	√
FSQR			Floating point square root	√	√	√
FSIN			Sine	√	√	√
FCOS			Cosine	√	√	√
FTAN			Tangent	√	√	√
FASIN			Arcsine	√	√	√
FACOS			Arc cosine	√	√	√
FATAN			Arctangent	√	√	√
FLN			Natural logarithm	√	√	√
FLOG			The base-10 logarithm of a number	√	√	√

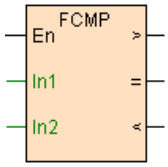
Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
FEXP			Nature exponential	√	√	√
FRAD			Angle convert to radian	√	√	√
FDEG			Radian convert to angle	√	√	√
FXY			Exponent	√	√	√

PLC floating point use IEEE754 standard, use 32 bit express floating point (occupy 2 registers), value range is ±2-126 to ±2+128 also as ±1.1755e-38 to ±3.4028e+38,format as follows:



FCMP(Floating point comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FCMP En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In1	Input1	√		
In2	Input2	√		
Out	Status output		√	Occupy 3 continuous component

[Instruction function and effect declare]

FCMP is floating point comparison instruction, at the same time output ＞. =. ＜ these 3 result.

[Instruction example]

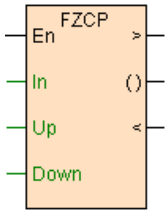
[Program description]

1. FCMP instruction get electricity from busbar and always execute.

2. When V0V1>23.456 then M10=ON, when V0V1=23.456 then M11=ON, when V0V1<23.456 then M12=ON.

FZCP(Floating point regional comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FZCP En, In, Up, Down, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Up	Region upper limit	√		
Down	Region lower limit	√		
Out	Status output		√	Occupy 3 continuous component

[Instruction function and effect declare]

1. FZCP is floating point regional comparison instruction, at the same time output ＞. =. ＜ these 3 result.

2. If region upper limit< region lower limit, then instruction automatic swap they .

[Instruction example]

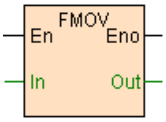
[Program description]

1. FZCP instruction get electricity from busbar and always execute.

2. When V0V1>9876.05 then M10=ON, when V0V1≤9876.05 moreover V0V1≥-23.12 then M11=ON, when V0V1<-23.12 then M12=ON.

FMOV(Floating point move)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FMOV En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

Floating point move instruction FMOV also name floating point valuation instruction,use for assigned floating point valuate to output register Out.

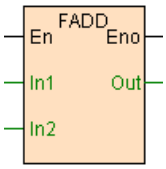
[Instruction example]

[Program description]

Program first scan valuate initial value,V0V1=35.6,V10V11=-2.632.

FADD(Floating point addition)

Instruction format and parameter specification

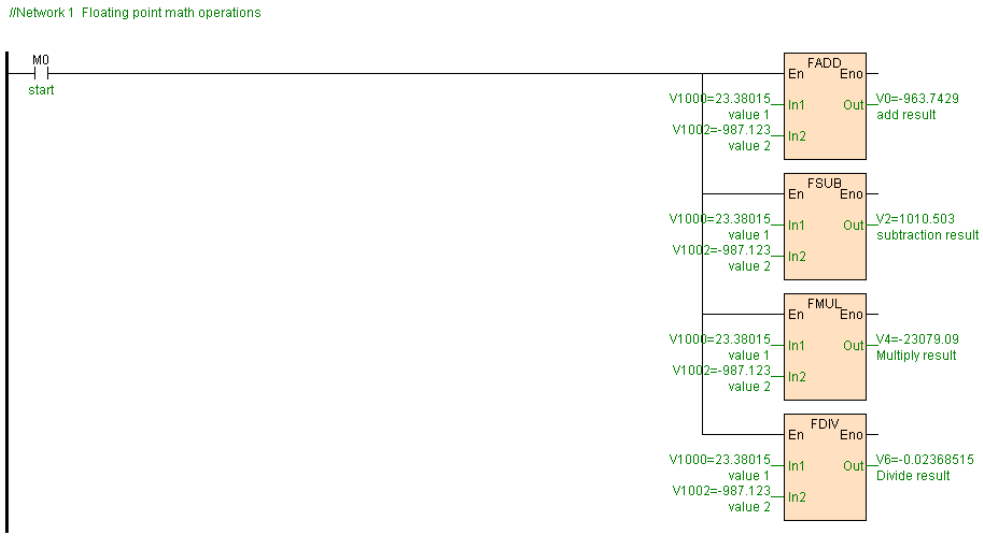
Language	LD	FBD	IL	Program example
Instruction format			FADD En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Augend	√		
In2	Addend	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FADD instruction use for floating point In1 add floating point In2 output to out.

[Instruction example]



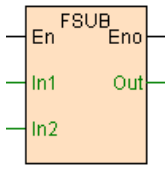
[Program description]

When M0=ON,excete as follows floating point arithmetic,V0V1= V1000V1001+V1002V1003;V2V3=V1000V1001-V1002V1003;V4V5=V1000V1001* V1002V1003;V6V7=V1000V1001\ V1002V1003,as follows table .

component	Initial value	Arithmetic result
V1000V1001	23.38015	V0V1 =-963.7429
		V2V3 =1010.503
V1002V1003	-987.123	V4V5 =-23079.09
		V6V7 =-0.02368515

FSUB(Floating point subtraction)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FSUB En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Minuend	√		
In2	Subtrahend	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

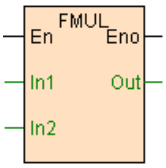
FSUB instruction use for floating point In1 subtracte floating point In2 output to out.

[Instruction example]

Refer to FADD instruction example.

FMUL(Floating point multiplication)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FMUL En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Multiplicand	√		
In2	Multiplier	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

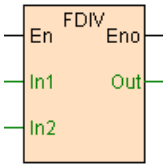
FMUL instruction use for floating point In1 multiply by floating point In2 output to out.

[Instruction example]

Refer to FADD instruction example.

FDIV(Floating point division)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FDIV En, In1, In2, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In1	Dividend	√		
In2	Divisor	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

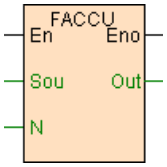
FDIV instruction use for floating point In1 divide floating point In2 output to out.

[Instruction example]

Refer to FADD instruction example.

FACCU(Accumulation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FACCU En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
N	Number of data	√		1~256
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FACCU instruction use for accumulation sum of N floating point start from Sou ,arithmetic result store to out.

[Instruction example]

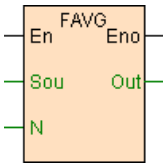
[Program description]

When M0=ON,V0V1= V1000V1001+V1002V1003+V1004V1005+V1006V1007,as follows table .

c omponent	Initial value	Arithmetic result
V1000V1001	198.012	V0V1 =316.4277
V1002V1003	23.781	
V1004V1005	-3.714	
V1006V1007	98.3487	

FAVG(Average)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FAVG En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
N	Number of data	√		1~256
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FAVG instruction use for get floating point average value N floating point start from Sou ,arithmetic result store to out.

[Instruction example]

[Program description]

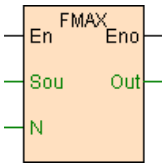
When M0=ON,V0V1=

(V1000V1001+V1002V1003+V1004V1005+V1006V1007)\4,as follows table .

component	Initial value	Arithmetic result
V1000V1001	198.012	V0V1=79.10693
V1002V1003	23.781	
V1004V1005	-3.714	
V1006V1007	98.3487	

FMAX(Floating point maximum)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FMAX En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
N	Number of data	√		2~256

Parameter	Parameter define	Input	Output	Declare
Eno	Enable output		√	
Out	Compare result output		√	

[Instruction function and effect declare]

FMAX instruction compare N floating point data start from Sou ,maximum output to Out.

[Instruction example]

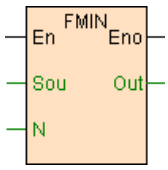
[Program description]

1. M0=ON, if V1000V1001=198.012. V1002V1003=23.781. V1004V1005=-3.714. V1006V1007=98.3487, then V0V1=198.012.

2. When M0=OFF, instruction stop execute ,Out remain unchanged.

FMIN(Floating point minimum)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FMIN En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		
N	Number of data	√		2~256
Eno	Enable output		√	
Out	Compare result output		√	

[Instruction function and effect declare]

FMIN instruction compare N floating point data start from Sou ,minimum output to Out.

[Instruction example]

[Program description]

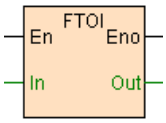
1. M0=ON, if V1000V1001=198.012. V1002V1003=23.781. V1004V1005=-3.714. V1006V1007=98.3487, then V0V1=-3.714.

2. When M0=OFF, instruction stop execute ,Out remain unchanged.

FTOI(Floating point convert to integer)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			FTOI En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FTOI instruction convert floating point to integer.

[Instruction example]

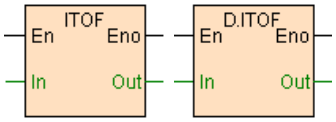
[Program description]

When M0=ON,FTOI instruction convert floating point V1000V1001 to integer result to V0V1,as follows table .

In component	Initial value	convert result
V1000V1001	-2345.987	V0V1 = -2346

ITOF. D.ITOF(Integer convert to floating point)

Instruction format and parameter specification

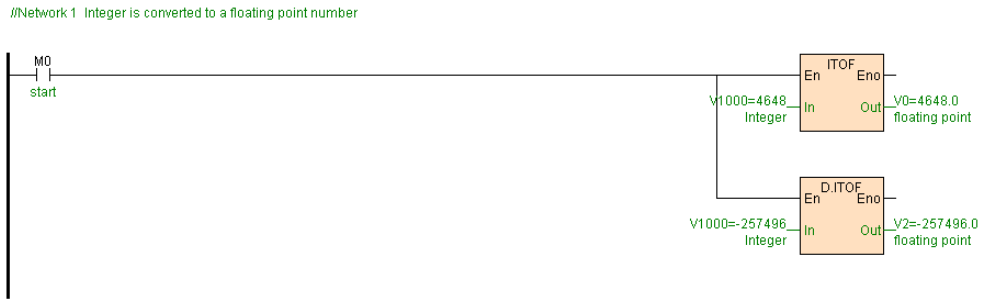
Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			ITOF En, In, Out D.ITOF En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

ITOF instruction convert Integer to floating point.

[Instruction example]



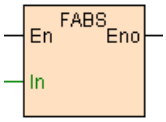
[Program description]

When M0=ON, ITOF instruction convert V1000 to floating point result store to V0V1, convert V1000V1001 to floating point result store to V2V3,as follows table .

In component	Initial value	convert result
V1000	4648	V0V1 = 4648.0
V1000V1001	-257496	V2V3 = -257496.0

FABS(Floating point absolute)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FABS En, In	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	

[Instruction function and effect declare]

FABS instruction get floating point In absolute value result again store to In.

[Instruction example]

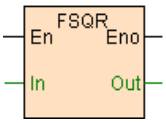
[Program description]

When M0=ON, get floating point V1000V1001 absolute value, as follows table .

component	Initial value	Absolute value result
V1000V1001	-23.3456	V1000V1001=23.3456

FSQR(Floating point square root)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FSQR En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FSQR instruction get floating point square root.

[Instruction example]

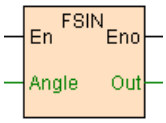
[Program description]

When M0=ON, get floating point V1000V1001 square root,as follows table .

component	Initial value	Result
V1000V1001	23.3456	V0V1=4.831728

FSIN(Sine)

Instruction format and parameter specification

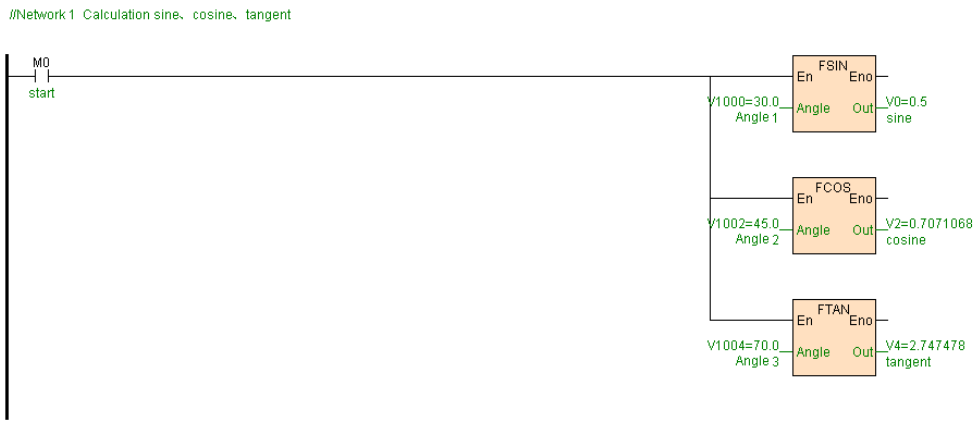
Language	LD	FBD	IL	Program example
Instruction format			FSIN En, Angle, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Angle	Angle	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FSIN instruction get sine of angle.

[Instruction example]



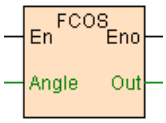
[Program description]

When M0=ON, get sine of floating point V1000V1001, cosine of floating point V1002V1003, tangent of floating point V1004V1005 ,as follows table .

Component	Initial value	Result
V1000V1001	30.0	SineV0V1=0.5
V1002V1003	45.0	CosineV2V3=0.7071068
V1004V1005	70.0	TangentV4V5=2.747478

FCOS(Cosine)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FCOS En, Angle, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Angle	Angle	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

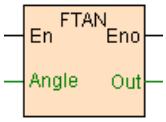
FCOS instruction get cosine of angle.

[Instruction example]

Refer to FSIN instruction example.

FTAN(Tangent)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FTAN En, Angle, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Angle	A ngle	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FTAN instruction get tangent of angle.

[Instruction example]

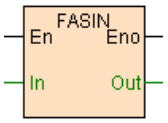
Refer to FSIN instruction example.

FASIN(Arcsine)

In

struction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			FASIN En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		range :-1 ~ 1
Eno	Enable output		√	
Out	Output radian		√	

[Instruction function and effect declare]

FASIN instruction get arcsine of In .InInput range :-1 ~ 1 among, if In exceed range , instruction not execute, Eno=OFF.

[Instruction example]

[Program description]

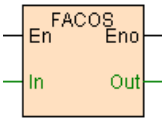
When M0=ON, get arcsine floating point V1000V1001 , arc cosine of floating point V1002V1003, arc tangent of floating point V1004V1005, as follows table .

c omponent	Initial value	Result

V1000V1001	0.8912	Arcsine V0V1=1.099984
V1002V1003	-0.3409	Arccosine V2V3=1.91867
V1004V1005	23.0912	Arctangent V4V5=1.527517

FACOS(Arc cosine)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FACOS En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		range :-1 ~ 1
Eno	Enable output		√	
Out	Output radian		√	

[Instruction function and effect declare]

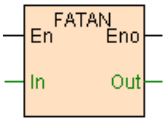
FACOS instruction get arc cosine of In .InInput range :-1 ~ 1 among , if In exceed range , instruction not execute, Eno=OFF.

[Instruction example]

Refer to FASIN instruction example.

FATAN(Arctangent)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FATAN En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output radian		√	

[Instruction function and effect declare]

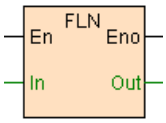
FATAN instruction get arc tangent of In.

[Instruction example]

Refer to FASIN instruction example.

FLN(Natural logarithm)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FLN En, In, Out	Download

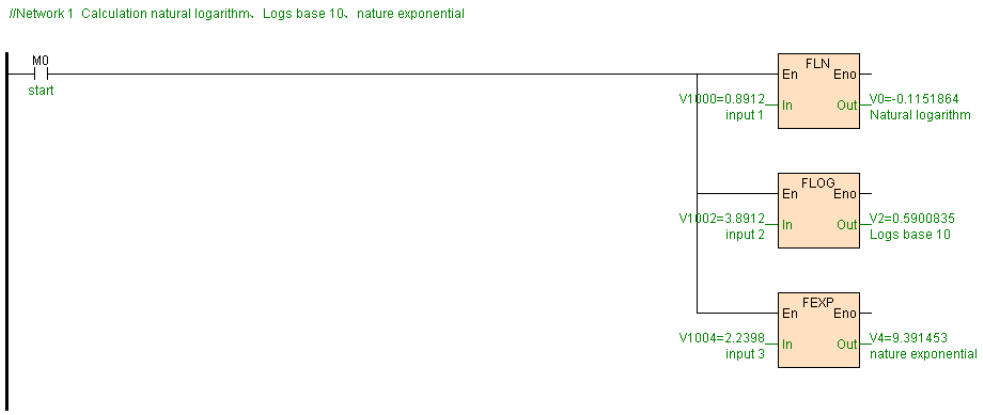
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FLN instruction get natural logarithm of In .InInput must greater than 0, no then instruction not execute, Eno=OFF.

[Instruction example]



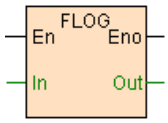
[Program description]

When M0=ON, get natural logarithm of floating point V1000V1001, the base-10 logarithm of floating point V1002V1003, nature exponential of floating point V1004V1005,as follows table .

component	Initial value	Result
V1000V1001	0.8912	Natural logarithmV0V1=-0.1151864
V1002V1003	3.8912	the base-10 logarithm of V2V3=0.5900835
V1004V1005	2.2398	Nature exponentialV4V5=9.391453

FLOG(The base-10 logarithm of a number)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FLOG En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FLOG instruction get the base-10 logarithm of In. In Input must greater than 0, no then instruction not execute, Eno=OFF.

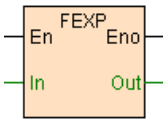
[Instruction example]

Refer to FLN instruction example.

FEXP(Nature exponential)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			FEXP En, In, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

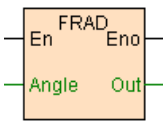
FEXP instruction get n ature exponential of In. In Input must less than 88.72284, no then instruction not execute, Eno=OFF.

[Instruction example]

Refer to FLN instruction example.

FRAD(Angle convert to radian)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FRAD En, Angle, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Angle	Angle	√		
Eno	Enable output		√	
Out	Output radian		√	

[Instruction function and effect declare]

FRAD instruction convert angle to radian.

[Instruction example]

[Program description]

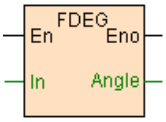
When M0=ON, convert angle V1000V1001to radian, convert radian V1002V1003 to angle, as follows table .

component	Initial value	Result
V1000V1001	30.0	V0V1=0.5235988
V1002V1003	1.57	V2V3=89.95438

FDEG(Radian convert to angle)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			FDEG En, In, Angle	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input radian	√		
Eno	Enable output		√	
Angle	A ngle		√	

[Instruction function and effect declare]

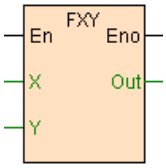
FDEG instruction convert radian convert to angle.

[Instruction example]

Refer to FRAD instruction example.

FXY(Exponent)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FXY En, X, Y, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
X	Base number	√		
Y	Exponent	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

FXY instruction get X^Y value. If X=0 moreover Y≤0 or X<0 moreover Y magnitude portion not equal to 0 instruction not execute, Eno=OFF.

[Instruction example]

[Program description]

When M0=ON, get V1000V1001^V1002V1003 value.

component	Initial value	Result
V1000V1001	30.0	V0V1=208.4877
V1002V1003	1.57	

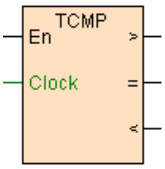
Clock instruction

System clock instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
TCMP			Real time clock comparison	√	√	√
TACCU			Time accumulative total	√	√	√
SCLK			Setup system clock	√	√	√
TIME			Time switch	√	√	√
DATE			Date switch	√	√	√
INVT			Count down	√	√	√

TCMP(Real time clock comparison)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TCMP En, Clock, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Clock	Clock compare start c omponent	√		Occupy 6 continuous component
Out	Status output		√	Occupy 3 continuous component

[Instruction function and effect declare]

1. TCMP use for compare system real time clock and clock setup clock ,if system real time clock ＞clock output＞state, if system real time clock =clock output=state, if system real time clock ＜clock output＜state.

2. Clock occupy 6 registers, respectively is year(0~9999). month(1~12). data(1~31). time(0~23). minute(0~59). second(0~59). if year set in 0~99 range , system default is 2000~2099 year.

3. If clock is invalid time, instruction not execute.

[Instruction example]

[Program description]

1. TCMP instruction get electricity from busbar and always execute.

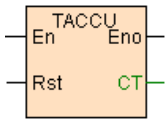
2. If V1000=2012. V1001=12. V1002=25. V1003=8. V1004=0. V1005=0, express setting time is 2012-12-25 8:0:0.

3. System clock exit in SV12~SV18, when system real time clock ＞Clock then M10=ON, when system real time clock =clock then M11=ON, when system real

time clock ＜clock then M12=ON.

TACCU(Time accumulative total)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TACCU En, Rst, CT	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Rst	Reset	√		
Eno	Enable output		√	
CT	Accumulative time		√	Occupy 6 continuous component

[Instruction function and effect declare]

1. TACCU instruction according to second unit accumulative En=ON time, output total accumulative second(CT. CT+1) and accumulative data (CT+2). hour(CT+3). minute(CT+4). second(CT+5) relative to the accumulative second.

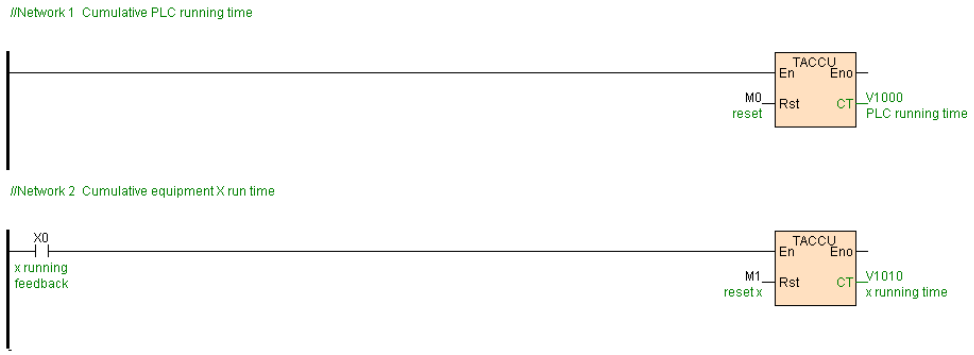
2.

Accumulative time

CT must use power-off preserve area register, default power-off preserve registers are V1000~V2047.

3. After accumulative second(CT. CT+1) reach maximum value 2147483647 automatic reset to 0.

[Instruction example]



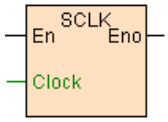
[Program description]

1. Network 1 get electricity from busbar , that is accumulative PLC running time .

2. Network 2 If X0 is equipmentx running feedback signal ,X0=ON, TACCU execute timing ,X0=OFF then not timing.

SCLK(Setup system clock)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			SCLK En, Clock	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Clock	Clock data start component	√		Occupy 6 continuous component

Parameter	Parameter define	Input	Output	Declare
Eno	Enable output		√	

[Instruction function and effect declare]

1. SCLK instruction modify the PLC real time clock by the set clock data .

2. Clock occupy 6 register, respectively is year(0~9999). month(1~12). data(1~31). hour(0~23). minute(0~59). second(0~59). if year set in0~99 range , system default is 2000~2099 year.

3. If clock is invalid time, instruction not execute.

4. SCLK instruction executed by edge.

5. Also can modify by program software menu[PLC/ set PLC clock], must not program. refer to "set PLC clock"

[Instruction example]

[Program description]

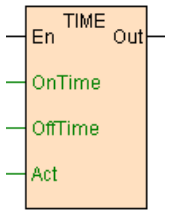
1. If V1000=2012. V1001=12. V1002=25. V1003=8. V1004=0. V1005=0, express setting time is 2012-12-25 8:0:0.

2. When M0=ON,PLC system clock setup is 2012-12-25 8:0:0.

TIME(Time switch)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			TIME En, OnTime, OffTime, Act, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
OnTime	ON actuation time	√		
OffTime	OFF actuation time	√		
Act	Control model	√		
Out	Status output		√	

[Instruction function and effect declare]

1. TIME instruction use week as control cycle, get by set ON. OFF actuation time control output.

2. Act is control model, it bits b0~b6 respectively are express monday~sunday control model, when it relevant bit is1 then express the data ON. OFF actuation time is valide, is 0 then express the data not actuation.

3. If ON actuation time <OFF actuation time , express in current data ON and OFF. if ON actuation time >OFF actuation time , express cross-day ON and OFF.

4. Ontime. Offtime if is register input, then high byte is hour(0~23) low byte is minute(0~59);if is constant input, input format is :hh:mm(as 8 clock 5 minute,input 08:05).

5. If Ontime is invalid time then ON actuation invalid. If Offtime is invalid time then OFF actuation invalid.

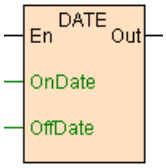
[Instruction example]

[Program description]

Act=127(01111111) express monday ~sunday valid, when M0=ON,TIME instruction execute, every day 8:30~16:30 among Y0=ON, others timeY0=OFF.

DATE(Date switch)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			DATE En, OnDate, OffDate, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
OnDate	ON actuation data	√		
OffDate	OFF actuation data	√		
Out	Status output		√	

[Instruction function and effect declare]

1. DATE instruction use year as control cycle, get by ON. OFF actuation time control output.

2. If ON actuation data <OFF actuation data , express in current year ON and OFF. if ON actuation data >OFF actuation data , express in cross-year ON and OFF.

3. OnDate. OffDate if is register input, then high byte is month(1~12) low byte is data(1~31); if is constant input, input format is :mm-dd(as august 5,Input 08-05).

4. If OnDate is invalid data then ON actuation invalid. If OffDate is invalid data then OFF actuation invalid .

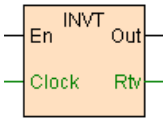
[Instruction example]

[Program description]

When M0=ON ,DATE instruction executing, from current year august 1to next year january 31Y0=ON, others time Y0=OFF.

INVT(Count down)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			INVT En, Clock, Out, Rtv	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Clock	Count down start time component	√		Occupy 6 continuous component
Out	Count down to output		√	
Rtv	Remaining time		√	Occupy 4 continuous component

[Instruction function and effect declare]

1. INVT instruction according to set

count down time

clock calculate from current value to set time still remaining data(Rtv). hour(Rtv+1). minute(Rtv+2). second(Rtv+3),when reach the timing time output Out.

2. Clock occupy 6 registers ,respectively year(0~9999). month(1~12). data(1~31). time(0~23). minute(0~59). second(0~59). if year set in 0~99 range , system default as 2000~2099 year.

3. if Clock is invalid time, instruction not execute.

[Instruction example]

[Program description]

1. If V1000=2013. V1001=12. V1002=25. V1003=8. V1004=0. V1005=0, express set count down time is 2013-12-25 8:0:0.

2. When M0=ON,INVT instruction start count down, automatic calculate from current value to 2013-12-25 8:0:0 remaining data. hour. minute. second, after

timing time reach output Y0=ON.

Communication instruction

Communication instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
SUM	SUM.LB		SUM verify	√	√	√
BCC	BCC.LB		BCC verify	√	√	√
CRC	CRC.LB		CRC verify	√	√	√
LRC	LRC.LB		LRC verify	√	√	√
COMM	COMM.LB		Free communications	√	√	√
MODR			Modbus read	√	√	√
MODW			Modbus write	√	√	√
HWRD			Speedbus read	√	√	√
HWWR			Speedbus write	√	√	√
RCV			Receive communication data	√	√	√
XMT	XMT.LB		Sent communication data	√	√	√
FROM			Extend module CR register read	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
TO			Extend module CR register write	√	√	√
TCPMDR			Modbus TCP read	√	√	√
TCPMDW			Modbus TCP write	√	√	√
TCPHWR			Speedbus TCP read	√	√	√
TCPHWW			Speedbus TCP write	√	√	√

Note:PLC support standard Modbus protocol. Speedbus protocol. freedom communication protocol, PLC use as slave no need any communication program, upper computer (configuration software. touch panel. text display etc. HMI) can direct use Modbus protocol access PLC. refer to "communication address code table "

PLC communication main features

:

1. Speedbus protocol is a efficient . high speed communication protocol, support disperse . blended data transfer, communication efficient very well.

2. PLC support maximum 5 communication port, the same function of all communication port , all can use for program . upload download program. monitor . networking.

3. 5 communication port fully independent , concurrent working , support different different baud rate . different protocol format . different manufacturer equipment connected in same network.

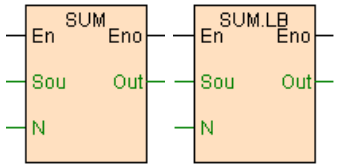
4. Need not worry about communication port collision problem, need not control communication instruction execute time sequence ,all communication instruction can concurrent get electricity execute.

5. Via communication instruction Out item can intuitive judgment communicate succeed or not , can use it for alarm the communicate fail between slave.

6. PLC communication is zero spare on network physics level , if slave equipment not support so high speed communication, can assigned SV141 to insert a interval among the communication instruction.

SUM. SUM.LB(SUM verify)

Instruction format and parameter specification

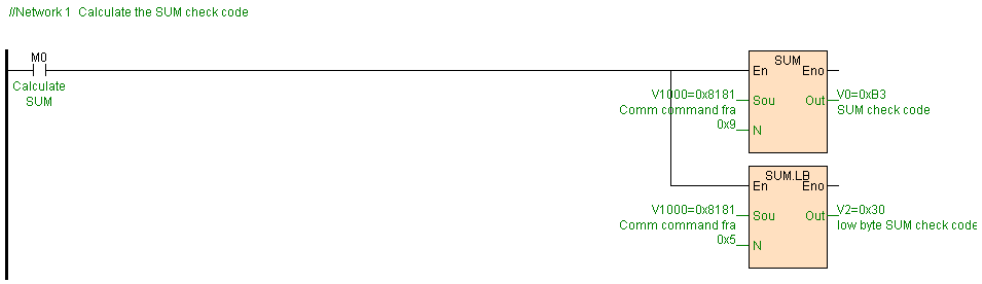
Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			SUM En, Sou, N, Out SUM.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		SUM: occupy (N+1)\2 continuous component,SUM.LB: occupy N continuous component
N	Verify number of bytes	√		1~256
Eno	Enable output		√	
Out	SUM verify code output		√	

[Instruction function and effect declare]

Sum verify code computing method: use for get accumulation sum start from Sou N number of bytes , get low byte, exceed 256 part overflow.SUM.LB is Low byte model, only calculate sum verify code of low byte, high byte notuse .

[Instruction example]



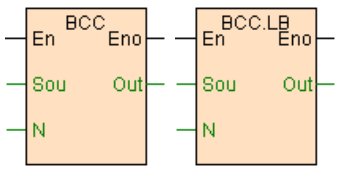
[Program description]

When M0=ON, calculate SUM verify code, as follows table .

Sou c omponent	Initial value	SUM Output	SUM.LB Output
V1000	0x8181	V0=0xB3	V2=0x30
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

BCC. BCC.LB(BCC verify)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			BCC En, Sou, N, Out BCC.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		BCC: occupy (N+1)\2 continuous component,BCC.LB: occupy N continuous component
N	verify sumber of bytes	√		1~256
Eno	Enable output		√	
Out	BCC code output		√	

[Instruction function and effect declare]

BCC verify code computing method: use for XOR operation start from Sou N Number of bytes .BCC.LB is Low byte model, only calculate low byte BCC verify code, high byte not use.

[Instruction example]

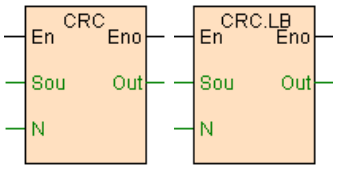
[Program description]

When M0=ON, calculate BCC verify code, as follows table .

Sou c omponent	Initial value	BCC Output	BCC.LB Output
V1000	0x8181	V0=0xB	V2=0x8A
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

CRC. CRC.LB(CRC verify)

Instruction format and parameter specification

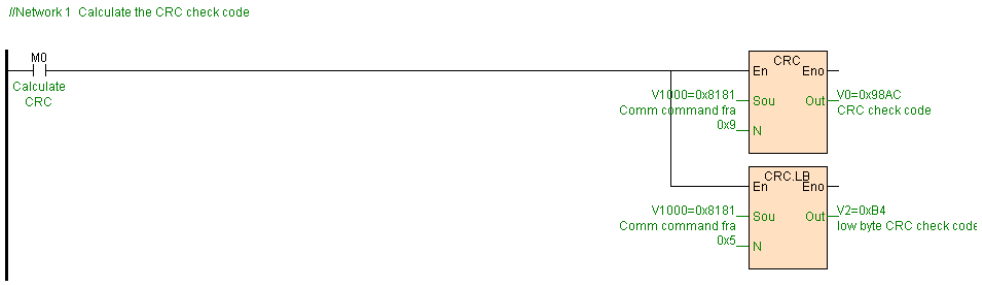
Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			CRC En, Sou, N, Out CRC.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		CRC: occupy (N+1)\2 continuous component,CRC.LB: occupy N continuous component
N	Verify number of bytes	√		1~256
Eno	Enable output		√	
Out	CRC code output		√	

[Instruction function and effect declare]

CRC calculate CRC verify code start from Sou N number of bytes .CRC.LB is Low byte model, only calculate low byte CRC verify code, high byte not use.

[Instruction example]



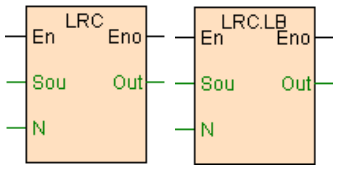
[Program description]

When M0=ON, calculate CRC verify code, as follows table .

Sou c omponent	Initial value	CRC Output	CRC.LB Output
V1000	0x8181	V0=0x98AC	V2=0xB4 V3=0x51
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

LRC. LRC.LB(LRC verify)

Instruction format and parameter specification

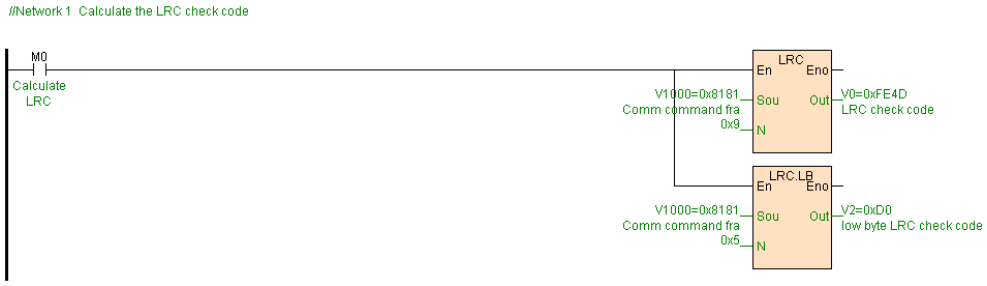
Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			LRC En, Sou, N, Out LRC.LB En, Sou, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Sou	Source start component	√		LRC: occupy (N+1)\2 continuous component,LRC.LB: occupy N continuous component
N	Verify number of bytes	√		1~256
Eno	Enable output		√	
Out	LRC code output		√	

[Instruction function and effect declare]

LRC verify code computing method: use for get accumulation sum start from Sou N number of bytes and then O 2's complement code. LRC.LB is low byte model, only calculate low byte LRC verify code, high byte not use.

[Instruction example]



[Program description]

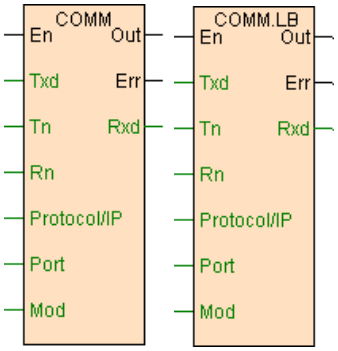
When M0=ON, calculate LRC verify code,as follows table .

Sou c omponent	Initial value	LRC Output	LRC.LB Output
V1000	0x8181	V0=0xFE4D	V2=0xD0 V3=0xFE
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

COMM. COMM.LB(Free communications)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			COMM En, Txd, Tn, Rn, Ptotocol/IP, Port, Mod,Out, Err, Rxd COMM.LB En, Txd, Tn, Rn, Ptotocol/IP, Port, Mod,Out, Err, Rxd	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Txd	Send data start component	√		
Tn	Send data number of bytes	√		0~512
Rn	Receive data number of bytes	√		0~512
Protocol/IP	Communication protocol or IP	√		
Port	Communication port	√		
Mod	Communication mode	√		0-Serial, 1-TCP, 2-UDP
Out	Communication complete output		√	
Err	Error indication		√	
Rxd	Receive data start component		√	

[Instruction function and effect declare]

1. When PLC communicate in freedom protocol between external equipment,use COMM instruction send and receive data .At the moment PLC is master, external equipment is slave.

2. When Tn=0 ,COMM instruction only receive data without send data .When Rn=0 ,COMM instruction only send data without receive data .When Tn=Rn=0 ,COMM instruction not execute.

3. Mod communication mode:

When Mod=0, perform serial communication through the serial port (RS232/RS485), the Protocol/IP defines the communication format (baud rate, data bit, stop bit, check mode), and the Port defines the serial port number.

When Mod=1, communicate via Ethernet in TCP mode, Protocol/IP defines the IP address, and Port defines the communication port.

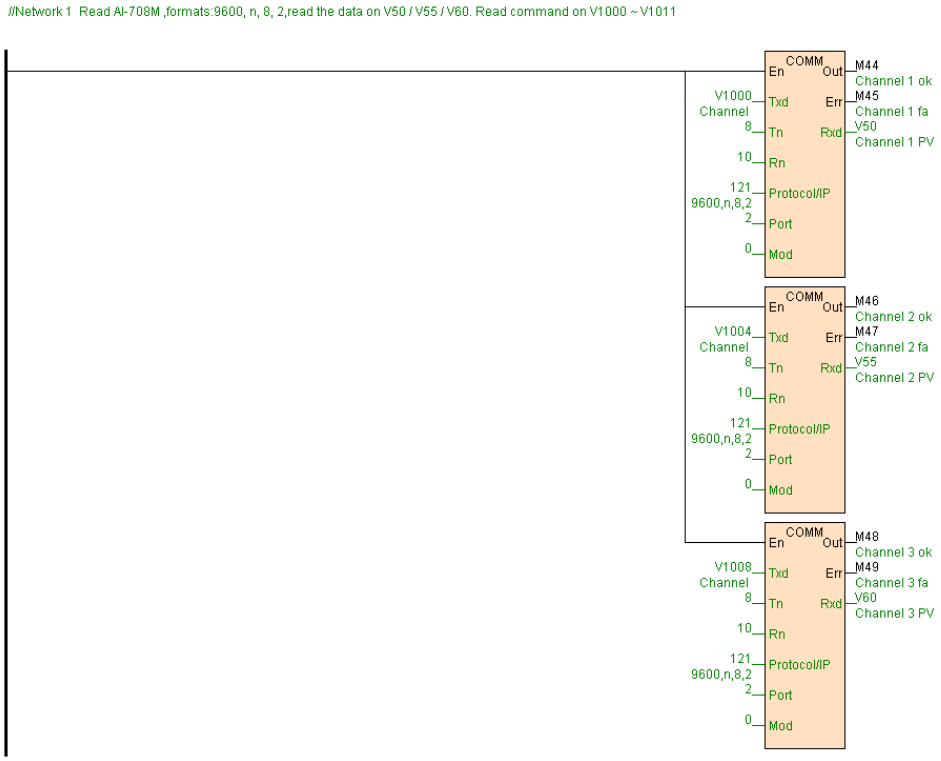
When Mod=2, it communicates via Ethernet in UPD mode, Protocol/IP defines the IP address, and Port defines the communication port.

4. When the COMM command is executed, Txd is the starting Tn bytes of data to the serial port specified by Port. After sending, if Rn>0, it will automatically turn to the receiving state, and when the receiving is completed, Out=ON, the received data Put it in Rxd; if Rn=0, no data will be received. Out=ON, the system will execute the next communication command. Err=ON if the communication command is not completed.

5. There are two ways to send the COMM instruction: high and low byte sending method (COMM) and only low byte sending method (COMM.LB)

6. The COMM command executes serial communication and can be used simultaneously with the XMT, MODR, MODW, HWRD, and HWWR commands. But it cannot use the same serial port as the RCV command.

[Instruction example]



[Program description]

1. According to AI-708M itinerant detector communication protocol ,read 3 channel measure value from AI-708M itinerant detector communication instruction put on initial register value table " read AI-708M itinerant detector command" :

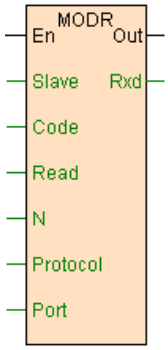
component	Initial value	Declare
V1000	0x8383	First channel read command
V1001	0x0152	
V1002	0x0000	
V1003	0x0155	
V1004	0x8484	Second channel read command
V1005	0x0152	
V1006	0x0000	
V1007	0x0156	
V1008	0x8585	Third channel read command
V1009	0x0152	
V1010	0x0000	
V1011	0x0157	

2. 3 COMM instruction get electricity from busbar and always execute, PLC automatic in sequence send communication command to AI-708M itinerant detector moreover use for returned measure value output to instruction Rxd.

3. Do not warry about communication port conflict , do not control communication instruction executed time sequence , system automatic executed completely .

MODR(Modbus read)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			MODR En, Slave, Code, Read, N, Ptotocol, Port, Out, Rxd	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slave	Slave equipment address	√		
Code	Function code	√		
Read	Read data start address	√		
N	Number of data	√		1~127

Parameter	Parameter define	Input	Output	Declare
Protocol	Communication protocol	√		
Port	Communication port	√		
Out	Communication complete Output		√	
Rxd	Receive data start component		√	Occupy N continuous component

[Instruction function and effect declare]

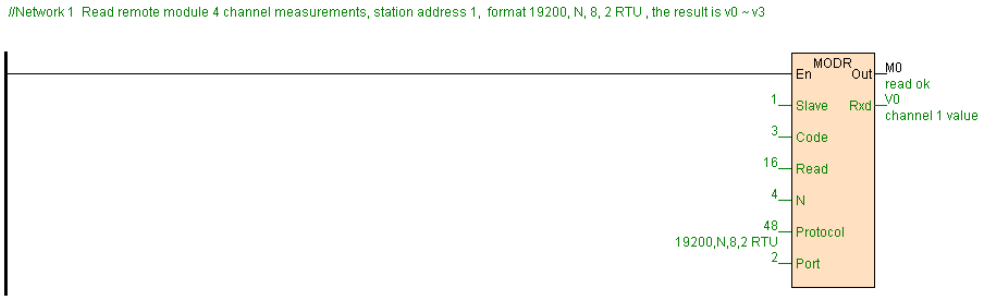
1. MODR instruction use for communication with all the third party equipment are support Modbus protocol.

2. When PLC communication with external equipment by the serial port ,use MODR instruction read data from external equipment . This moment PLC as master ,external equipment as slave.

3. MODR instruction do not write any verify code, it automatic verified the returned data ,verify correct Out=ON, read data put on Rxd.

4. MODR instruction can use with COMM . XMT. MODW. HWRD. HWWR instruction at the same time .but can not use the same communication port with RCV instruction.

[Instruction example]



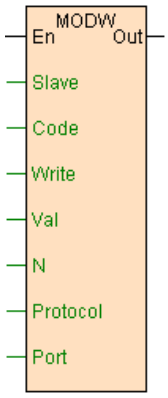
[Program description]

1. MODR instruction read station 1 external module (If module is S04AI) 4 channel measured value (CR Parameter no.16~19), read data store to V0~V3.

2. Different model module the CR number not the same, detail information refer to Hardware manual" extend module Parameter" section.

MODW(Modbus write)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			MODW En, Slave, Code, Write, Val, N, Ptotocol, Port, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slave	Slave equipment address	√		
Code	Function code	√		
Write	Write target start address	√		
Val	Be rote data start component	√		Occupy N continuous component

Parameter	Parameter define	Input	Output	Declare
N	Number of data	√		1~127
Protocol	Communication protocol	√		
Port	Communication port	√		
Out	Communication complete Output		√	

[Instruction function and effect declare]

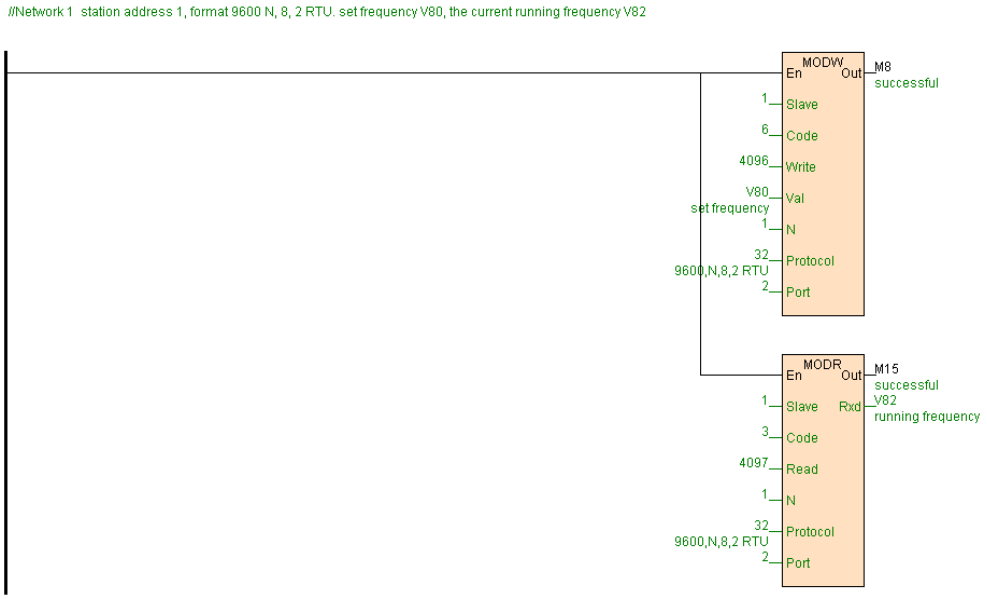
1. MODW instruction use for communication with all the third party equipment are support Modbus protocol.

2. When PLC communication with external equipment by the serial port ,use MODW instruction write data to external equipment . This moment PLC as master ,external equipment as slave Modbus .

3. MODW instruction do not write any verify code, it automatic verified the returned data ,verify correct Out=ON express write succeed.

4. MODW instruction can use with COMM . XMT. MODR. HWRD. HWWR instruction at the same time .but can not use the same communication port with RCV instruction.

[Instruction example]



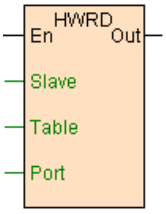
[Program description]

1. According to inovance inverter communication protocol (please refer to inovance inverter manual part of communication), preset frequency Modbus address is 4096,MODW instruction write V80 value to inverter real time.

2. Running frequency Modbus address is 4097,MODR instruction read the current frequency of the inverter store to V82 .

HWRD(Speedbus read)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			HWRD En, Slave, Table, Port, Out	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slave	Slave equipment address	√		
Table	Speedbus read communication table	√		
Port	Communication port	√		
Out	Communication complete Output		√	

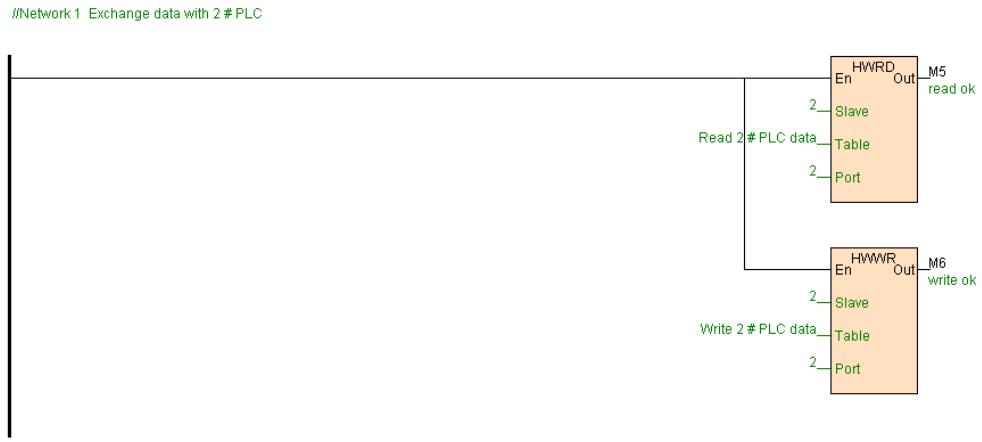
[Instruction function and effect declare]

1. HWRD instruction according to defined "Speedbus read communication table " automatic swap the data with slave .

2. Speedbus protocol support disperse . blended data transfer, communication efficient very well.

3. HWRD instruction can use with COMM . XMT. MODR. MODW. HWWR instruction at the same time .but can not use the same communication port with RCV instruction .

[Instruction example]



[Program description]

1. Define Speedbus read communication table " read 2# PLC data "as follows:

Sequence number	Read data from slave	Write date to slave
1	X0	M10
2	X3	M11
3	V11	V80
4	V12	V81
5	AI0	V20
6	AI1	V21

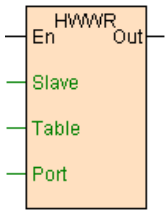
2. Define Speedbus write communication table " write 2# PLC data "as follows:

Sequence number	Read data from slave	Write date to slave
1	X0	M100
2	X1	M101
3	V0	V100
4	V50	V102
5	Y4	M0
6	Y5	Y0
7	V60	V200
8	V61	V201

3. HWRD. HWWR instruction get electricity from busbar and always execute, according above define Speedbus read (write) communication table ,automatic swap data with 2#PLC .

HWWR(Speedbus write)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			HWWR En, Slave, Table, Port, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slave	Slave equipment address	√		
Table	Speedbus write communication table	√		
Port	Communication port	√		
Out	Communication complete Output		√	

[Instruction function and effect declare]

1. HWWR instruction according to define "Speedbus write communication table " automatic swap data with slave.

2. Speedbus protocol support disperse . blended data transfer, communication efficient very well.

3. HWWR instruction can use with COMM . XMT. MODR. MODW. HWRD instruction at the same time .but can not use the same communication port with RCV instruction .

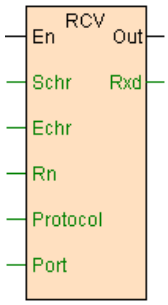
[Instruction example]

Refer to HWRD instruction example.

RCV(Receive communication data)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			RCV En, Schr, Echr, Rn, Ptotocol, Port, Out, Rxd	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Schr	start character	√		
Echr	ending character	√		
Rn	Receive data Number of bytes	√		0~512
Protocol	Communication protocol	√		
Port	Communication port	√		
Out	Communication complete Output		√	
Rxd	Receive data start component		√	

[Instruction function and effect declare]

1. At upper computer is communication master ,PLC is communication slave , moreover upper computer must use freedom communication protocol need use RCV instruction.

2. RCV instruction passive receiving data sent from upper computer , if need response to upper computer then use XMT instruction sent the response data.

3. Schr is start character define, if Schr=0 express no start character, if Schr high byte=0 low byte<>0 express only one start character (e.x.Schr=0x003A,start character is 0x3A), if Schr high byte<>0 express2start character (e.x. Schr=0x833A,start character is 0x3A. 0x83).

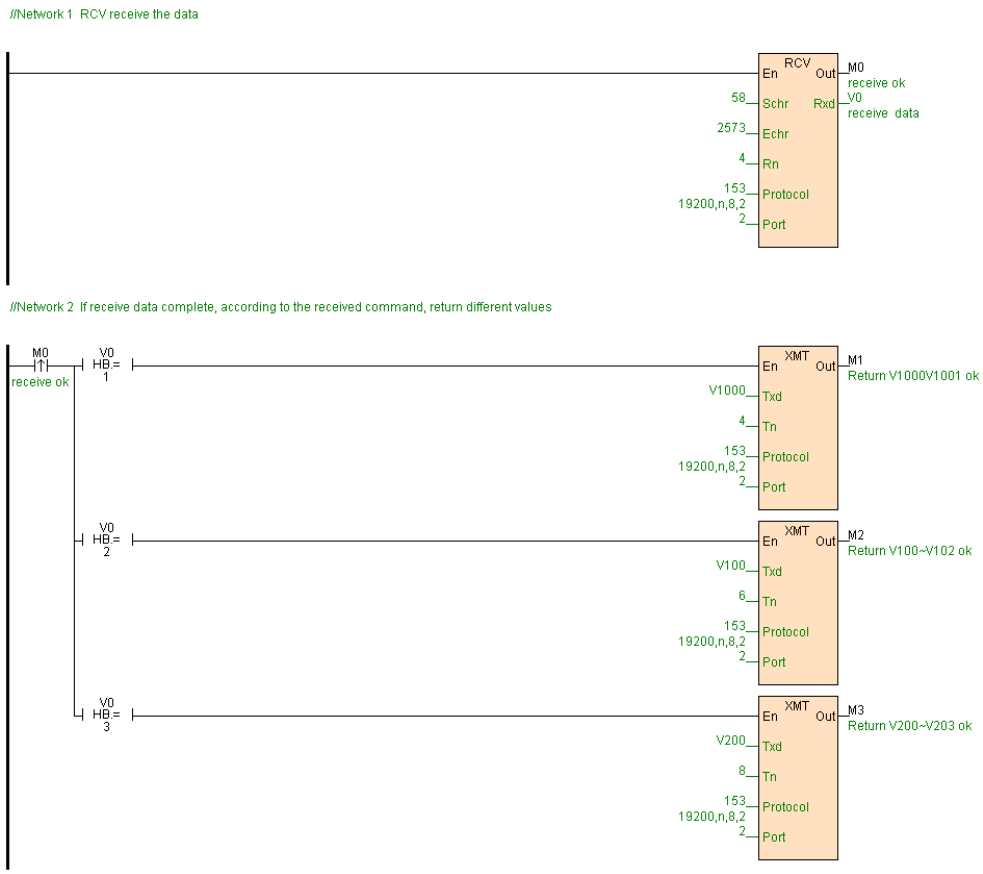
4. Echr is ending character define, if Echr=0 express no ending character, if Echr high byte=0 low byte<>0 express only one ending character (e.x. Echr=0x000D, ending character is 0x0D), if Echr high byte<>0 express2 ending character (e.x. Echr=0x0A0D, ending character is 0x0D. 0x0A).

5. If define start character or ending character ,RCV according to start character. ending character process match , only match correct express communication succeed , receive data output to Rxd;When Schr and Echr all is 0,that is start character and ending character all not define, then RCV instruction use for according to communication port communicate overtime to judge the start end of the communication frame .When receive the first byte of a new frame or communicate overtime RCV instruction will automatic reset Out and Rxd.

6. Rn is receive data number of bytes, e.x. :RCV instruction want receive 22 bytes send from upper computer , then assign Rn=22.Note: if the length of the command send from upper computer not fixed, then use for receive number of bytes define to 0, express disregard receive number of bytes.

7. One communication port only use one RCV instruction, moreover use with COMM. MODR. MODW. HWRD. HWWR instructionuse at the same communication port .

[Instruction example]



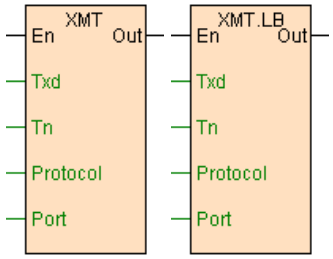
[Program description]

1. Network 1 start one RCV passive receiving instruction, start character match 0x3A, ending character match 0x0D 0x0A, receive 4 number of bytes, communication format 19200,n,8,2, communication port 2,receive data put on V0V1,receive data correct then each communicate M0=ON.

2. When M0=ON receive data correct, compare second byte (V0 high byte), if equal to 1 then execute XMT instruction return V1000V1001 , if equal to 2 then execute XMT instruction return V100~V102, if equal to 3 then execute XMT instruction return V200~V203.

XMT. XMT.LB(Sent communication data)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 8 bit Instruction format			XMT En, Txd, Tn, Ptotocol, Port, Out XMT.LB En, Txd, Tn, Ptotocol, Port, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Txd	Send data start component	√		
Tn	Send data Number of bytes	√		0~512
Protocol	Communication protocol	√		
Port	Communication port	√		
Out	Communication complete Output		√	

[Instruction function and effect declare]

1. At upper computer is communication master ,PLC is communication slave , moreover upper computer must use freedom communication protocol need use XMT instruction.

2. XMT instruction general use for cooperate RCV instruction , RCV instruction receive data send from upper computer , if need response to upper computer then use XMT instruction send response data.

3. XMT instruction have two send modes :high low byte send mode (XMT) and only send low byte mode (XMT.LB).

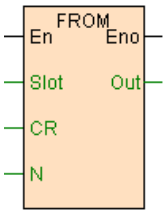
4. XMT instruction can repeat ,XMT instruction different from COMM instruction ,it only can send data can not receive data .

[Instruction example]

Refer to RCV instruction example.

FROM(Extend module CR register read)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FROM En, Slot, CR, N, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slot	Module position number	√		
Cr	CR register be read	√		
N	Number of CR be read	√		1~120
Eno	Enable output		√	

Parameter	Parameter define	Input	Output	Declare
Out	Return start component		√	Occupy N continuous component

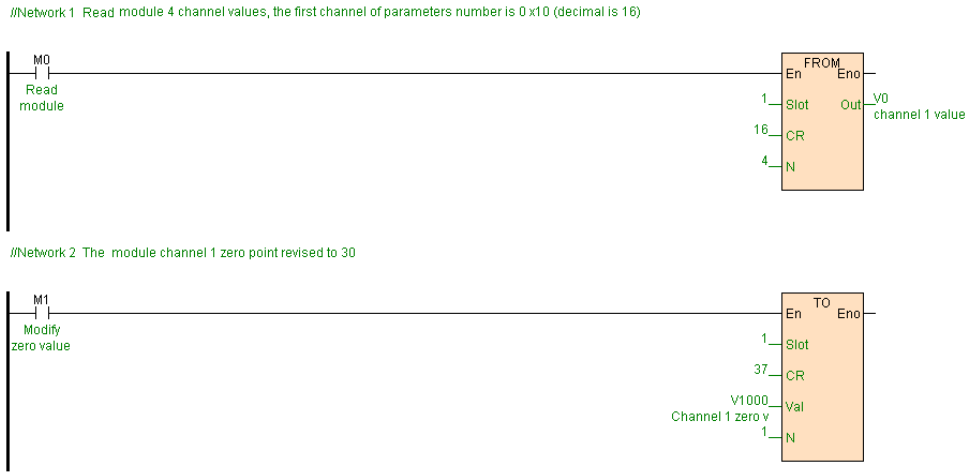
[Instruction function and effect declare]

1. FROM use for read extend module parameter get by parallel bus in program .

2. PLC automatic allocation extend module IO channel corresponding component address on parallel bus, moreover real time refresh IO of extend module , so general not use FROM instruction.detail refer to "PLC hardware configure " section.

3. Different model extend module CR register not the same ,detail refer to hardware manual "extend module parameter" section.

[Instruction example]



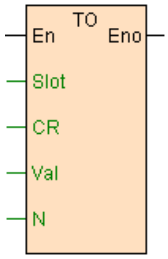
[Program description]

1. When M0=ON, read first extend module 4 Input channel measure value .

2. When M1=ON, modify the zero point corrected value to 30 (If V1000=30)of first channel of the extend module .

TO(Extend module CR register write)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TO En, Slot, CR, Val, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Slot	Module position number	√		
Cr	CR register be wrote	√		
Val	Data start component be wrote	√		Occupy N continuous component
N	Number of CR be wrote	√		1~120
Eno	Enable output		√	

[Instruction function and effect declare]

1. TO use for write extend module parameter get by parallel bus in program. May "PLC hardware configure " window configure module parameter, when PLC program be downloaded automatic download ,So general need not use TO instruction.

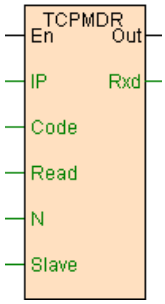
2. Different model extend module CR register not the same ,detail refer to hardware manual "extend module parameter" section.

[Instruction example]

Refer to FROM instruction example.

TCPMDR(Modbus TCP read)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TCPMDR En, IP, Code, Read, N, Slave, Out, Rxd	Download

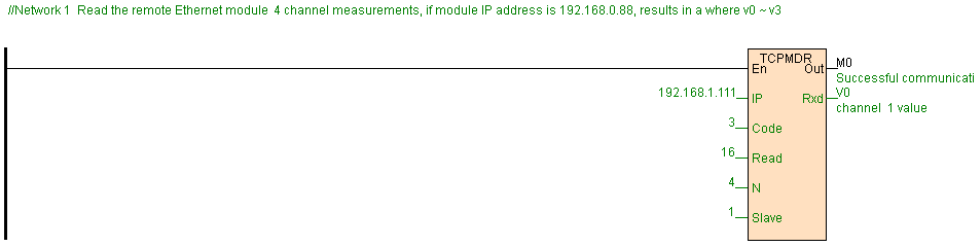
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
IP	Slave equipment IP address	√		
Code	Function code	√		
Read	Read data start address	√		
N	Number of data	√		1~127
Slave	Slave equipment address	√		

Parameter	Parameter define	Input	Output	Declare
Out	Communication complete Output		√	
Rxd	Receive data start component		√	Occupy N continuous component

[Instruction function and effect declare]

TCPMDR instruction is used for reading the data of the device supporting Modbus TCP protocol.

[Instruction example]



[Program description]

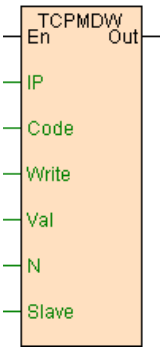
1. TCPMDR instruction will read the four channels' measurement value (CR parameters No. 16 to 19) of the remote module of the IP address 192.168.1.111 , and then the reading data will be stored in V0 ~ V3.

2. Different model module the CR number not the same, detail information refer to Hardware manual" extend module Parameter" section.

TCPMDW(Modbus TCP write)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

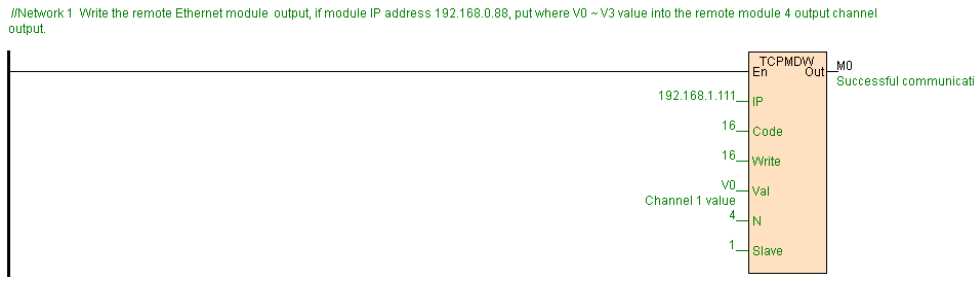
Language	LD	FBD	IL	Program example
Instruction format			TCPMDW En, IP, Code, Write, Val, N, Slave, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
IP	Slave equipment IP address	√		
Code	Function code	√		
Write	Write target start address	√		
Val	Be rote data start component	√		Occupy N continuous component
N	Number of data	√		1~127
Slave	Slave equipment address	√		
Out	Communication complete Output		√	

[Instruction function and effect declare]

TCPMDW instruction writes the data to the device supporting the Modbus TCP protocol.

[Instruction example]

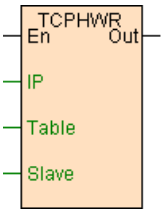


[Program description]

1. TCPMDW instruction writes the data V0 ~ V3 to the four output channels (CR parameters No. 16 to 19) of the remote module of the IP address 192.168.1.111.
2. Different model module the CR number not the same, detail information refer to Hardware manual" extend module Parameter" section.

TCPHWR(Speedbus TCP read)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TCPHWR En, IP, Table, Slave, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
IP	Slave equipment IP address	√		
Table	Speedbus read communication table	√		

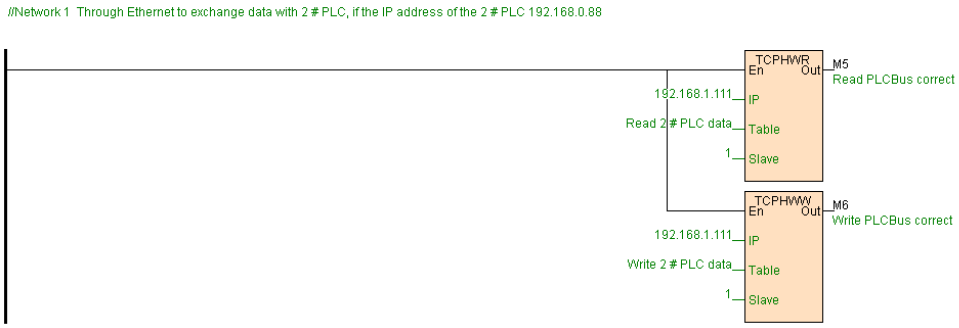
Parameter	Parameter define	Input	Output	Declare
Slave	Slave equipment address	√		
Out	Communication complete Output		√	

[Instruction function and effect declare]

1. TCPHWR instruction automatically exchanged the data with the slave device according to the definition of "Speedbus read communication table " using Speedbus TCP protocol.

2. Speedbus TCP protocol support disperse . blended data transfer, communication efficient very well.

[Instruction example]



[Program description]

1. Define Speedbus read communication table " read 2# PLC data "as follows:

Sequence number	Read data from slave	Write date to slave
1	X0	M10
2	X3	M11
3	V11	V80
4	V12	V81
5	AI0	V20
6	AI1	V21

2. Define Speedbus write communication table " write 2# PLC data "as follows:

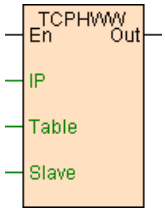
Sequence number	Read data from slave	Write date to slave

1	X0	M100
2	X1	M101
3	V0	V100
4	V50	V102
5	Y4	M0
6	Y5	Y0
7	V60	V200
8	V61	V201

3. TCPHWR, TCPHWW instructions will execute after getting charged from the busbars, according to the definition of "Speedbus read (writte) communication table", and will automaticaly exchange the data with the 2# PLC of IP address 192.168.1.111.

TCPHWW(Speedbus TCP write)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TCPHWW En, IP, Table, Slave, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
IP	Slave equipment IP address	√		
Table	Speedbus write communication table	√		
Slave	Slave equipment address	√		

Parameter	Parameter define	Input	Output	Declare
Out	Communication complete Output		√	

[Instruction function and effect declare]

1. TCPHWW instruction automatically exchanged the data with the slave device according to the definition of "Speedbus write communication table " using Speedbus TCP protocol.

2. Speedbus TCP protocol support disperse . blended data transfer, communication efficient very well.

[Instruction example]

Refer to TCPHWR instruction example.

Interrupt instruction

Interrupt instruction list as follows

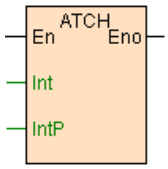
Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
ATCH			Interrupt binding	√	√	√
DTCH			Interrupt release	√	√	√
ENI			Enable interrupt	√	√	√
DISI			Disable interrupt	√	√	√

1. PLC support up to 52 system interrupt source, include pulse output. edge catch . high speed counter and timer interrupt. refer to "system interrupt table ".
2. Interrupt program only use ATCH instruction binding corresponding interrupt, moreover system generate the interrupt executed once ,others any time not execute. refer to "interrupt program".
3. Interrupt program must try to do short and small ,only necessary instruction in interrupt program execute.

ATCH(Interrupt binding)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			ATCH En, Int, IntP	Download

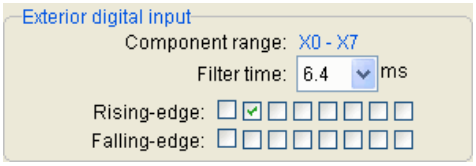
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Int	Interrupt number	√		
IntP	Interrupt program name	√		
Eno	Enable output		√	

[Instruction function and effect declare]

ATCH instruction use for binding IntP interrupt program and Int interrupt number , when system generate Int interrupt, automatic execute binding IntP interrupt program, instruction general executed by edge.

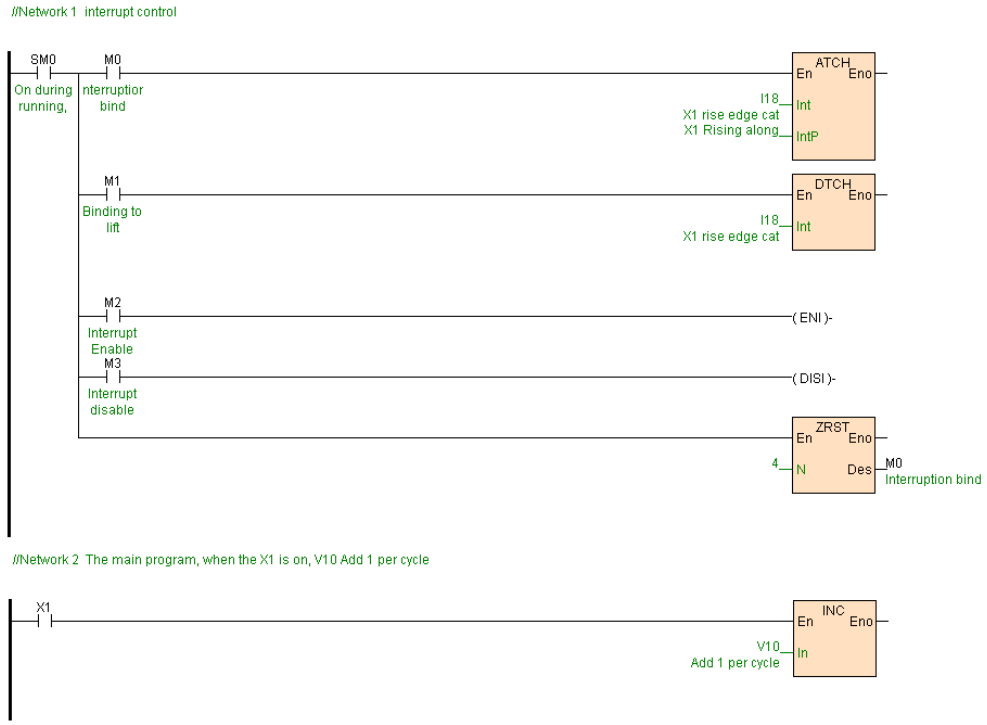
[Instruction example]

1. At "PLC hardware configure " window configure "DI digital Input Parameter" open X1 rising edge catch function .



2. Write interrupt program "X1 rising edge catch", only one instruction,V0 increase 1,as follows.

3. Write main program "ATCH" as follows:

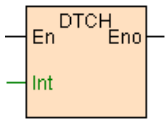


[Program description]

1. M0=ON, use for Interrupt program binding "X1 rising edge catch " and Interrupt number I18(X1 rising edge interrupt), thus when X1 from OFF go to ON ,system generate I18 interrupt moreover call interrupt program "X1 rising edge catch".
2. M1=ON, release interrupt number I18 bind, by now even if system generate I18 interrupt also not execute interrupt program.
3. M2=ON, enable system interrupt, thus when X1 from OFF go to ON, system generate I18 interrupt.
4. M3=ON, disable system interrupt, thus system not generate any interrupt.

DTCH(Interrupt release)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			DTCH En, Int	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Int	Interrupt number	√		
Eno	Enable output		√	

[Instruction function and effect declare]

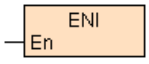
DTCH instruction release Int Interrupt number bind interrupt program, instruction general executed by edge.

[Instruction example]

Refer to ATCH instruction example.

ENI(Enable interrupt)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			ENI En	Download

Parameter	Parameter define	Input	Output	Declare

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

[Instruction function and effect declare]

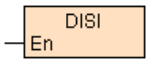
ENI instruction open system interrupt function , system default open interrupt, instruction general executed by edge.

[Instruction example]

Refer to ATCH instruction example.

DISI(Disable interrupt)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(DISI)-		DISI En	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

[Instruction function and effect declare]

DISI instruction shielding system interrupt function , after instruction executed no more generate system interrupt, instruction general executed by edge.

[Instruction example]

Refer to ATCH instruction example.

Program control instruction

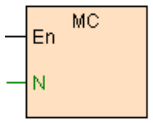
Program control instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
MC			Master control	√	√	√
MCR			Master control clear	√	√	√
FOR			Loop command	√	√	√
NEXT			Loop end	√	√	√
WAIT			Delay wait	√	√	√
CALL			Call subroutine	√	√	√
EXIT			Condition exit	√	√	√
REWD			Scanning time reset	√	√	√
JMPC			Condition jump	√	√	√
LBL			Jump label	√	√	√

MC(Master control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			MC En, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
N	Label	√		

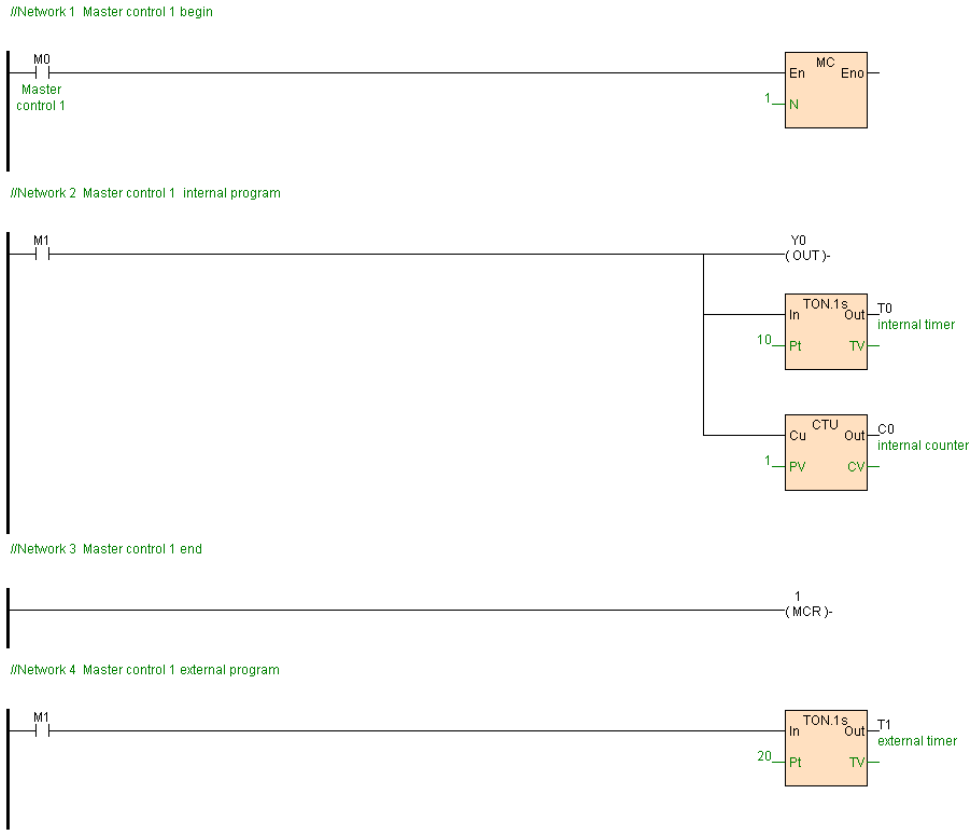
[Instruction function and effect declare]

1. MC is master control start instruction, N is Master control number, when En=ON, the instructions inner master control N (between MC N and MCR N instruction) as usual executed .

2. When En=OFF ,the instructions inner master control N (between MC and MCR instruction) skiped, moreover reset OUT instruction output. timer coil T and current value TV. counter coil Cand current value CV at the same time .

3. MC-MCR instruction support nested structure , maximum may be up to 8 level.

[Instruction example]



[Program description]

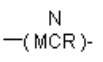
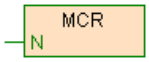
1. Network 1 define master control 1 start , network 3 define master control 1 finish .

2. When M0=ON, master control 1 program (Network 2) normal execution, when M1=ON,Y0=ON, timer T0 start timing , counter C0 increase 1,CV0=1,C0=ON. Master control 1 external Network 4 timer T1 start timing .

3. When M0=OFF, then master control 1 program (Network 2) instruction skip, moreover reset Y0=OFF. T0=OFF. TV0=0. C0=OFF. CV0=0.Master control 1 external Network 4 timer T1 keep on timing.

MCR(Master control clear)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			MCR N	Download

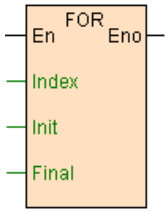
Parameter	Parameter define	Input	Output	Declare
N	Label	√		

[Instruction function and effect declare]

MCR is master control finish instruction ,N is master control number, only pair use with MC instruction .

FOR(Loop command)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FOR En, Index, Init, Final	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Index	Loop index	√		
Init	S tart value	√		
Final	Stop value	√		

Parameter	Parameter define	Input	Output	Declare
Eno	Enable output		√	

[Instruction function and effect declare]

1. FOR instruction and NEXT instruction must be used in pairs , among FOR sign the start of the circulate, NEXT sign the finish of the circulate . The statement between FOR and NEXT named loop body .

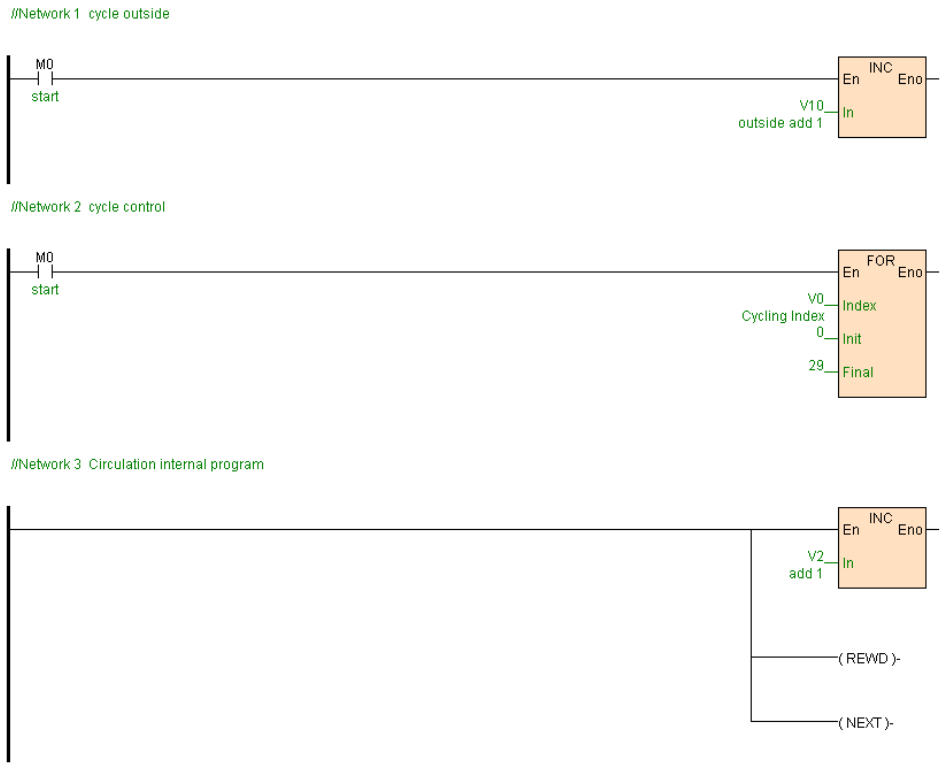
2. FOR/NEXT instruction repeatedly execute the statement in the loop body until reach cycle count, among, count value of the cycle count store to parameter index, moreover Init assigned Index start value, Final assigned Index stop value .

3. A integral loop body include in a loop body , also name loop nesting . FOR/NEXT support nesting, nested depth maximum are 8 level.

4. In loop body , user can modify stop value Final, thus modify the stop condition of the loop .

5. If many cycle count , then the program execute time possible exceed system watchdog timer value , thus possible triggering system watchdog action cause safeguard stop, by now must add REWD instruction in the loop body to reset watchdog.

[Instruction example]



[Program description]

1. M0=ON, each scan cycleV10 increase 1,Network 2 start one loop, loop index V0, cycle count 30(0~29),Network 3 loop body instruction,V2 each increase 1(after executed a loop then add 30), thus V2 must be V10 30 times.
2. REWD instruction reset watchdog ,NEXT is loop end define.

NEXT(Loop end)

Instruction format and parameter specification

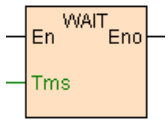
Language	LD	FBD	IL	Program example
Instruction format	—(NEXT)-	NEXT	NEXT	Download

[Instruction function and effect declare]

Define the end of the loop ,only be used in pairs with FOR instruction .

WAIT(Delay wait)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			WAIT En, Tms	

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Tms	Delay time	√		Unit 0.1ms
Eno	Enable output		√	

[Instruction function and effect declare]

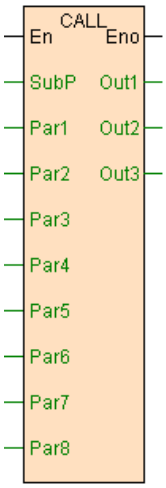
1. WAIT instruction make PLC program execute pause, pause time setup by parameter Tms .

2. WAIT instruction extend the scan cycle, may be trigger the watchdog actuation , must try to do not use.

CALL(Call subroutine)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			CALL En, SubP(0-8 input parameter names)(0-3 output parameter names)	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
SubP	Sub routine name	√		
Par1~8	Input parameter(Can be no parameters)	√		
Eno	Enable output		√	
Out1~3	Output parameter (Can be no parameters)		√	

[Instruction function and effect declare]

1. CALL instruction use for call subroutine. Call subroutine can with parameter or without parameter, use programming software new subroutine window or subroutine attribute window, may be modify and edit the parameter of the subroutine ;When SubP parameter item input the subroutine name, programming

software use for according to the defined parameter of the subroutine the program give the input . output parameter item and parameter name to call the subroutine , instruction input output parameter item enable input data type and the data type of the subroutine defined parameter must be no difference

2. Use CALL call subroutine, if the subroutine called not defined parameter, then without parameter call,CALL instruction Input parameter item and output parameter item omit .to with parameter call subroutine ,parameter therefore transfer and assign in sequence one to one mode.

3. Use CALL instruction reach call subroutine program named main program , subroutine may be called repeat, support nest call subroutine ,maximum nesting 8 level.

4. Subroutine parameter declare:

A. One subroutine maximum may be define 8 input(IN) or input output (IN_OUT) parameter and 3 output(OUT) parameter;

B. Each parameter all have the parameter type(IN-input parameter,OUT-output parameter,IN_OUT—input output parameter) and data type, when use CALL instruction call the subroutine , instruction input output parameter item allow input data type no difference of the data type of the subroutine defined parameter .

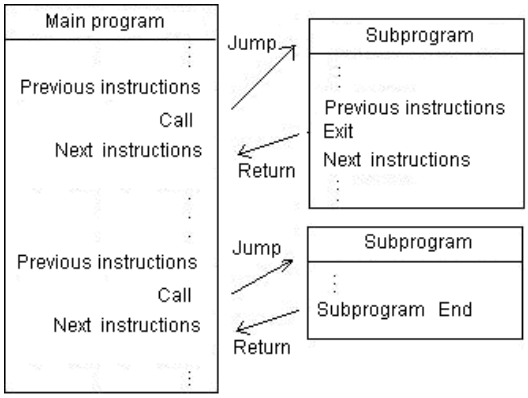
5. subroutineParameter的Parameter typeas follows table 所示:

Parameter type	Mean
IN	Input parameter, by CALL instruction use for parameter data incoming subroutine
IN_OUT	Input output parameter ,first CALL instruction use for parameter data incoming subroutine ,subroutine execut use for again incoming result to CALL instruction parameter
OUT	Output parameter ,subroutine after executed again incoming result to CALL instruction parameter

6. subroutine data type as follows:

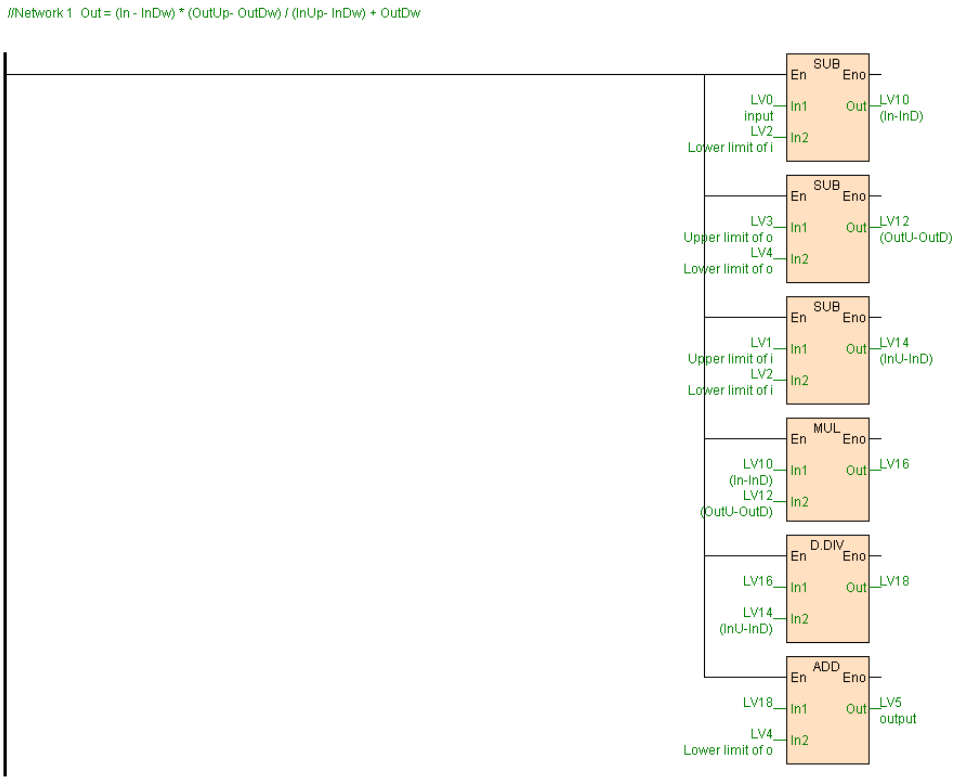
Data type	Mean
BOOL	Also known as bool type ,ON or OFF
WORD	16 bit word, may be data range is -32768~32767
DWORD	Double word,32 bit ,may be data range is -2147483648~2147483647,occupy 2 continuous component
INT	Integer,16 bit ,may be data range is -32768~32767
DINT	Double integer,32 bit ,may be data range is -2147483648~2147483647,occupy 2 continuous component
REAL	Single precision floating point ,32 bit ,occupy 2 continuous component

7. Call subroutine PLC program executing sequence schematic diagram follows:

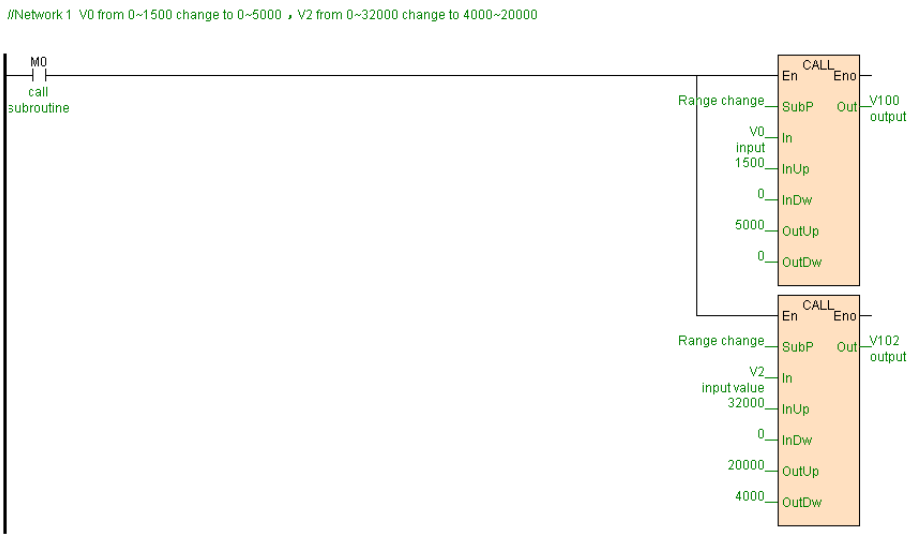


[Instruction example]

1. Program one subroutine "measuring range transfer ", the subroutine achieve SC instruction same function ,as follows.



2. Program main program "CALL"as follows:



[Program description]

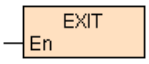
1. Subroutine with 4 input parameter and 1output parameter, calculate Out = (In - InDw) * (OutUp- OutDw) / (InUp- InDw) + OutDw.

2. Main program when M0=ON,2 CALL instruction call the same subroutine "measuring range transfer", because different the parameter transfered , achieve different range transfer function ,as follows:

Parameter	Parameter value	Call subroutine result
InUp	1500	When V0=0, then V100 =0 When V0=100, then V100 =333 When V0=1200, then V100 =4000 When V0=1250, then V100 =4166
InDw	0	
OutUp	5000	
OutDw	0	
InUp	32000	When V2=0, then V102 =4000 When V2=1600, then V102 =4800 When V2=16000, then V102 =12000 When V2=28000, then V102 =18000
InDw	0	
OutUp	20000	
OutDw	4000	

EXIT(Condition exit)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(EXIT)-		EXIT En	

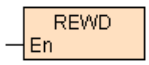
Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

[Instruction function and effect declare]

EXIT is condition exit instruction, only use which need early exit from subroutine or interruption , normal exit subroutine or interruption without add the instruction.

REWD(Scanning time reset)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	—(REWD)-		REWD En	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

[Instruction function and effect declare]

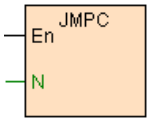
1. REWD instruction use for reset watchdog.

2. When program need loop execution at some condition , thus possible triggering system watchdog actuation cause protect stop ,by now must add REWD instruction within the loop body to reset watchdog.

3. REWD instruction make watchdog out of action, must try to do not use.

JMPC(Condition jump)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			JMPC En, N	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
N	Jump label	√		

[Instruction function and effect declare]

1. JMPC is with condition jump instruction, when En=ON, program use for jump to label N appoint position next instruction continue execution, if En=OFF then do not jump ,use original instruction sequence execution.

2. N Jump label must use LBL instruction define, if label not exist, then the instruction not execute.

[Instruction example]

[Program description]

1. Network 1 INC instruction from busbar get electricity , each scan cycle use for V0 increase 1.

2. When M0=OFF,JMPC instruction not execute,Network 3 INC instructionfrom busbar get electricity ,each scan cycle use for V2 increase 1.

3. When M0=ON, JMPC instruction execution , jump to label1 back continue execution,by now V2 not increase 1.

LBL(Jump label)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format	N —(LBL)-	LBL N	LBL N	Download

Parameter	Parameter define	Input	Output	Declare
N	Jump label	√		

[Instruction function and effect declare]

LBL instruction use for define jump label, to JMPC instruction use .

Special Function instruction

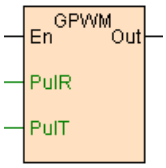
Special function instruction list as follows

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
GPWM			General pulse width modulation	√	√	√
FTC			Fuzzy temperature control	√	√	√
PID			PID control	√	√	√
HAL		D.HAL	Upper limit alarm	√	√	√
LAL		D.LAL	Lower limit alarm	√	√	√

Instruction name	8 bit model	32 bit model	Instruction function	Support language		
				LD	FBD	IL
LIM		D.LIM	Range limitation	√	√	√
SC		D.SC	Linear conversion	√	√	√
VC			Valve control	√	√	√
TTC			Temperature curve control	√	√	√
APID			Self-tuning PID	√	√	√

GPWM(General pulse width modulation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			GPWM En, PulR, PulT, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PulR	Pulse duty factor	√		Unit 0.1%, range 0~1000
PulT	Pulse output period	√		Unit ms
Out	Pulse width modulation output		√	

[Instruction function and effect declare]

1. GPWM instruction use for common transistor output point Yn output pulse which variable pulse period . duty facto.

2. Maximum pulse period is 32767, when PulT≤0 not output pulse.

3. When PulT＞0, if PulR＞0 moreover PulR＜1000 ,Out output pulse of duty factor is PulR. period is PulT, if PulR=0 then Out output low level, if PulR≥1000 then Out output high level singal.

4. PulR. PulT value cn be modified real time.

[Instruction example]

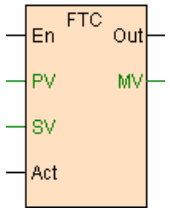
[Program description]

1. When M0=ON, from Y20 output the pulse which duty factor is 30%. period is 30ms.

2. When M0=OFF, pulse output stop.

FTC(Fuzzy temperature control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			FTC En, PV, SV, Out, MV	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
PV	Measure value	√		Unit 0.1℃
SV	Set value	√		Unit 0.1℃
Act	Co ntrol mode	√		0-reaction,1-direct action, others invalid.
Out	Pulse width modulation output		√	
MV	Control output		√	

[Instruction function and effect declare]

1. FTC instruction use for fuzzy temperature control special purpose instruction, the instruction need not set any parameter completely automatic regulate output ,simpleness use .

2. Act is control mode,Act=0 is reaction(PV increase ,MV output trend decrease, normal use for heat control),Act=1 is direct action(PV increase,MV output trend increase,normal use for cool control).

3. Instruction have 2 output mode, Out pulse width modulation output(frequency 1Hz),MV is control output(range 0~1000).

4. En is instruction enable item , when En=ON instruction execute ;When En=OFF instruction stop execute ,Out=OFF,MV=0.

5. Many external factor will influence temperature control effective , such as: sensor accuracy . sensor install position . heat function big or small etc., if FTC

temperature control effective unsatisfactory ,may be use PID or program experience control arithmetic oneself .

[Note]: When the FTC instruction first run, it will automatically control the output Out=On, MV=1000, this process is used to calculate the fuzzy factor, and should be used when PV is in room temperature,and the calculation process will take less than 1.5 minutes. The calculated fuzzy coefficients will be stored in the instruction, and it wil not be lost when the plc is shut down. If the program is re-downloaded,then the FTC instruction which is again in the first run will automatically calculate the fuzzy factor.

[Instruction example]

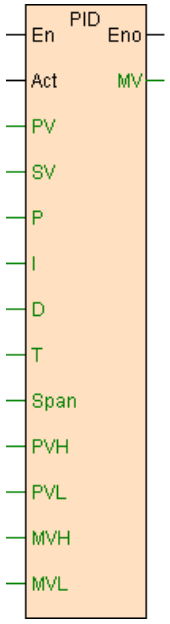
[Program description]

When M0=ON,FTC instruction execute, according to deviation of AI0 and V1000 use fuzzy control arithmetic to control output .When M0=OFF, then Y3=OFF,V100=0.

PID(PID control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			PID En, Act, PV, SV, P, I, D, T, Span, PVH, PVL, MVH, MVL, MV	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Act	Co ntrol mode	√		0-reaction,1-direct action, others invalid.
PV	Measure value	√		
SV	Set value	√		
P	Proportionnality coefficient	√		Unit %
I	Integral time	√		Unit 10ms
D	Deri vative time	√		Unit 10ms
T	Sampling period	√		Unit 10ms

Parameter	Parameter define	Input	Output	Declare
Span	Dead band	√		
PVH	PV m easuring upper limit	√		
PVL	PV measuring lower limit	√		
MVH	MV measuring upper limit	√		
MVL	MV measuring lower limit	√		
Eno	Enable output		√	
MV	Control output		√	

[Instruction function and effect declare]

1. Act is control mode,Act=0 is reaction(PV increase ,MV output trend decrease, normal use for heat control),Act=1 is direct action(PV increase,MV output trend increase,normal use for cool control).

2. En is instruction enable item, when En=ON instruction execute; When En=OFF instruction stop execute ,MV=0.

3. Many external factor will influence temperature control effective ,need time and again adjust P. I. D these 3 parameters to meet the control object, also can program experience control arithmetic oneself .

4. P. I. D parameters direct influence reality control effective stand and fail , below are the three parameters function in PID control :

- Proportionnality coefficient (P) regulating effect : is a percentage data. is response system deviation in proportion ,system in case of appear deviation,

proportional control immediately generate regulating effect to reduce deviation. Proportional action big ,may be accelerate regulate, reduce deviation, but oversize proportion , make system stability descend , even cause system unstable.

- Integral time (I) regulating effect : make system eliminate steady state error, improve indifference .Because have error , integral control as for as proceed, until have not error, integral control stop , integral control output a constant .Integral functional strong and weak depend on integral time I,I big ,integral functional stronger, regulate response slower .Otherwise I small then integral functional weaker, regulate response faster, add integral control make system stability descend, dynamic response slower .Integral functional common use with others two regulating mode combine, make up PI regulator or PID regulator .When Integral time set to 0, no integral.
- Derivative time (D) regulating effect : differential functional reflect system deviation signal rate of change , have foreseeability , can foresee deviation change trend , so can generate ahead of control function, before deviation not yet production , already be differential regulating effect eliminate. thus, may be improve system dynamic response . At derivative time suitable selected , may be reduce overshoot, reduce regulating time. Differential functional amplified action the noise interference , that is the overflow differential functional ,harmful to system anti-interference .Moreover ,differential corresponsive rate of change, when Input unchanged, differential functional output is zero. differential functional can not alone used , must use with others two regulating mode combine, make up PI regulator or PID regulator .When derivative time set to 0, no iderivative.

5. Incremental PID arithmetic :

$\triangle u(n) = Kp * \triangle e(n) + Ki * e(n) + Kd (\triangle e(n) - \triangle e(n-1)))$

$= Kp * \triangle e(n) + Ki * e(n) + Kd * (e(n) – 2 * e(n-1) + e(n-2))$

$u(n) = u(n-1) + \triangle u(n)$ reaction

$u(n) = u(n-1) - \triangle u(n)$ direct action

Among:u(n):MV Output

u(n-1):last output

$\triangle u(n)$: output incremental

Kp=P / 100:proportionnality coefficient

Ki=Kp * T / I: integral coefficient

Kd=D / T: derivative coefficient

e(n)=(SV - PV): deviation

e(n-1):last deviation

e(n-2):last last deviation

[Instruction example]

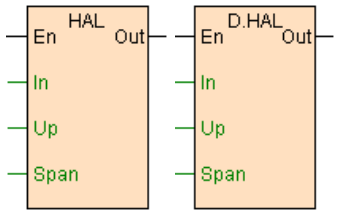
[Program description]

1. When M0=ON,PID instruction execute, according to AI0 and V1000 deviation get through PID arithmetic Control output.When M0=OFF, instruction not execute,AQ0=0.

2. If need use for PID instruction analog output turn into digital duty factor output, use GPWM instruction ,PulT=5000 express pulse period 5 second, Note PID instruction MV output range setup to 0~1000.

HAL. D.HAL(Upper limit alarm)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			HAL En, In, Up, Span, Out D.HAL En, In, Up, Span, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Up	Upper limit alarm value	√		
Span	Dead band	√		
Out	Status output		√	

[Instruction function and effect declare]

HAL instruction is upper limit alarm instruction. If In>(Up+Span), then Out=ON; If In<(Up-Span), then Out=OFF, when In and Up deviation value less than equal to dead band value span,Out remain unchanged.

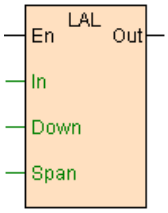
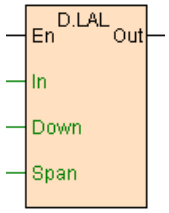
[Instruction example]

[Program description]

When M0=ON start upper limit alarm function .If AI0 is temperature measure value ,V1000=3000(300°),V1001=20(2°) , when AI0>3020(302°) Y3=ON; When AI0<2980(298°) Y3=OFF.

LAL. D.LAL(Lower limit alarm)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			LAL En, In, Down, Span, Out D.LAL En, In, Down, Span, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		

Parameter	Parameter define	Input	Output	Declare
In	Input	√		
Down	Lower limit alarm value	√		
Span	Dead band	√		
Out	Status output		√	

[Instruction function and effect declare]

LAL instruction is lower limit alarm instruction. If In<(Down-Span), then Out=ON; If In>(Down+Span), then Out=OFF, when In and Down deviation value less than equal to dead band value span,Out remain unchanged.

[Instruction example]

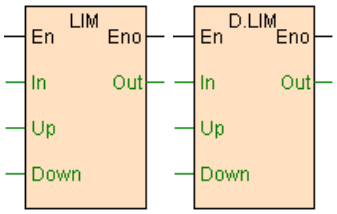
[Program description]

When M0=ON start lower limit alarm function .If AI0 is temperature measure value ,V1000=300(30°),V1001=20(2°) , when AI0<280(28°) Y3=ON; When AI0>320(32°) Y3=OFF.

LIM. D.LIM(Range limitation)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			LIM En, In, Up, Down, Out D.LIM En, In, Up, Down, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
Up	U pper limit	√		
Down	Lower limit	√		
Eno	Enable output		√	
Out	Output		√	

[Instruction function and effect declare]

1. LIM is range limitation instruction, if In>Up then Out=Up, if Down≤In≤Up then Out=In, if In<Down then Out=Down.

2. if Up≤Down, then the instruction not execute.

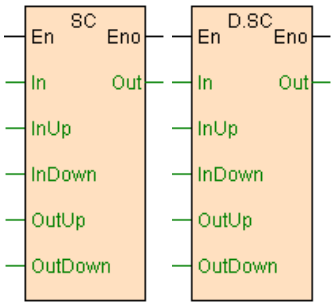
[Instruction example]

[Program description]

When M0=ON start range limitation. If V1000=800,V1001=20, when V0<20 AQ0=20;When V0>800 AQ0=800, when 20≤V0≤800 AQ0=V0.

SC. D.SC(Linear conversion)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
16. 32 bit Instruction format			SC En, In, InUp, InDown, OutUp, OutDown, Out D.SC En, In, InUp, InDown, OutUp, OutDown, Out	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
In	Input	√		
InUp	Input upper limit	√		
InDown	Input lower limit	√		
OutUp	Output upper limit	√		
OutDown	Output lower limit	√		
Eno	Enable output		√	
Out	Transfer output		√	

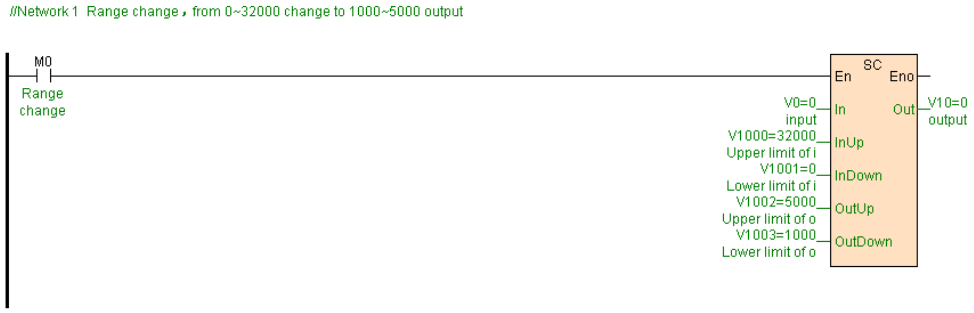
[Instruction function and effect declare]

1. SC instruction according to input. output bound In input proceed linear conversion transfer output to out.

2. Linear conversion formula : out = (In - InDown) * (OutUp- OutDown) / (InUp-InDown) + OutDown.

3. if InUp=InDown, then the instruction not execute.

[Instruction example]



[Program description]

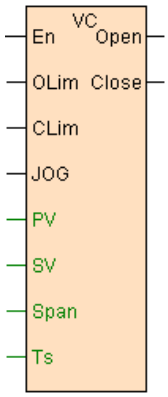
When M0=ON, SC V0 transfer output to V10,as follows table .

Component	Initial value	Transfer result
V1000	32000	When V0=0, then V10 =1000 When V0=100, then V10 =1012 When V0=12000, then V10 =2500 When V0=28000, then V10 =4500
V1001	0	
V1002	5000	
V1003	1000	

VC(Valve control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example

Language	LD	FBD	IL	Program example
Instruction format			VC En, OLim, CLim, JOG, PV, SV, Span, Ts, Open, Close	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
OLim	Valve open limit	√		
CLim	Valve close limit	√		
JOG	Valve jog	√		
PV	Measure value	√		
SV	Set value	√		
Span	Dead band	√		
Ts	Motor running time	√		Unit 秒
Open	Valve open output		√	
Close	Valve close output		√	

[Instruction function and effect declare]

1. VC is specific to valve control special purpose instruction, like one valve controler.

2. En is instruction enable item , when En=ON instruction execute, when En=OFF instruction stop working,Open=OFF,Close=OFF.

3. VC instruction according to valve current opening PV and valve set opening SV compare, if both difference value greater than dead band span, then action to Valve control output (open or close), each the motor running time longest is Ts(second).

4. Open valve open control:when (SV - PV)> Span ,Open=ON valve open output; When (SV - PV)≤ Span ,Open=OFF valve open stop .In control valve process, if valve open limit Olim=ON ,Open=OFF.

5. Close valve close control:when (PV - SV)> Span ,Close=ON valve close output; When (PV - SV)≤ Span ,Close=OFF valve close stop .In control valve process, if valve close limit Clim=ON,Close=OFF.

6. Valve jog control:when JOG from OFF go to ON, use for according to SV. PV. Span three current value make out open or close the valve(control according to and on-off control the same as), each jogthe motor running time longest is Ts(second).

7. If Ts＞0,valve guard time valid ,In Ts valve not reach the preset position (Set value), on-off action stop , in case valve appear mechanical error and yet damage the motor which running time too long.

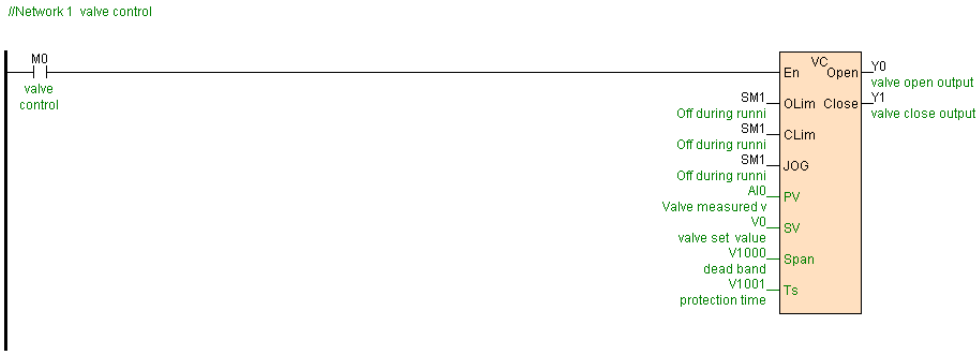
8. If Ts=0, express time-out output, note the valve lose protect function .

9. If Span＜0 or Ts＜0, then instruction not execute.

10. If valve without limit protection output,Olim. Clim may be use SM1 input, express not use limit protect function .

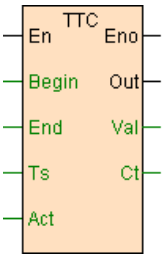
11. VC instruction can cooperate PID etc. instruction realize more complex control function .

[Instruction example]



TTC(Temperature curve control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			TTC En, Begin, End, Ts, Act, Out, Val, Ct	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Begin	Start point value	√		

Parameter	Parameter define	Input	Output	Declare
End	End point value	√		
Ts	Control time	√		Ts＞0 unit is second,Ts＜0 unit is minute
Act	Control mode	√		
Eno	Enable output		√	
Out	Status output		√	
Val	Output value		√	
Ct	Current time		√	unit is second,occupy 2 continuous component

[Instruction function and effect declare]

1. TTC instruction use for control data value in given time from start point value (Begin) to end point value (End) changing process, that is ratio control.

2. Ts is control time,Ts＞0 unit is second,Ts＜0 unit is minute,Ts=0 then invalid instruction not execute, without unit is second or minute instruction all each second arithmetic utput once.

3. When Act=0 (restart mode), at TTC instruction start ,reset Out=OFF, then from Begin start timing control output Val.

4. When Act=1 (memory mode), at TTC instruction start , reset Out=OFF moreover read current value Val, according the condition to execute of Val in Begin~End interzone.

1). Begin<End (ascent stage), if Val≤Begin then from start point value (Begin) start timing control output Val; if Val≥End then timing out ,Val invariant

moreover Out=ON; if Val at Begin~End interzone then from Val start timing control output Val.

2). Begin>End (decline stage), if Val≥Begin then from Begin start timing control output Val; if Val≤End then timing out ,Val invariant moreover Out=ON; if Val at Begin~Endinterzone then from Val start timing control output Val.

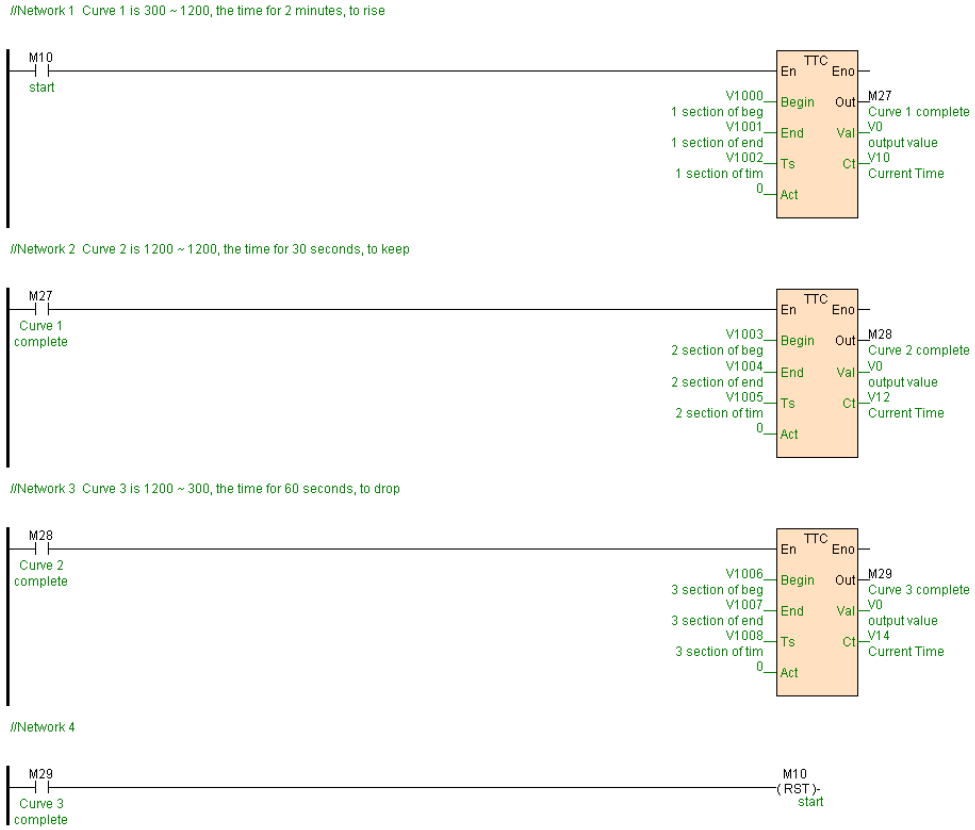
3). Begin=End (maintain stage), if Val=Begin then from start point value (Begin) start timing control output Val; if Val≠Begin then timing out ,Val invariant moreover Out=ON.

5. Instruction in processing, parameters Begin. End. Ts can not be modified (modified not take effect at real time, need re-execute the instruction).

6. Use TTC instruction may be generate mulit-segment curve, may be cooperate PID etc. others instruction realize more complex control function .

7. At Act=1(memory mode),Val output item should use preserv component .

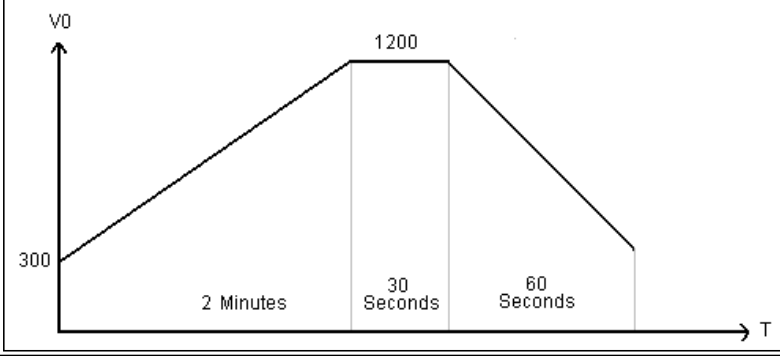
[Instruction example]



[Program description]

If "3 segment curve initial data" as follows table .

Component	**Initial value**	**Output curve**
V1000	300	
V1001	1200	
V1002	-2	
V1003	1200	
V1004	1200	
V1005	30	
V1006	1200	
V1007	300	
V1008	60	



1. When M10=ON, start first segment curve , from 300 to 1200 take 2 minute, complete M27=ON.

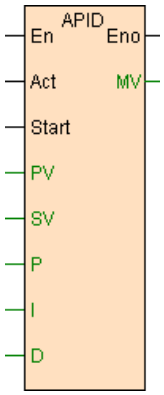
2. When M27=ON, start second segment curve ,from 1200to 1200 take30 second , complete M28=ON.

3. When M28=ON, start third segment curve ,from 1200 to 300 take 60 second, complete M29=ON.

4. When M29=ON,reset M10=OFF.

APID(Self-tuning PID control)

Instruction format and parameter specification

Language	LD	FBD	IL	Program example
Instruction format			APID En, Act, Start, PV, SV, P, I, D, MV	Download

Parameter	Parameter define	Input	Output	Declare
En	Enable	√		
Act	Control mode	√		0-reaction,1-direct action, others invalid.
Start	Start Self-tuning	√		

Parameter	Parameter define	Input	Output	Declare
PV	Measure value	√		
SV	Set value	√		
P	Proportionnality coefficient	√		Unit %
I	Integral time	√		Unit 10ms
D	Deri vative time	√		Unit 10ms
Eno	Enable output		√	
MV	Control output		√	

[Instruction function and effect declare]

1. Act is control mode,Act=0 is reaction(PV increase ,MV output trend decrease, normal use for heat control),Act=1 is direct action(PV increase,MV output trend increase,normal use for cool control).

2. En is instruction enable item, when En=ON instruction execute; When En=OFF instruction stop execute ,MV=0.

3. When Start=ON, it starts the auto-tuning, after the finishing of the self-tuning the instruction will automatically write the calculated P. I. D parameter value to the concerning register corresponding to the terminal. If the parameters obtained by the auto-tuning is not satisfactory, you can reactivate "Start" to carry out the auto-tuning again, or manually input the parameter. There is much external factors affecting the control effect, it's necessary to repeatedly adjust the P. I. D

parameters to meet the requirements of the control object,besides, the experienced technican can also adjust the parameters themselves.

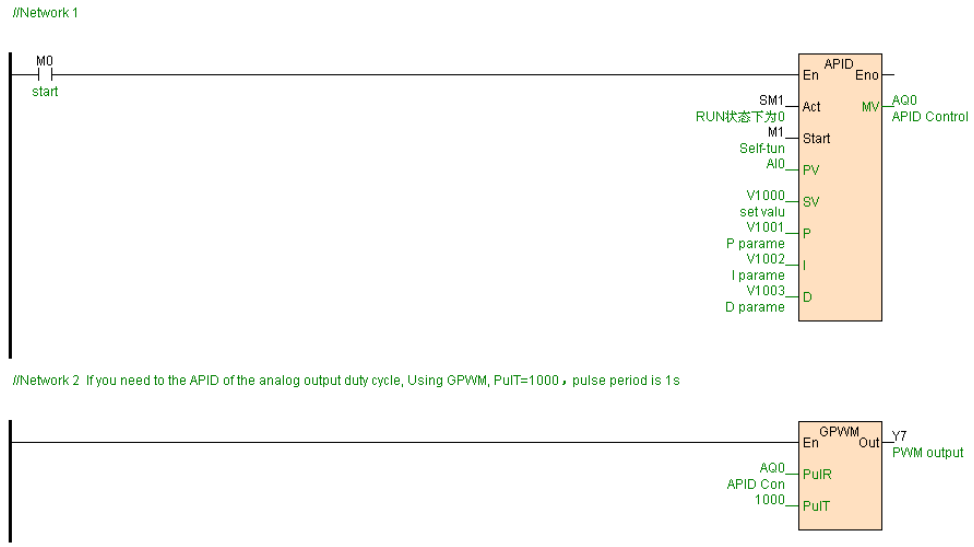
4. P. I. D parameters direct influence reality control effective stand and fail , below are the three parameters function in PID control :

- Proportionnality coefficient (P) regulating effect : is a percentage data. is response system deviation in proportion ,system in case of appear deviation, proportional control immediately generate regulating effect to reduce deviation. Proportional action big ,may be accelerate regulate, reduce deviation, but oversize proportion , make system stability descend , even cause system unstable.
- Integral time (I) regulating effect : make system eliminate steady state error, improve indifference .Because have error , integral control as for as proceed, until have not error, integral control stop , integral control output a constant .Integral functional strong and weak depend on integral time I,I big ,integral functional stronger, regulate response slower .Otherwise I small then integral functional weaker, regulate response faster, add integral control make system stability descend, dynamic response slower .Integral functional common use with others two regulating mode combine, make up PI regulator or PID regulator .When Integral time set to 0, no integral.
- Derivative time (D) regulating effect : differential functional reflect system deviation signal rate of change , have foreseeability , can foresee deviation change trend , so can generate ahead of control function, before deviation not yet production , already be differential regulating effect eliminate. thus, may be improve system dynamic response . At derivative time suitable selected , may be reduce overshoot, reduce regulating time. Differential functional amplified

action the noise interference , that is the overflow differential functional ,harmful to system anti-interference .Moreover ,differential corresponsive rate of change, when Input unchanged, differential functional output is zero. differential functional can not alone used , must use with others two regulating mode combine, make up PI regulator or PID regulator .When derivative time set to 0, no iderivative.

[Note]: Auto-tuning should be performed when PV is in the room temperature and the entire auto-tuning process takes less than 1.5 minutes. The PID coefficient obtained from the auto-tuning will be automatically written to the corresponding P、I、D terminals, so the P、I、D terminals of the instruction should use latched registers.

[Instruction example]



[Program description]

1. When M0=ON,APID instruction execute, according to AI0 and V1000 deviation get through APID arithmetic Control output.When M0=OFF, instruction not execute,AQ0=0.

2. If need use for APID instruction analog output turn into digital duty factor output, use GPWM instruction ,PulT=1000 express pulse period 1 second.

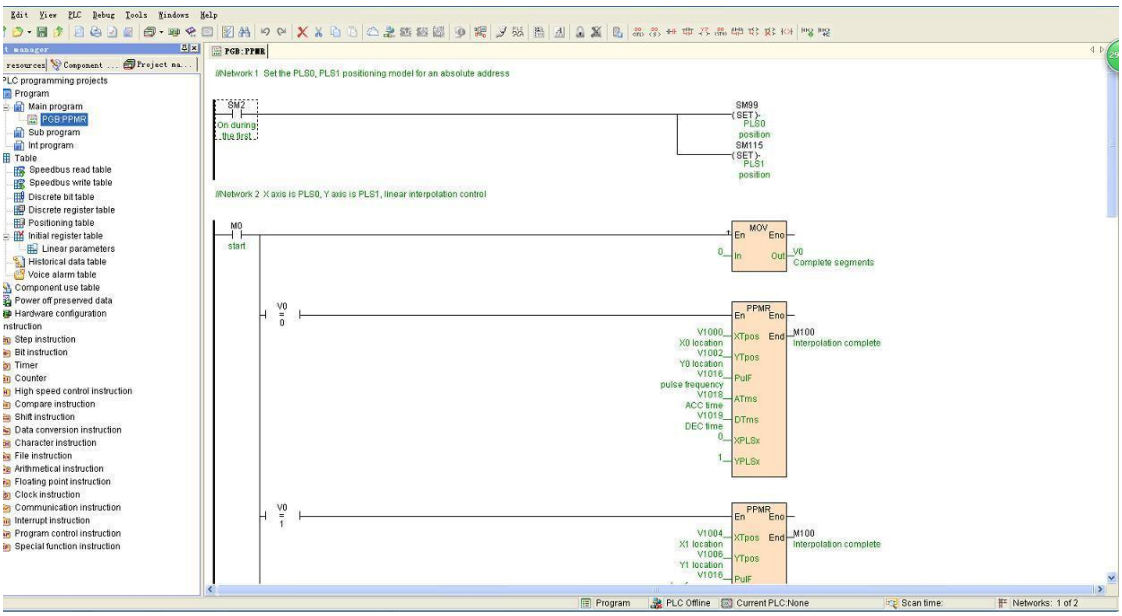
Programming operation manual

This section introduce operation of PLC programming software .

Programming environment

Overview

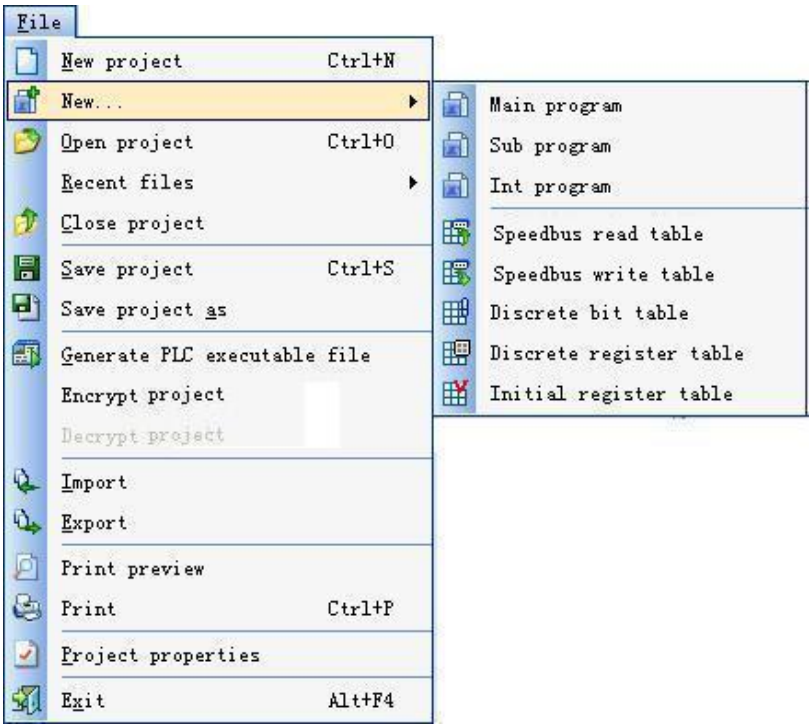
Programming software main interface include: menu. tools bar . project management. working area and status bar etc..



Menu

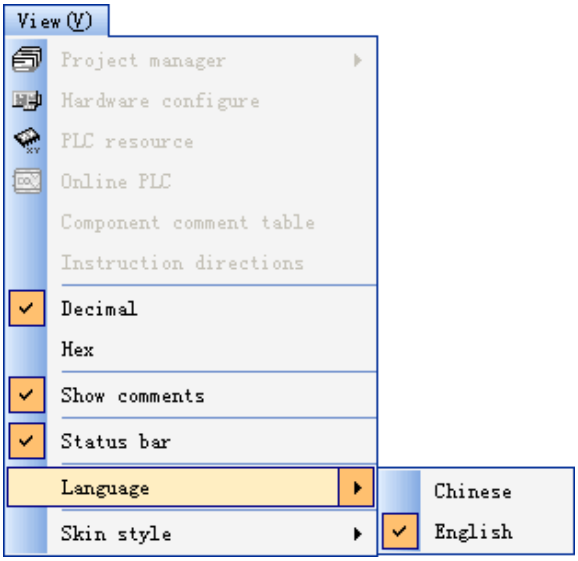
Menu include all function of programming software , compose by main menu and multistage submenu.

1. "File" submenu : use for mange the related operations of the program project file.

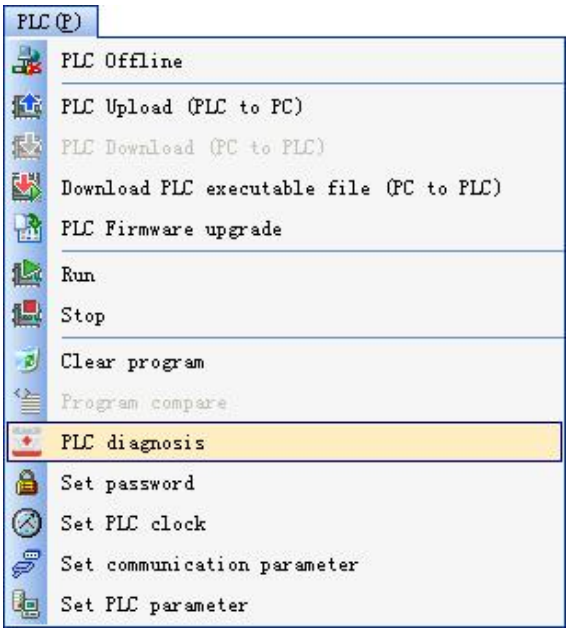


2. "Edit" submenu : use for edit the program related operations.

3. "Search" submenu : use for open all kinds of resource window, change language. display mode etc..

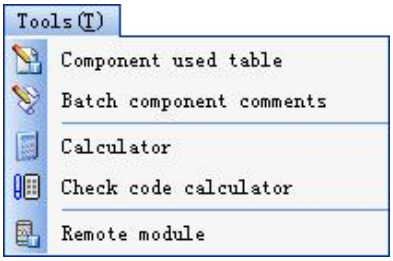


4. "PLC" submenu : use for all kinds of control PLC.

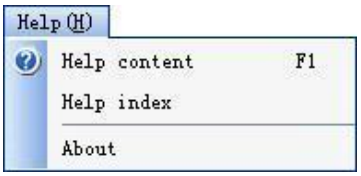


5. "Debug" submenu : supply a group related tools to convenience debug the program.

6. "Tools" submenu : supply a group convenience tools, as: manage the remote module.



7. "Help" submenu : supply online Help function.



Tools bar

Tools bar include PLC programming software common function , so that user can program quickly. When move mouse onto tools bar button the button name will be displayed.

1. Standard tools bar :

2. LD language tools bar :

3. FBD language tools bar :

4. Simulator tools bar :

5. "Interpolation simulator" window tools bar :

6. "Remote module" window tools bar :

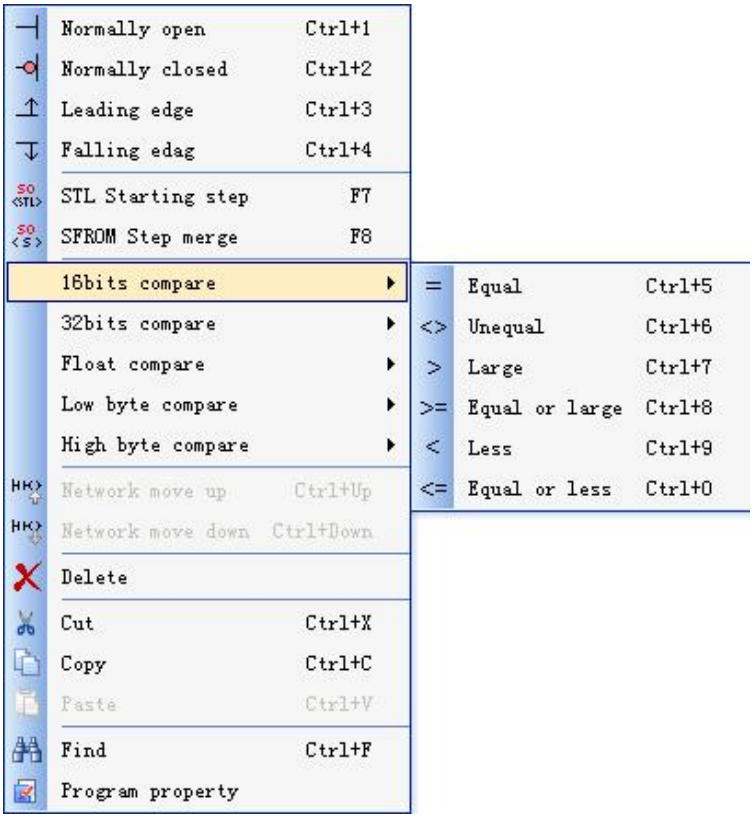
7. "Batch component comment" window tools bar :

8. Print previewtools bar :

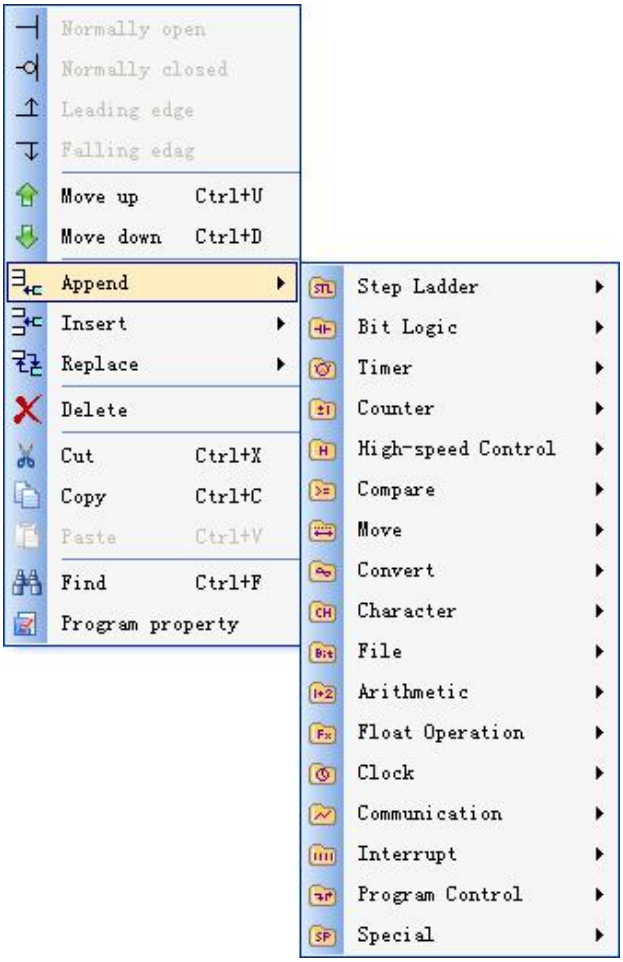
Right click menu

For improve program efficiency, programming software use vast right click menu. Via mouse right button click to popup right click menu.

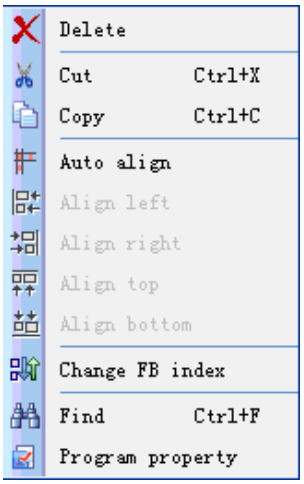
1. At LD language programming environment , pitch up switch right click menu:



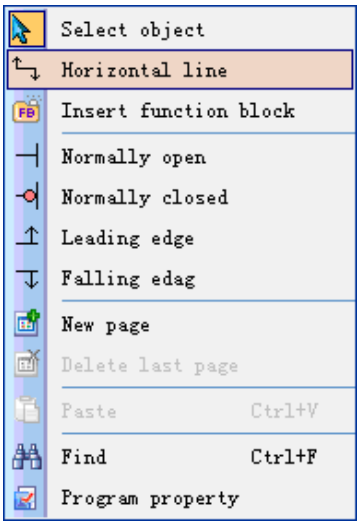
2. At LD language programming environment , pitch up instruction or instruction item right click menu:



3. At FBD language programming environment , pitch up instruction right click menu:

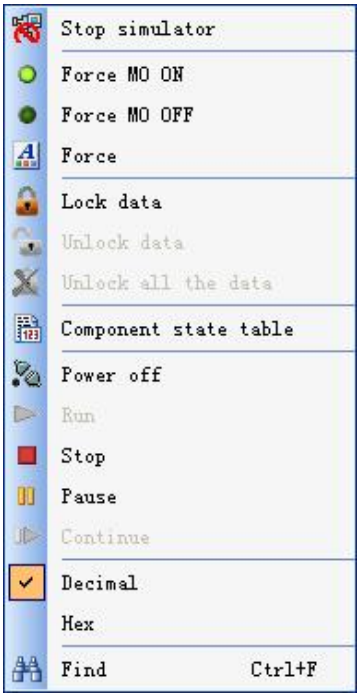


4. At FBD language programming environment , pitch up instruction item right click menu:

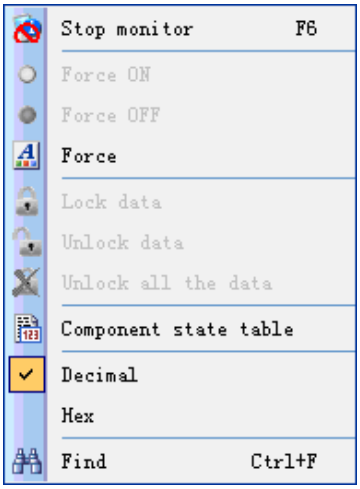


5. At IL language programming environment right click menu:

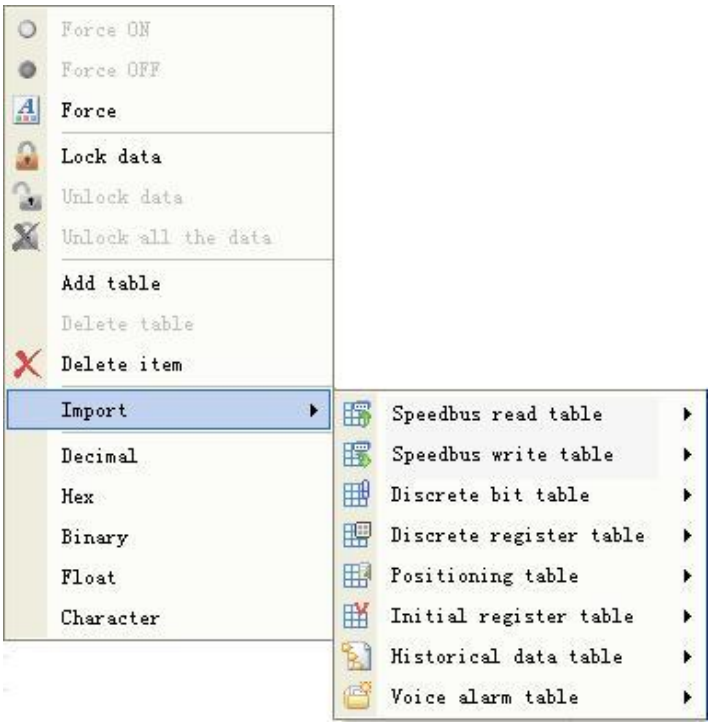
6. At simulation environment right click menu:



7. At online monitoring environment right click menu:



8. At "Component status table" window right click menu:



9. "PLC hardware configuration" window. "Project manager" window. "Online PLC" window etc. also support right click menu.

Shortcut key

PLC programming software supply abundant shortcut key, during user write the program, may be use shortcut key improve programming speed and efficiency, make the programming operation easy and efficient ,shortcut key listing as follows:

Category	Shortcut key	Function and purpose
menu operation	Alt + F	Open "File" menu
	Alt + E	Open "Edit" menu

	Alt + V	Open "Search" menu
	Alt + P	Open "PLC" menu
	Alt + D	Open "Debug" menu
	Alt + T	Open "Tools" menu
	Alt + W	Open " window" menu
	Alt + H	Open "Help" menu
Program project operation	Ctrl + N	New program project
	Ctrl + O	Open program project
	Ctrl + S	Save program project
	Ctrl + P	Print
	Alt + F4	Exit programming software
Edit	Ctrl + Z	Recover
	Ctrl + R	Redo
	Ctrl + X	Cut

	Ctrl + C	Copy
	Ctrl + V	Paste
	Ctrl + G	Go to …
	Ctrl + F	Search
	F3	Search next
	Ctrl + A	Select all
Debug	F5	Start "Online monitor"
	F6	Stop "Online monitor "
Program edit	Ctrl + 1	LD program :normal open
	Ctrl + 2	LD program :normal close
	Ctrl + 3	LD program :rising edge switch
	Ctrl + 4	LD program :failling edge switch
	Ctrl + 5	LD program :equal to switch(16 bit compare)

	Ctrl + 6	LD program :not equal to switch(16 bit compare)
	Ctrl + 7	LD program :greater than switch(16 bit compare)
	Ctrl + 8	LD program :greater than or equal to switch(16 bit compare)
	Ctrl + 9	LD program :less than switch(16 bit compare)
	Ctrl + 0	LD program :less than or equal to switch(16 bit compare)
	Ctrl + Alt + 5	LD program :equal to switch(32 bit compare)
	Ctrl + Alt + 6	LD program :not equal to switch(32 bit compare)
	Ctrl + Alt + 7	LD program :greater than switch(32 bit compare)
	Ctrl + Alt + 8	LD program :greater than or equal to switch(32 bit compare)
	Ctrl + Alt + 9	LD program :less than switch(32 bit compare)

	Ctrl + Alt + 0	LD program :less than or equal to switch(32 bit compare)
	Ctrl + Shift + 5	LD program :equal to switch(floating point number compare)
	Ctrl + Shift + 6	LD program :not equal to switch(floating point number compare)
	Ctrl + Shift + 7	LD program :greater than switch(floating point number compare)
	Ctrl + Shift + 8	LD program :greater than or equal to switch(floating point number compare)
	Ctrl + Shift + 9	LD program :less than switch(floating point number compare)
	Ctrl + Shift + 0	LD program :less than or equal to switch(floating point number compare)
	Alt + 5	LD program :equal to switch(low byte compare)

	Alt + 6	LD program :not equal to switch(low byte compare)
	Alt + 7	LD program :greater than switch(low byte compare)
	Alt + 8	LD program :greater than or equal to switch(low byte compare)
	Alt + 9	LD program :less than switch(low byte compare)
	Alt + 0	LD program :less than or equal to switch(low byte compare)
	Alt + Shift + 5	LD program :equal to switch(high byte compare)
	Alt + Shift + 6	LD program :not equal to switch(high byte compare)
	Alt + Shift + 7	LD program :greater than switch(high byte compare)
	Alt + Shift + 8	LD program :greater than or equal to switch(high byte compare)
	Alt + Shift + 9	LD program :less than switch(high byte compare)

	Alt + Shift + 0	LD program :less than or equal to switch(high byte compare)
	Ctrl + U	LD. IL program :application instruction move up
	Ctrl + D	LD. IL program :application instruction move down
	Ctrl + Up	LD program :network move up
	Ctrl + Down	LD program :network move down
	F7	LD program :STL step start
	F8	LD program :SFROM step combine
	F9	LD program :series switch
	F10	LD program :parallel switch
	F11	LD program :output coil
	F12	LD program :STO step transfer instruction
	Ctrl+B	LD program :output branch

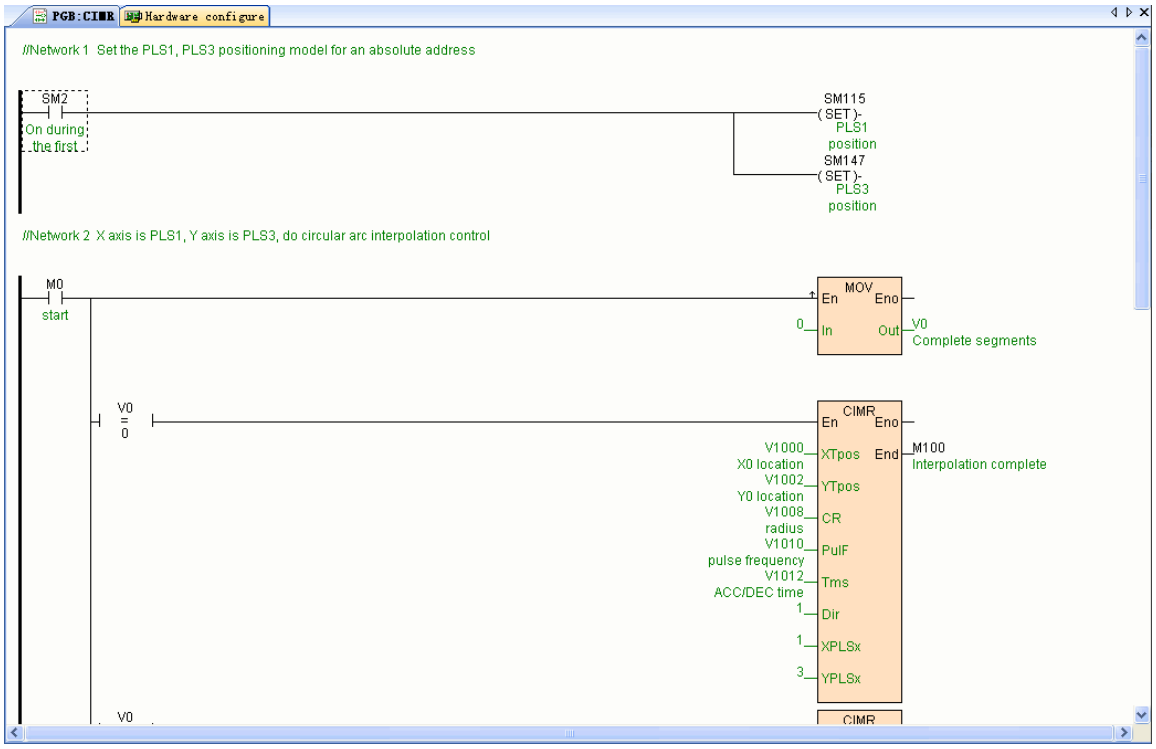
	Ctrl+H	LD program :delete output branch
	Ctrl+L	LD program :add network
	Ctrl+I	LD program :insert network
Online Help	F1	Open online help
Common use	ESC	Cancel operation
	DEL	Delete pitch up object

Status bar

Status bar use for indicate programming software current status and display relate operation prompt message, via menu [Search/status bar]open or close it.

Working area

User working area is use for write control program . configure PLC hardware information.At the area, user may be edit current program or PLC hardware configure information.

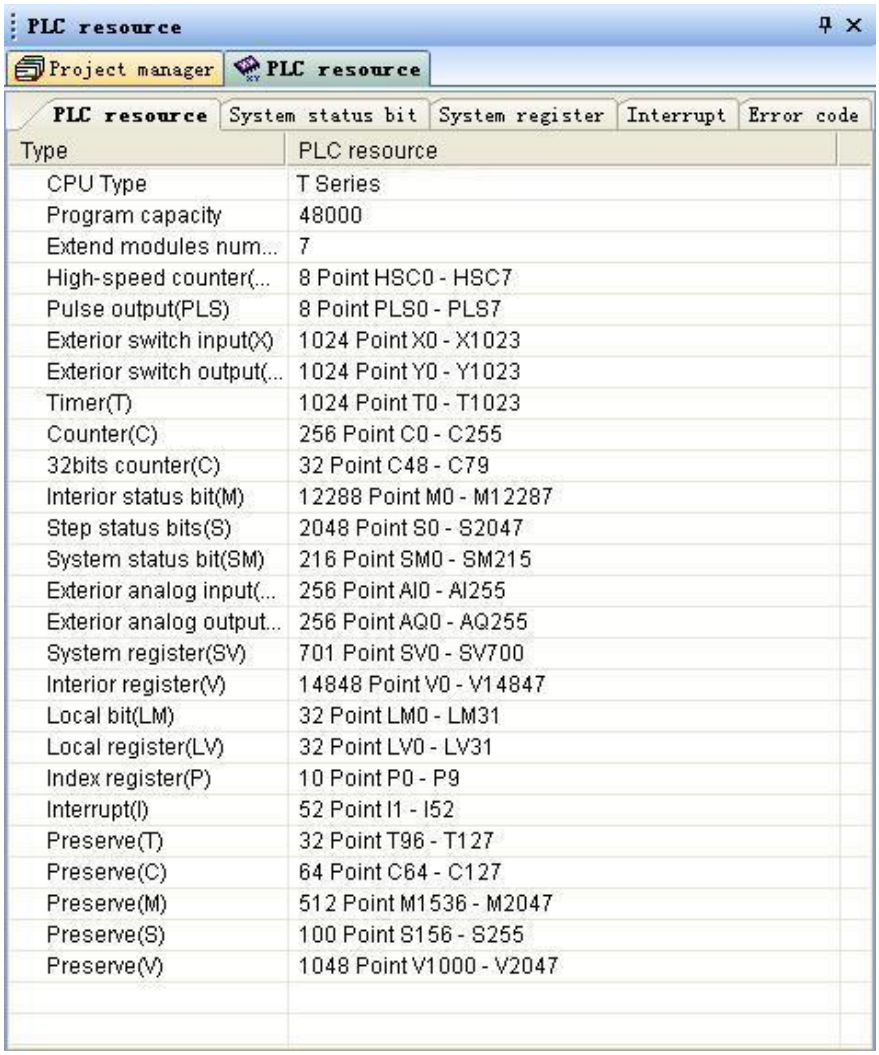


Project manager

Via menu [Search/project manager] open "Project manager" page, it use tree form structure display total project all content: project name. main program block. sub program . interrupt program . table. PLC hardware configure. all kinds of instruction etc.. Project manager support right click menu, convenience user manage operate the project .

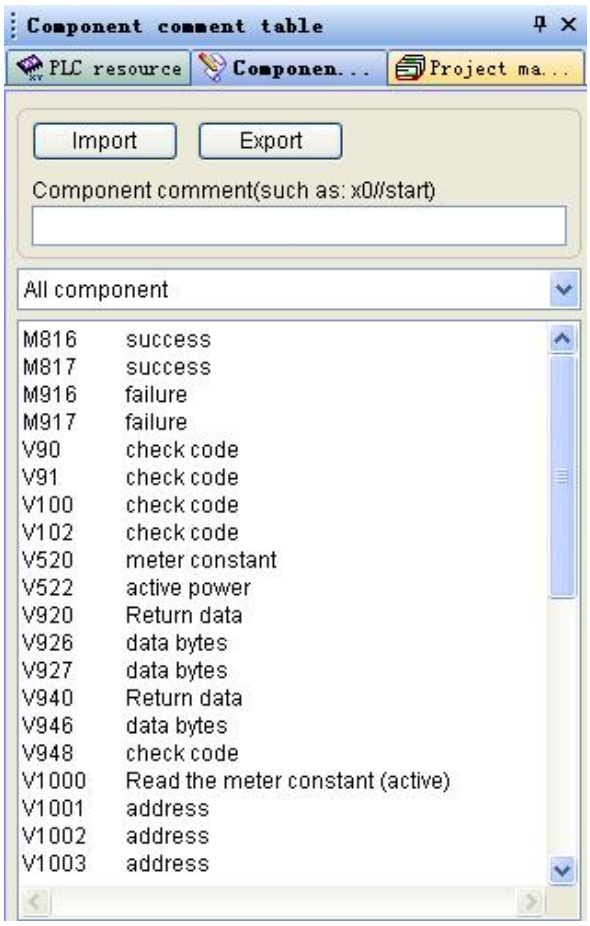
PLC resource

Via menu [Search/PLC resource] open "PLC resource" page,"PLC resource" page total 5 sub pages, list current program project CPU all resources of MPU, and system status bit. system register. interrupt and error code.



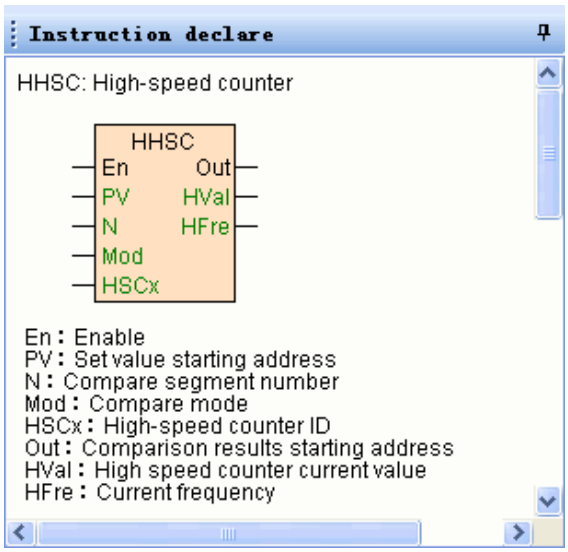
Component comment table

Via menu [Search/component comment table]open "component comment table" page, it list current program project all component comment.



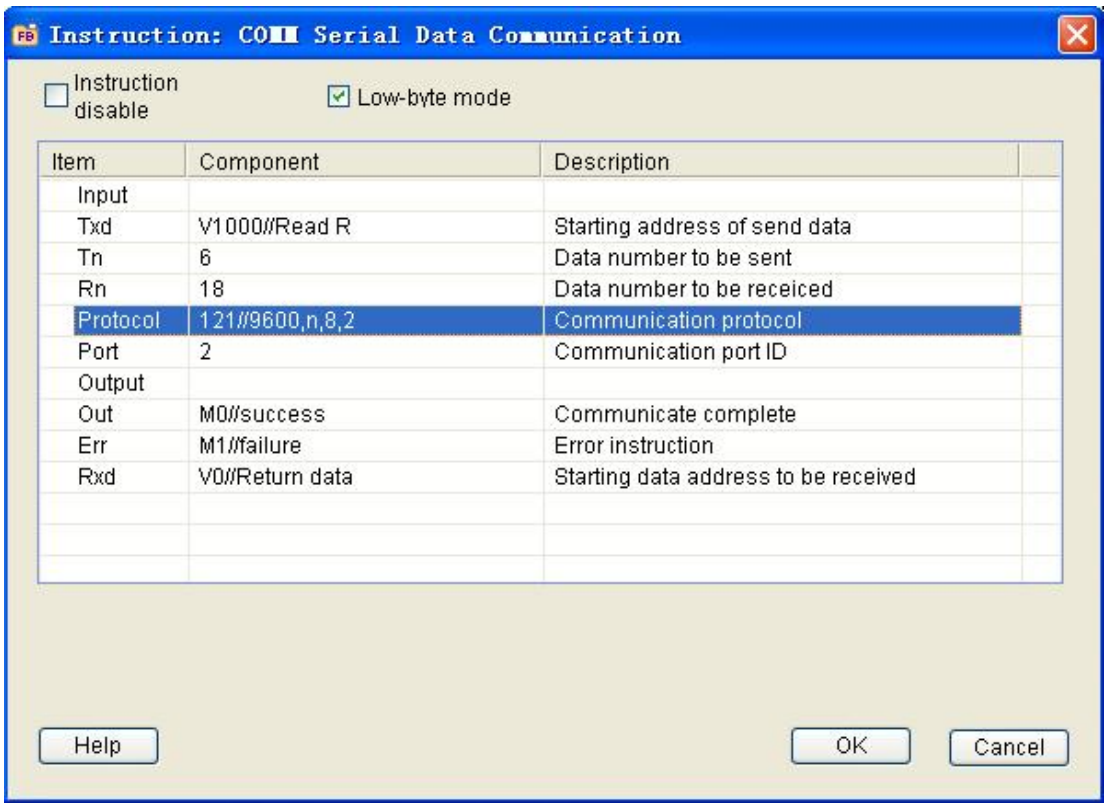
Instruction declare window

Via menu [Search/instruction declare window] open "instruction declare window" page, it list current pitched up instruction brief declare, convenience user understand the instruction.



Instruction attribute window

Double click the application instruction in the program, can open the instruction attribute window, different content will be display according to different instruction.



When select "Instruction forbidden" , the instruction can not be executed and display turn to grey .

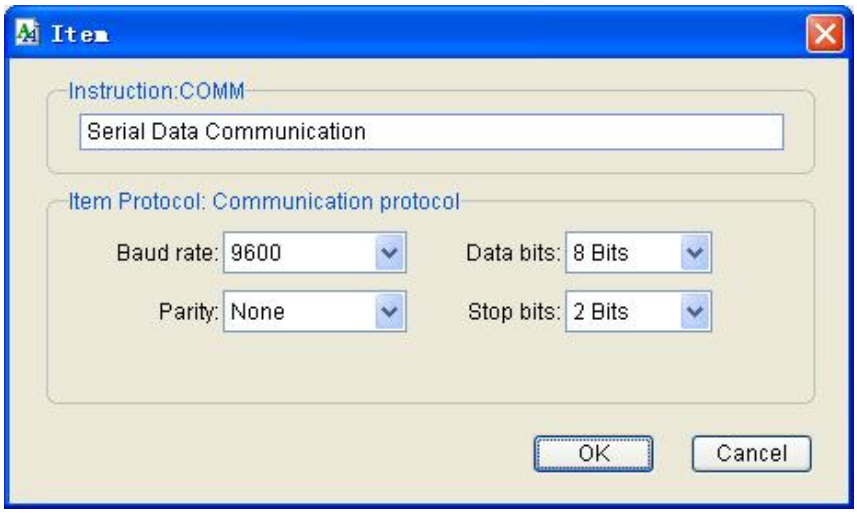
When select "Low byte mode" ,the instruction is 8 bit mode; when select "32 bits" ,the instruction is 32 bit mode.

At the window also my be modify the timer time base. counter number. input and output parameter of the instruction.

Item define window

At " instruction attribute window" double click input or output item can open "Item" window,different content will be display according to the input

or output item of different instruction, at "Item" window can define the parameter via configuration mode, need not consult material can convenience input complicated combination parameter, as: communication instruction COMM Protocol item.



Program structure

User program have main program . sub program . interrupt program three type program block constitute.

Main program block. sub program . interrupt program three sum total can not exceed 31 blocks.

All program block all can alone setting up the password, realize local encryption function.

Main program

Main program is the main part and frame of user program ,PLC permit many main program block ,between blocks of the main program category vis-a-vis, system is running, all main program block will be executed cycling, the executed sequence is at "project manager" sequence of the catalogue tree , executed from up to below.

User program at least include a main program block.

May via menu [Debug/ program executed block] regulate the executed sequence .

Sub program

sub program is function independent . program block which may be called by others program block,sub program may be with parameter or without parameter, a sub program maximum define 8 input (IN) or input output (IN_OUT) parameter and 3 output (OUT)parameter.

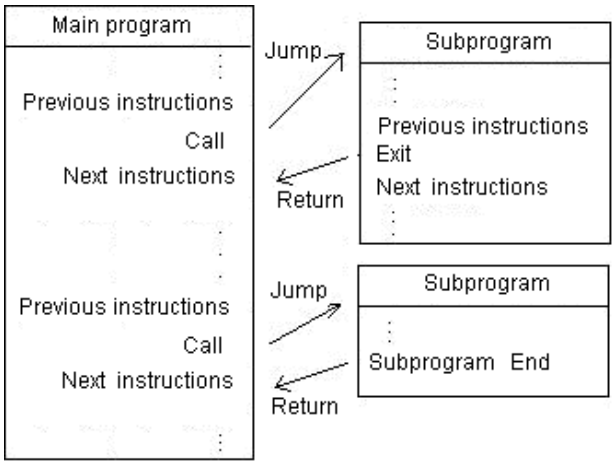
User program may be without sub program ,also include one or many sub programs .

Sub program will be executed after called by CALL instruction.

Sub program may call other sub program ,but can not call itself, also can not circulate called.

Sub program enable nest called, include program block call the sub program, the nest called depth can not exceed 8 stages.

Sub program called schematic diagram as follows:



Interrupt program

Interrupt program is a program which deal with system interrupt event . System assign a interrupt number for each interrupt event, refer to "System interrupt table".

Interrupt program executed must meet 2 conditions : one is use ATCH instruction binding the interrupt program and the interrupt event (interrupt number), two is system happen the interrupt event. Among, catch X0~X7 rising edge or falling edge interrupt must at "PLC hardware configuration " window "X digital input parameter" open.

When system happen the interrupt event, program executed broke, program automatic jump to the interrupt event binded interrupt program executing, until interrupt program executed complete, system return to normal program executing.

User program may be without interrupt program , also may be include one or many interrupt program .

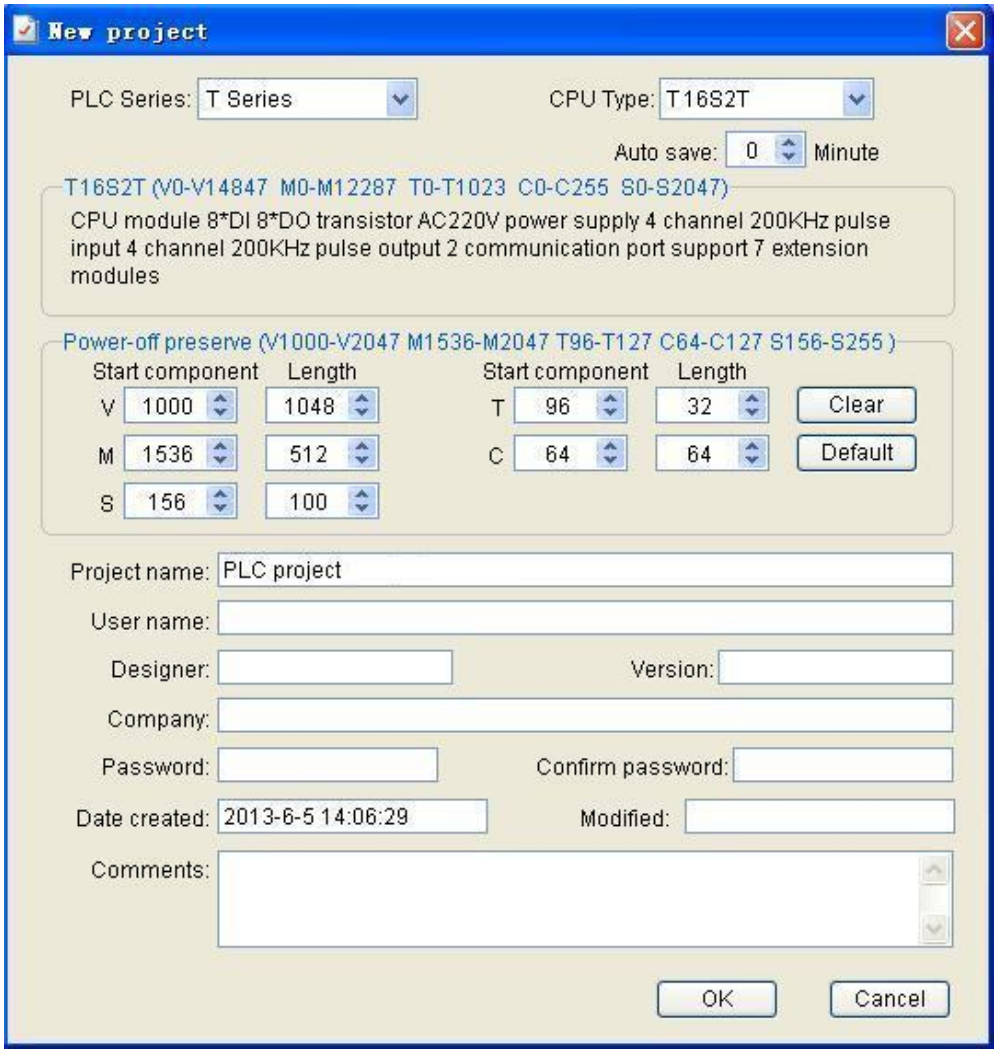
System interrupt controlled by ENI (interrupt enable)and DISI (interrupt prohibit) instruction ,at interrupt prohibit, system can not happen interrupt event. Default is enable interrupt.

Note:interrupt program only executed once at corresponding interrupt event happen.

Program project build

New program project

Via click menu [File/new program project]. click tools bar  button or use shortcut key Ctrl+N,open "new program project" window.



At "PLC series " select among pull listing "PLC series ".

At "CPU type " select among pull listing "CPU also MPU type ".

Latched area after power off may be defined just as user wishes, may set V. M. S. T. C five type component latched area after power off start component and the length. System default latched area after power off listing as follow:

Component type	Latched area after power off	Component number
V	V1000~V2047	1048
M	M1536~M2047	512
S	S156~S255	100
T	T96~T127	32
C	C64~C127	64

At "Project name" input new project name, The project name will be display in main frames of the project manager.

At " User name". "Designer". "Versions". "Company" etc. column input the project relate to information.

If program project protected by password, then at "password" and "affirm password " input protect password.

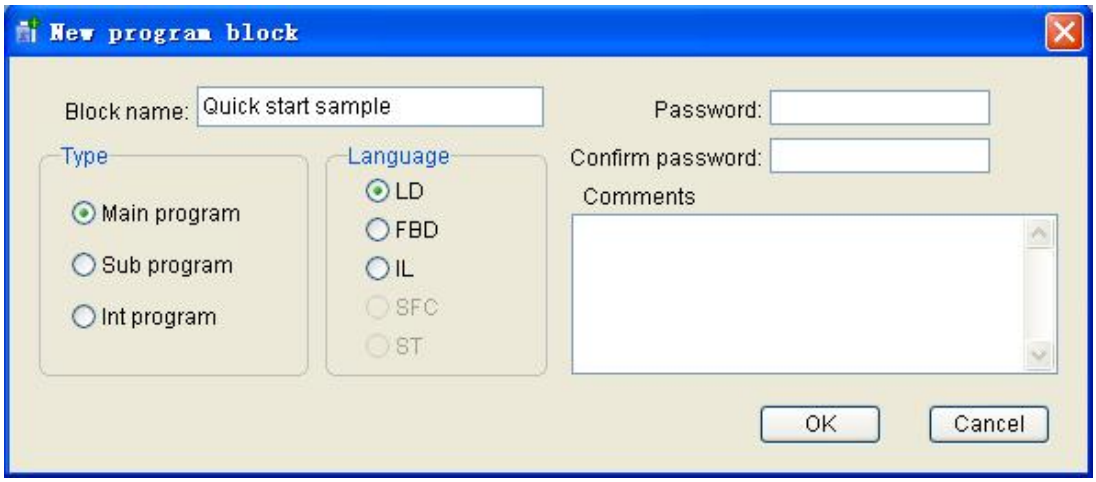
At "Note" may be input the project relate to note information.

Press "Confirm" button build a blank program project.

New program block

User program have main program . sub program . interrupt program three type program block constitute. User build many program block according to control request.

Click menu [File/New...../Main program block] or click tools bar button "main program block", open "new program block" window.



At "block name" input the name of new program block(the name will be displayed in the project manager), the select program block type and programming language, may be write some comment declare at note .

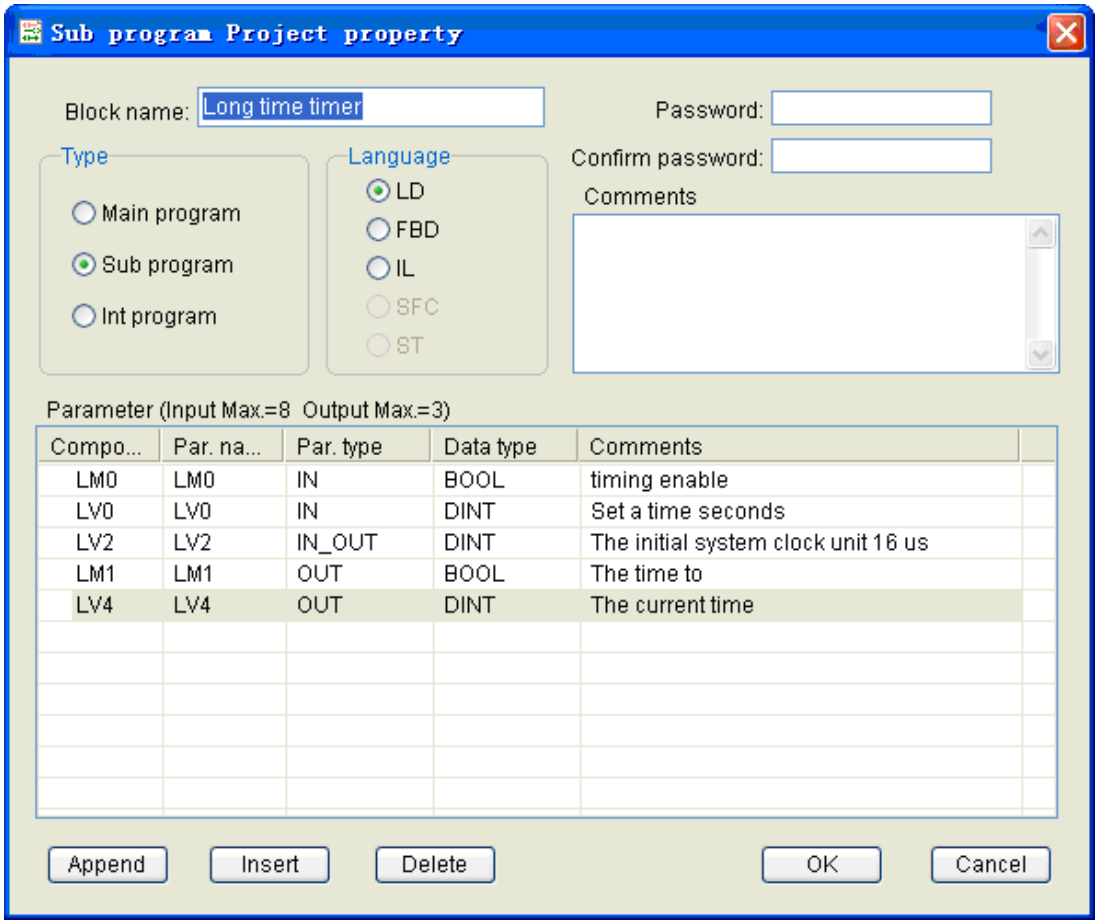
Password" and" confirm password" two text is use for set the program block (program block. sub program . interrupt program) protect password. If set password for the program block, then request user input correct

password and then search or edit it, each program block may be set independent password, realize local cypher function.

User may select oneself favorite programming language to write program , press "confirm" button to build a blank program block.

New sub program

When build a new sub program or open sub program attribute window, bottom of the window will listed the sub program parameter table, as follows.



One sub program maximum define 8 input (IN) or input output (IN_OUT) parameters and 3 output (OUT) parameters. Each parameter has it parameter type (IN. OUT. IN_OUT) and data type (BOOL. WORD. DWORD. INT. DINT. REAL).

When use CALL instruction call sub program ,CALL instruction input output parameter data type must fit the parameter data type of the sub program defined.

System automatic distributive the parameter address, when select BOOL type of the data, address from LM0 distributive, address of others parameters type from LV0 distributive.

Press "Add" button add a new parameter line, new parameter type default to IN, data type default to BOOL.

Click parameter name field of the parameter ,at textbox input the parameter name, parameter name maximum 5 characters, must defined different name of each parameter , the name will display at CALL instruction.

Click the parameter type field ,at select among pull listing the parameter type of the parameter (IN. IN_OUT. OUT), system will automatic sort the parameters according to the parameter type .

Click the parameter data type field, at select among pull listing data type of the parameter (BOOL. WORD. DWORD. INT. DINT. REAL).

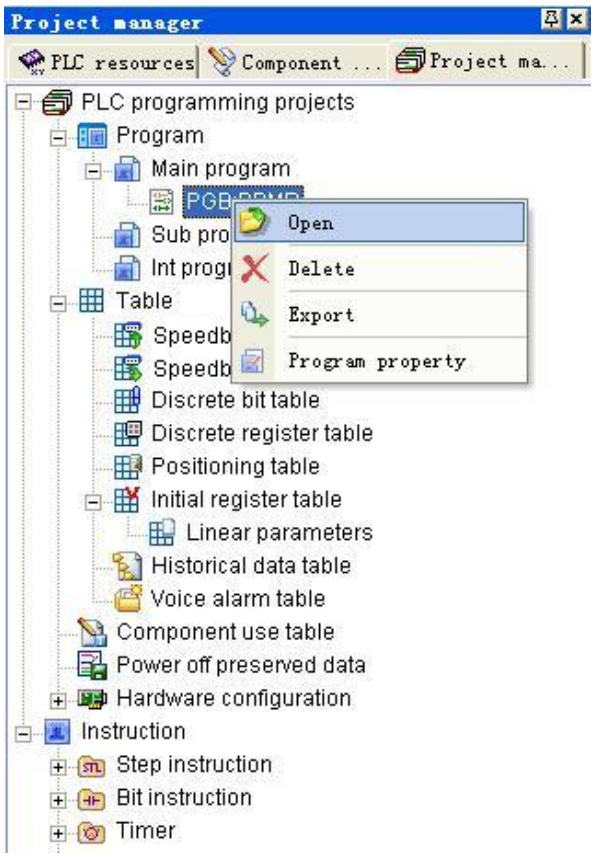
Press "Insert" button ,insert a new parameter before current line.

Click parameter line will be deleted , press "Delete" button , the parameter will be deleted.

). May input the comment of the parameter at the comment field .

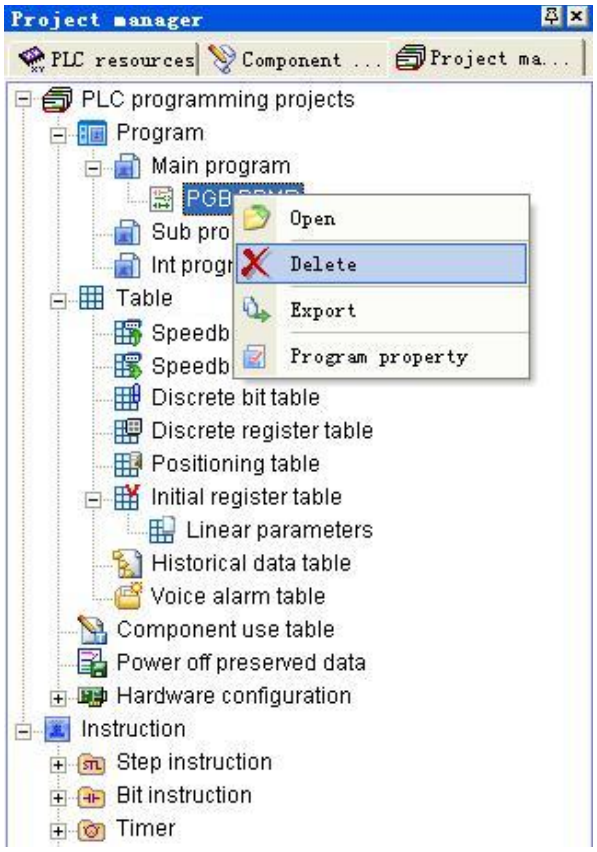
Open program block

Double click "Project manager" directory tree program block name or via right button menu "Open " open the program block .



Delete program block

Right button click "Project manager" window will be deleted program block name, at popup right click menu click "Delete" .



Program block execution sequence adjust

Click menu [Debug/ program block execution sequence], open " program block execution sequence" window.

Listbox listed current program project all main program block , number is the current execution sequence, pitch up will be adjusted execution sequence program block ,click "Move up" or "move down" button, may adjust the program block execution sequence.

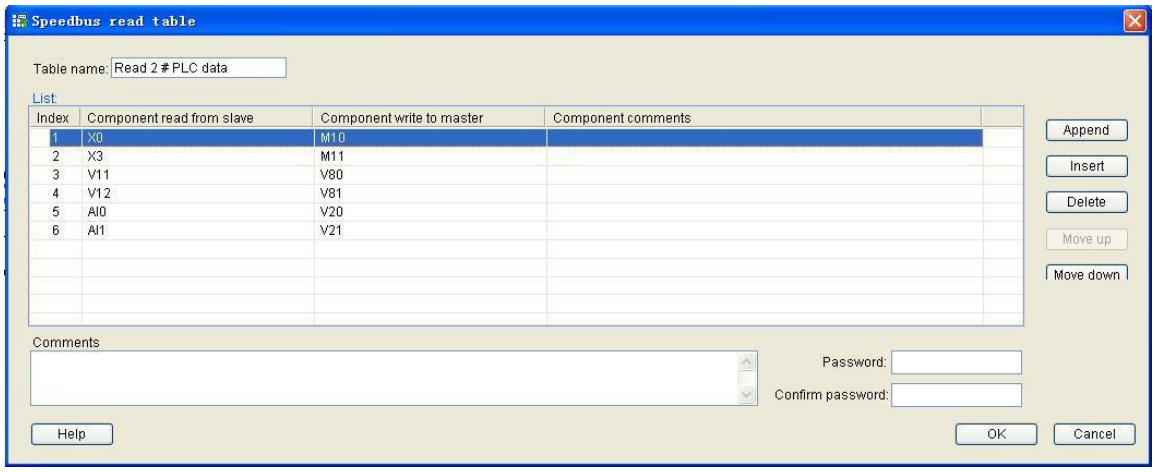
Sub program called by CALL instruction,interrupt program executed trigger via system interrupt , can not adjust the execution sequence.

Instruction use table

PLC programming software supply many instruction use table, use these tables may reduce many program workload, save program space, realize as initialized the data etc. function. Each table can be set password independent, also may be import or export .

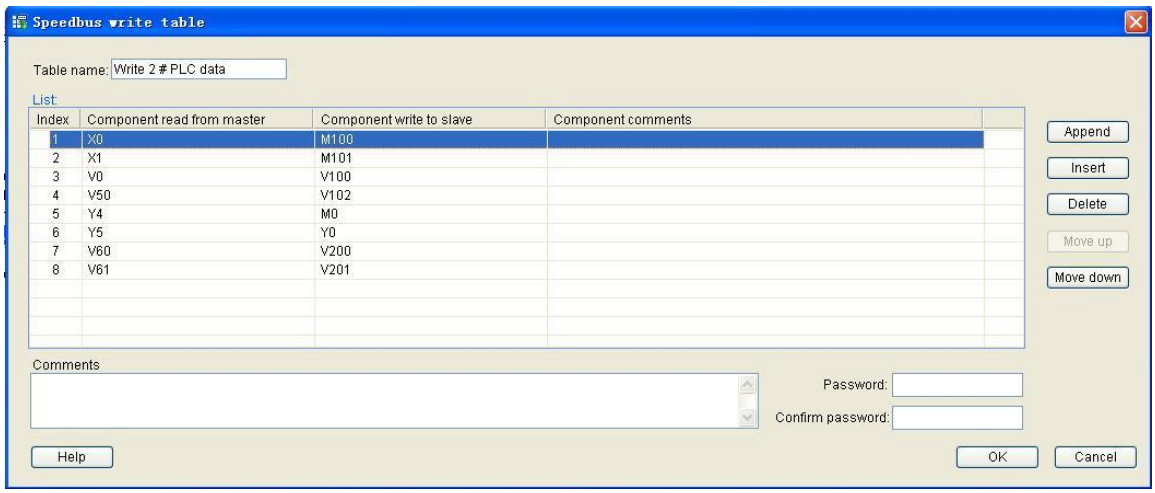
Speedbus read read communication table

"Speedbus read communication table " as the HWRD instruction Table item parameter ,when communicate between PLC and PLC via Speedbus protocol ,HWRD instruction according to "Speedbus read communication table " automatic read data from slave and then write to master PLC. As follows:



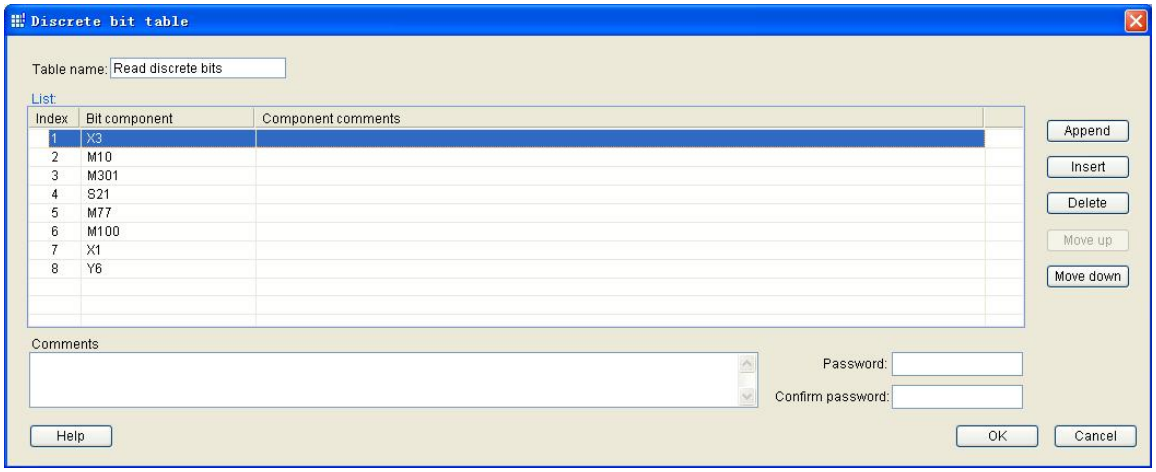
Speedbus write communication table

"Speedbus write communication table " as HWWR instruction Table item parameter ,when communicate between PLC and PLC via Speedbus protocol ,HWWR instruction according to "Speedbus write communication table " automatic read data from master PLC and then write to slave PLC. As follows:



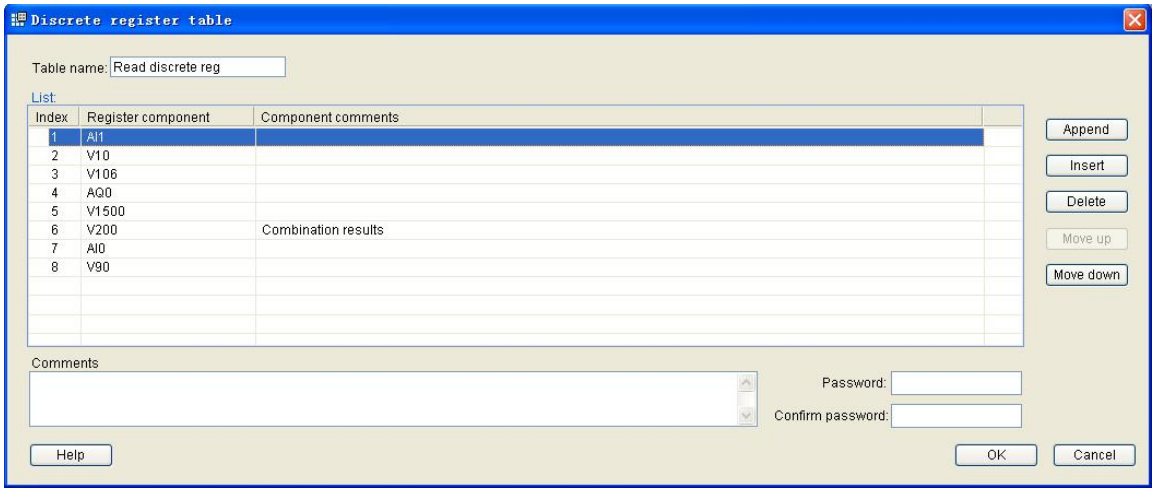
Disperse bit component table

"Disperse bit component table" define a group disperse bit component, use for BUNB. BUNW. BDIB. WDIB instruction Table item parameter , quick realize bit component combination or disperse, convenience upper computer access. As follows :



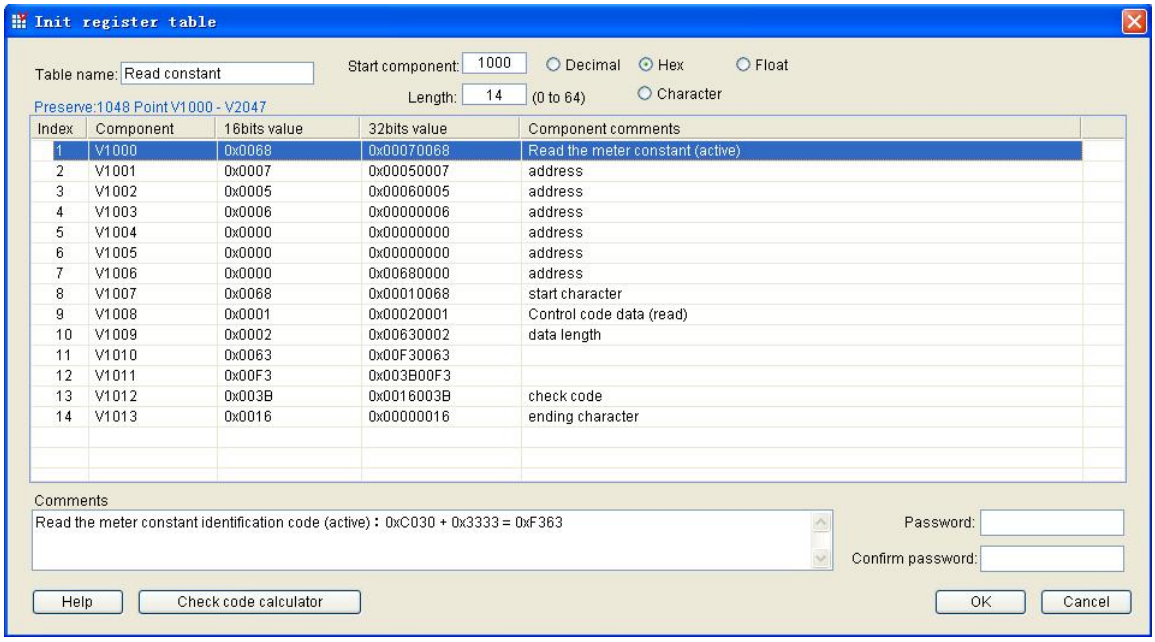
Disperse register component table

"Disperse register component table " define a group disperse register component, use for WUNW. WDIW instruction Table item parameter , quick realize register component combination or disperse, convenience upper computer access. As follows :



Initial register value table

Use for defined the initial value of register component ,when program project downloaded to PLC , set initial value in the table will be downloaded follow the program ,may define integer. long integer. floating point number. character type data. as follows:



"Start component" assign the V component start address, "Length" assign the number of components.

"Decimal". "Hexadecimal". "Floating point number". "Character" select the input mode of the data.

At "Decima". "Hexadecimal" mode ,at "16 bit register value" field input 16 bit integer, at "32 bit register value" field input 32 bit integer. Note:"32 bit register value" field display the 2 continuation register value of border

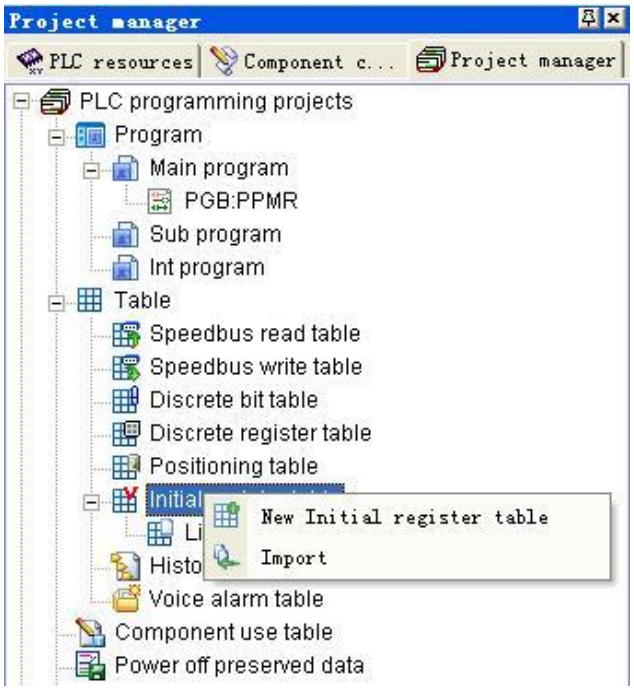
upon, as upper drawing the first line V1000 register "32 bit register value" are V1000V1001 value.

At "Floating point number" mode direct input the floating point number. Note :floating point number is 32 bit single precision value, occupy 2 continuation registers.

At "Character" mode can direct input character .

New table

Right button click "Project manager" window will be new build table name, at popup right click menu click "New";click menu[File/New.....]or click Tools bar  button select the table type which will be new build , thus will be build many kinds of tables. Different interface and content according to different table .



At "Table name" input the name of new table (the name will be displayed in project manager), may write some note declare in the note text field .

"Password " and "Confirm password" two text field are use for set the protect password of the table. If the table password be set , then require after user input the correct password can be search or edit, each table can be set password independent, realize locality encryption function.

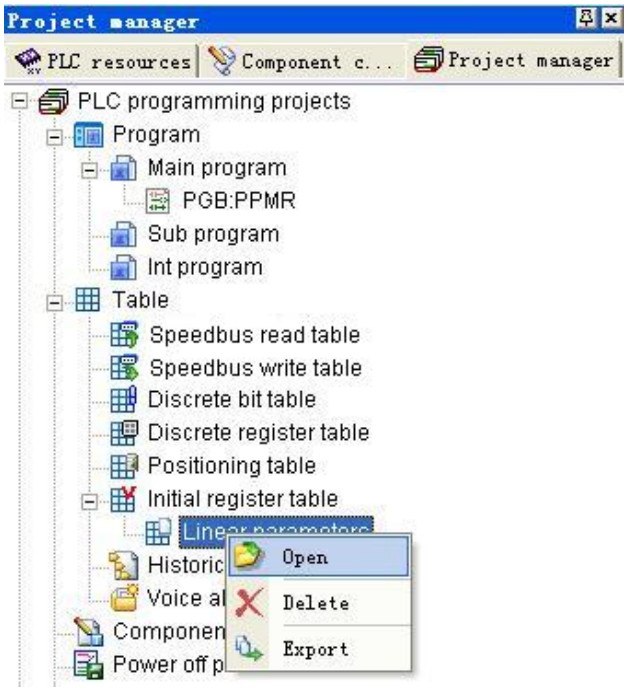
The "initial register value table " defined the "Start component" and "Length ", then input value of the component. Others kind of tables via "Add". "Insert". "Delete". "Move up" and "Move down " button to edit the content of the table.

At "Note" input the note of the table, at " component note" bar Edit component note.

Press "Confirm" complete build the new table.

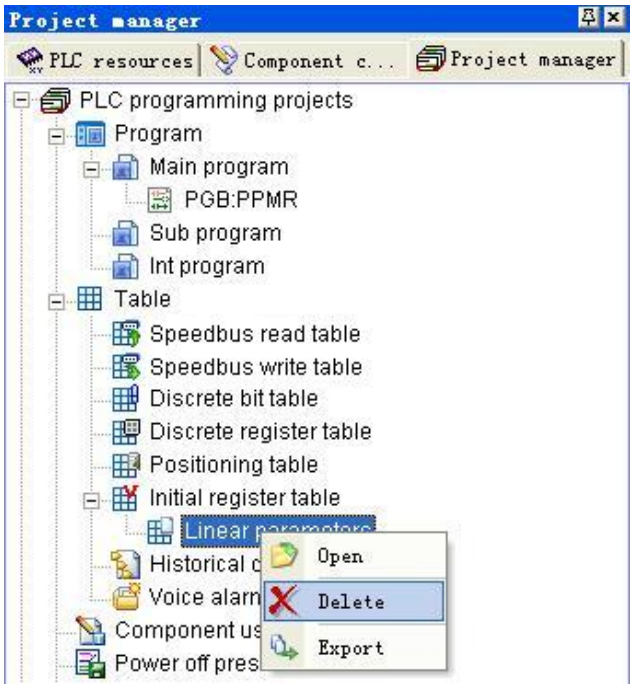
Open table

Double click "Project manager" catalog of tree or via right button menu "Open " may open the table.



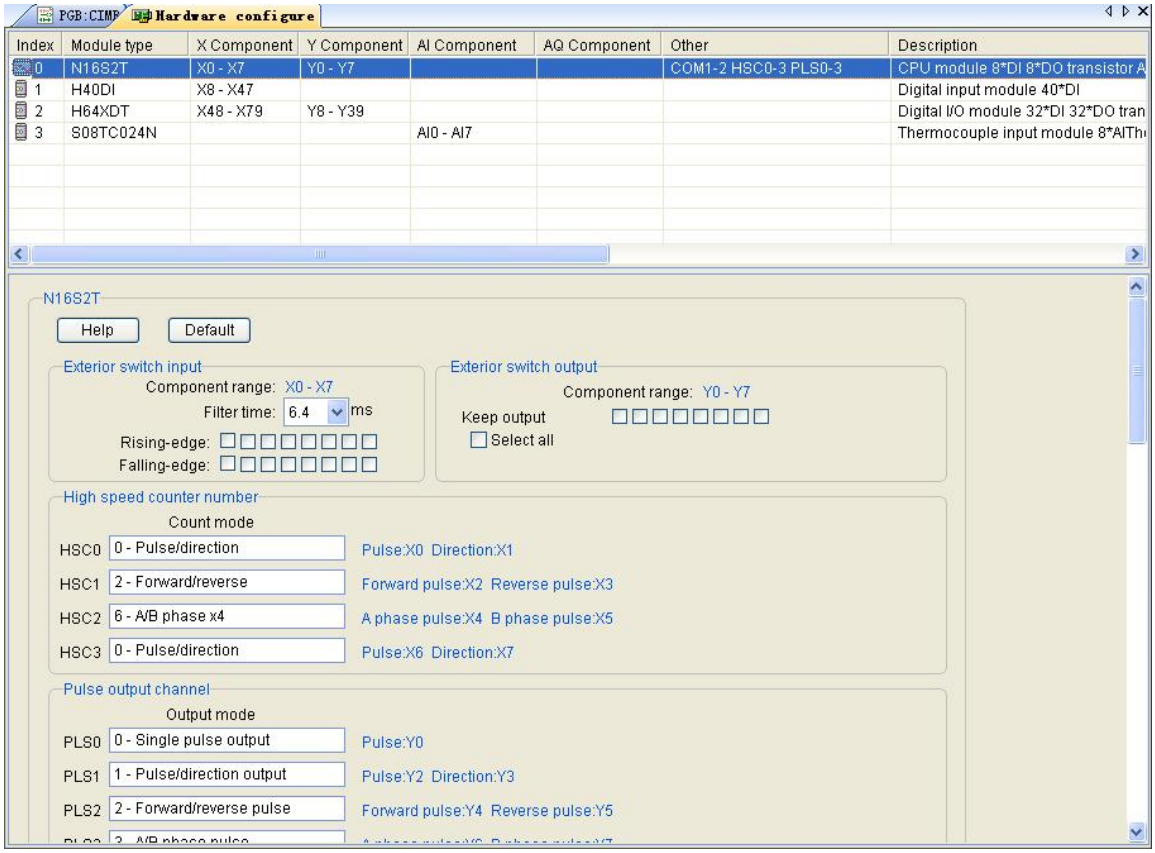
Delete table

Right button click "Project manager" window the table name which will be Delete, at popup right click menu just click "Delete".



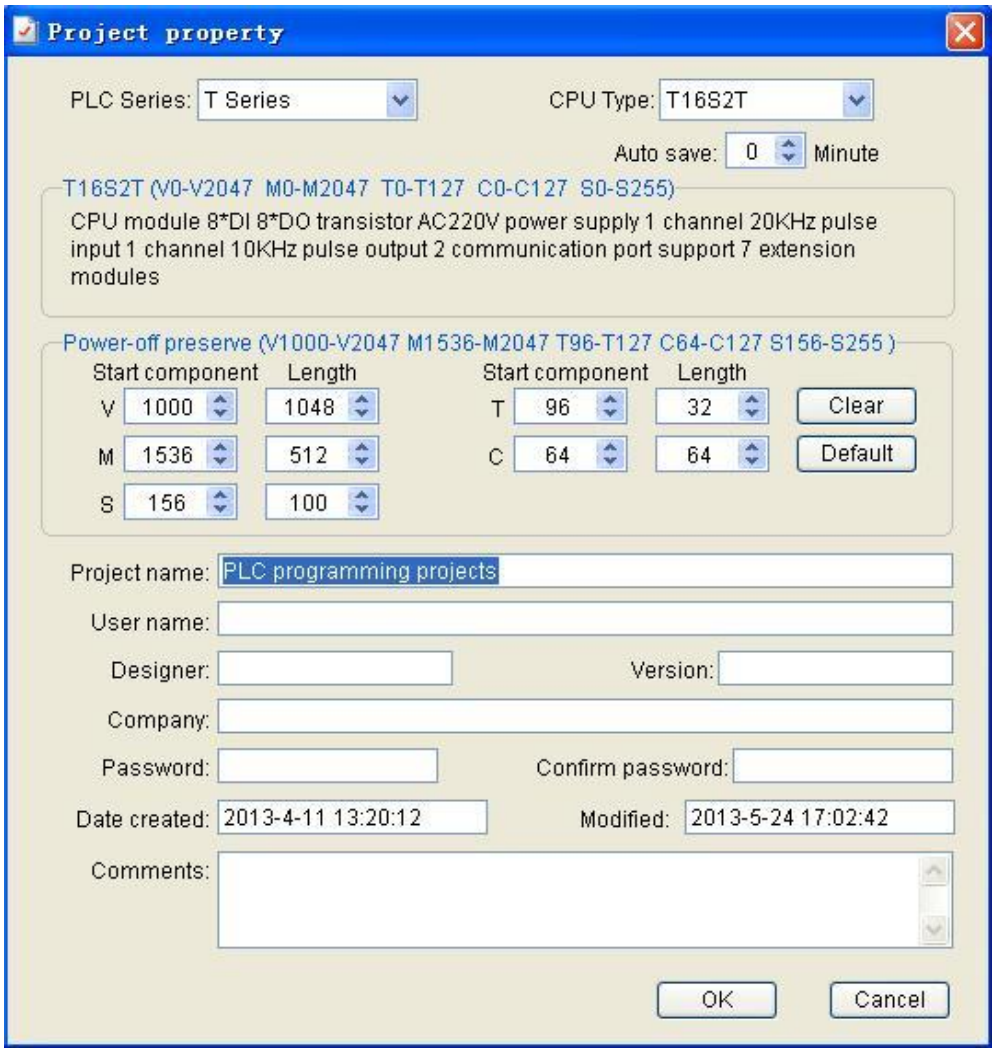
PLC hardware configuare

Double click "Project manager" catalog tree "PLC hardware configuration " hybrid or menu[Search/PLC hardware configuration], open "PLC hardware configuration " window. user in rise reality hardware model add extend modules ,may be add. delete or regulate the position of the extend modules, may be configurate the parameter of configurated MPU and extend modules (No more the default value is OK).



PLC MPU model

PLC MPU type at new program project may be selected and confirm, if want to modify the MPU type menu [File/ program project attribute]Open " program project" window, select correct "PLC series " and "CPU type ",press "Confirm" button .

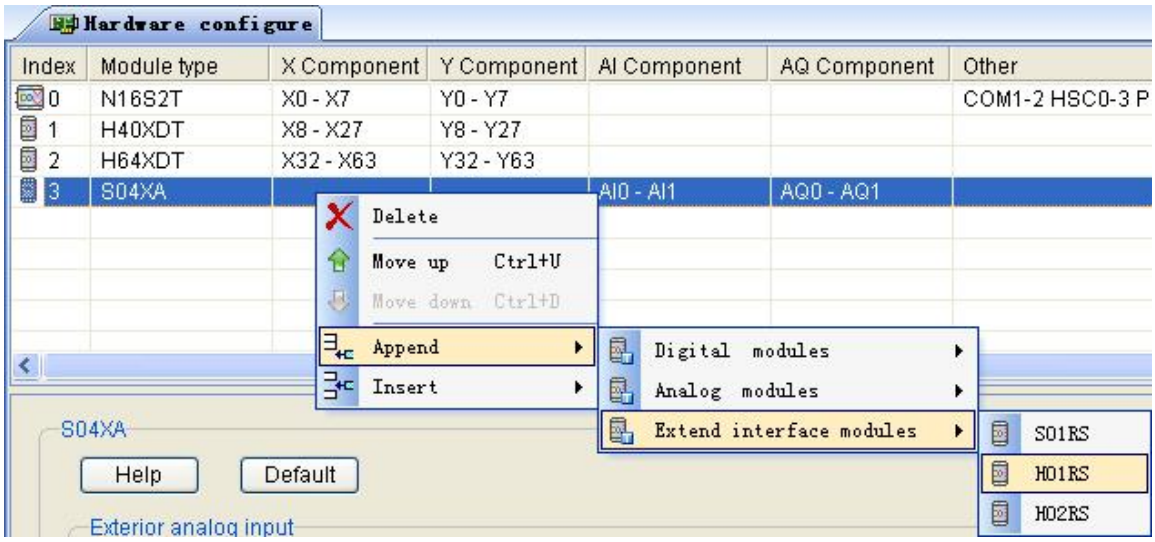


Extend block edit

1. Open up project manager "PLC module" catalogue tree, find the module model will be added, double click the module or drag the module to the hardware configuration table, the module added to the current position in the PLC hardware configuration table.

2. Callout right button menu, may be add. insert or delete the extend module ,also may be via move up(Ctrl + U) or move down (Ctrl + D) regulation the module position.

3. Click the module will be deleted , press "Del" key, the module will be deleted .



Assign external I/O component

PLC external I/O component (X. Y. AI. AQ) will be automatic assigned by system , component number is decimal system, when add. delete or regulation the module position system will afresh regulation .

Note

:AI and AQ component number is continuation number, but X and Y component number is not continuouation. Because

X and Y component group in 8 individual

, when module I/O point number is not 8 times will be generated the not

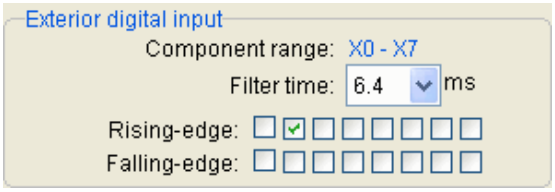
continuation number

.

X digital input parameter

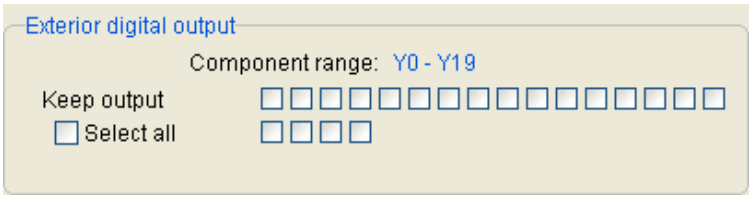
Can set external digital input "Filter time", filter time get over long then response get over slow. filter efficiency get over well, filter time get over short then response get over fast. filter efficiency get over poor.

CPU built-in X0~X7 digital input support edge catch function, if ant to use edge catch, must tick relevant X channel "Rising edge catch" or "Falling edge catch" option.



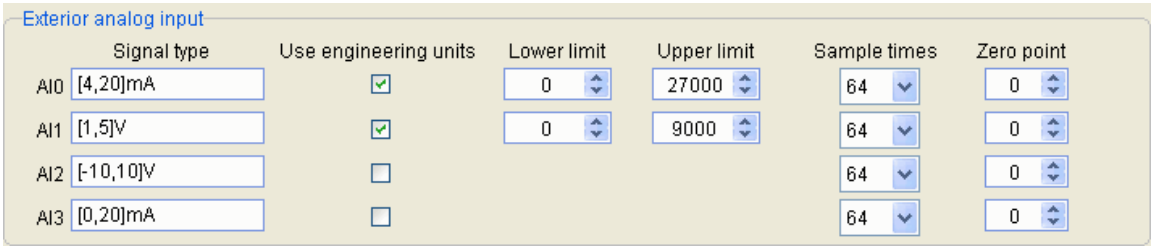
Y digital output parameter

'hen configurate the Y channel to "Stop output" ,when PLC stop the Y channel will be keep ON output.

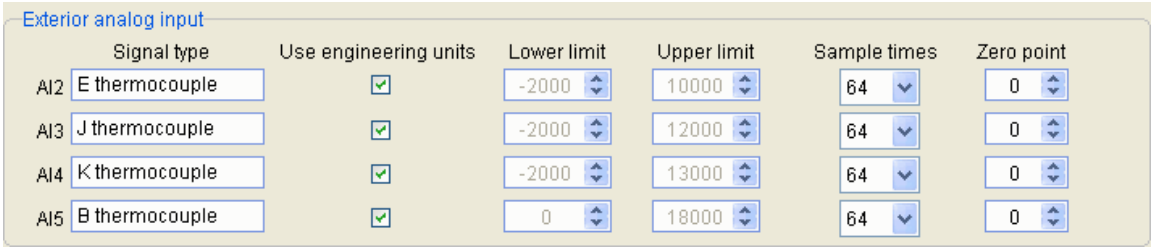


AI analog input parameter

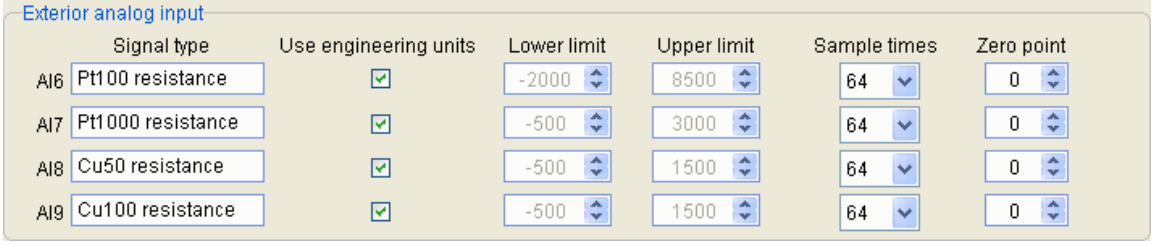
Linear A input support 6 type signal :[4,20]mA. [0,20]mA. [1,5]V. [0,5]V. [0,10]V. [-10,10]V.



Thermocouple module support:S. K. T. E. J. B. N. R. Wre3/25. Wre5/26. [0,20]mV. [0,50]mV. [0,100]mV.



Thermal resistance module support:Pt100. Pt1000. Cu50. Cu100.



May configurate AI channel use quantities, if use quantities then the channel measuring range resolve via "Upper limit" and "Lower limit" , if don't use quantities then use 0~32000 code value.

"Sample times" influence the AI channel response time and filter efficiency ,sample times get over big then response time get over long. filter efficiency get over well ;sample times get over small then response time get over short. filter efficiency get over poor .

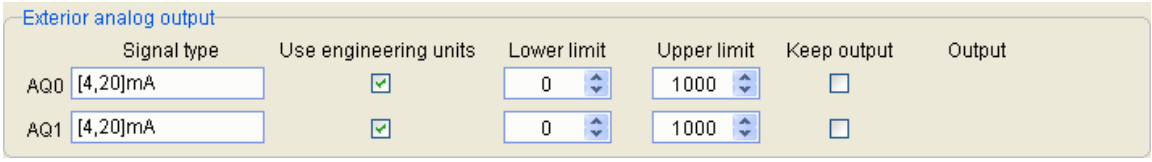
If external sensor generate zero deviation, may via set "zero revise" to regulation moreover don't replace the sensor. when sensor minus deviation the zero deviation set to positive value, when sensor positive deviation the zero deviation set to minus value .

AQ analog output parameter

AQ output support signal type :[4,20]mA. [0,20]mA. [1,5]V. [0,5]V. [0,10]V. [-10,10]V, each channel may be set independent.

Configurate AQ channel may use quantities, if use quantities then the channel measuring range resolve via "Upper limit" and "Lower limit" , if don't use quantities then use 0~32000 code value.

When configurate AQ channel "Stop output", may set a output value, thus when PLC stop the AQ channel will keep output the set value.

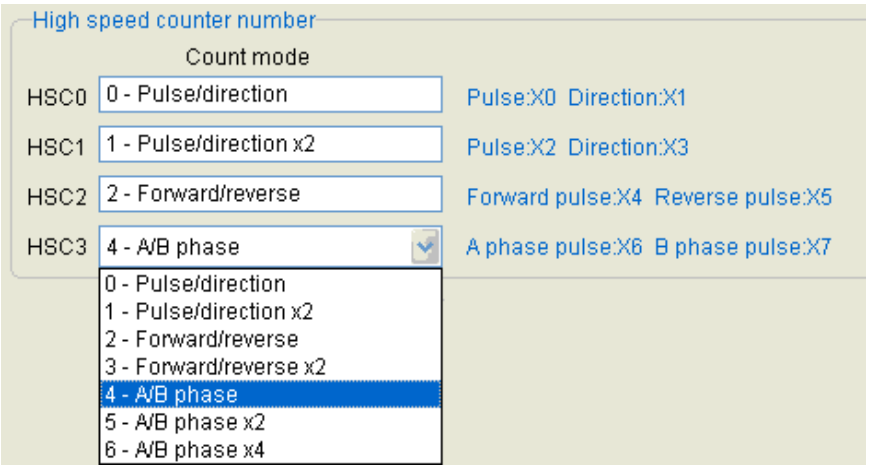


HSC high speed counter parameter

High speed counter support :pulse/direction . positive/negative pulse . A/B phase pulse input model, support 1. 2. 4 frequency multiplication counting model

High speed counter channel signify by HSCx ,each channel use 2 high speed pulse input point.

If there is no high speed counter HHSC in program, don't need configurate parameter, relevant high speed input point may be use in common X point.

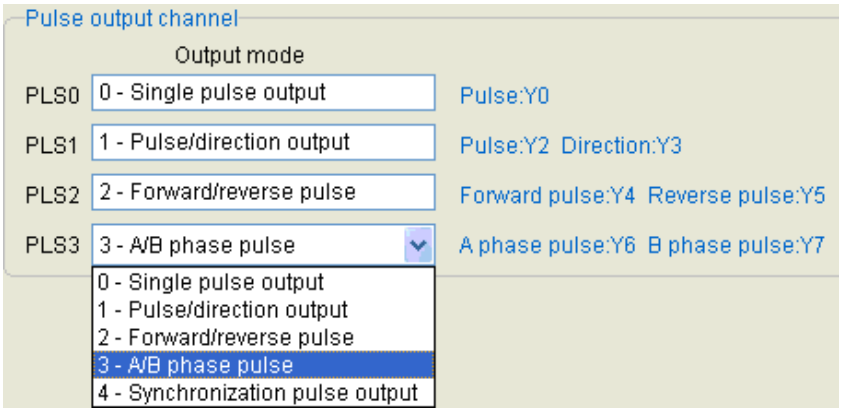


PLS high speed output parameter

High speed pulse output support : single pulse. pulse/direction . positive/negative pulse . A/B phase pulse . synchronization pulse output, total 5 type output mode.

High speed pulse output channel signify by PLSx ,each channel use 2 high speed pulse output point.

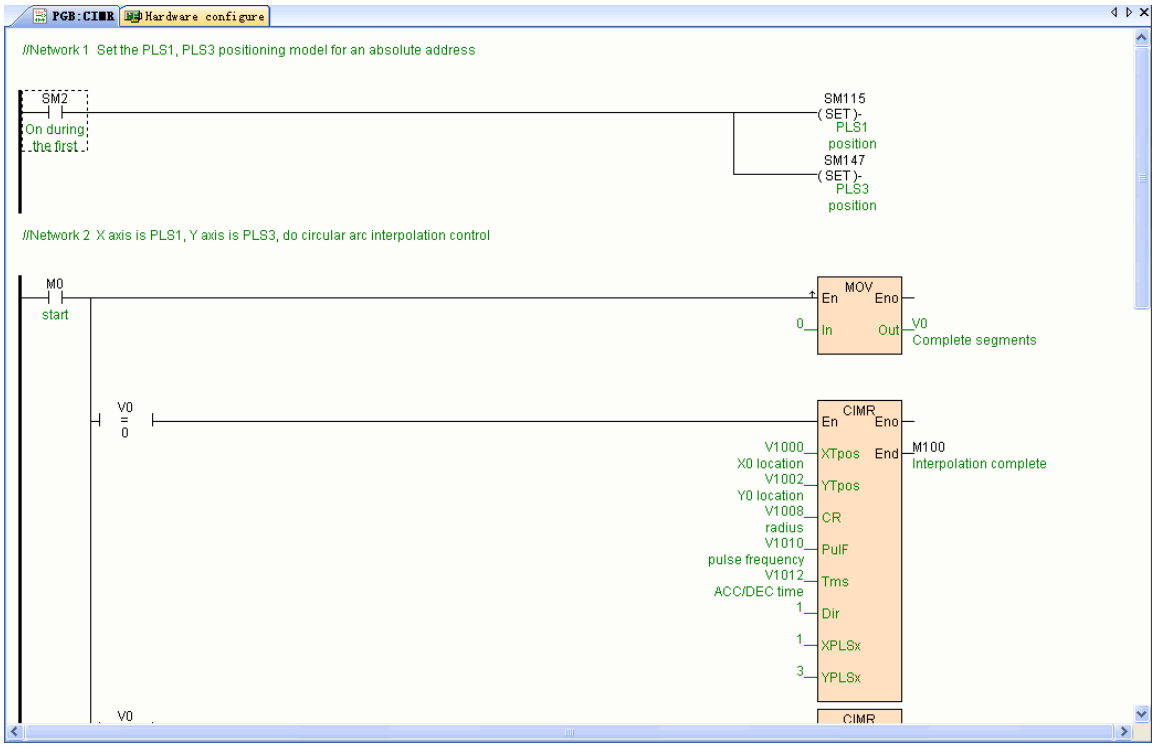
If there is no high pulse output instruction (PLSY. PLSR. PPMR. CIMR etc.) in program, don't need configurate parameter, relevant high speed output point may be use in common Y point.



LD ladder diagram programming

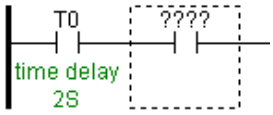
LD work area

At LD work area proceed the LD language edit, as add network. delete network. regulate network sequence, series or parallel switch, add . delete instruction etc..

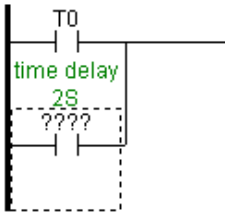


Switch edit

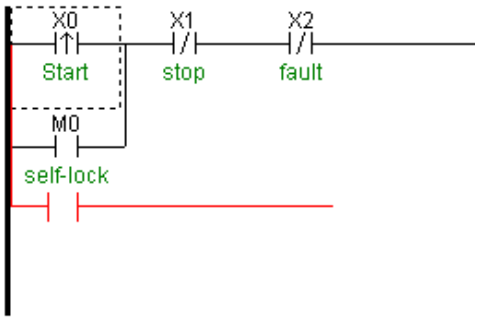
Series a switch:click button or press "F9".

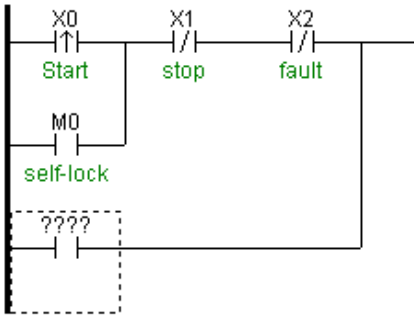


Parallel a switch:click button or press "F10".



Parallel branch: click tools bar button , mouse change to , mouse move to the start position will be parallel branch (X0 switch position) and click , then mouse move to the end position (X2 switch position) and click, thus complete a parallel branch.

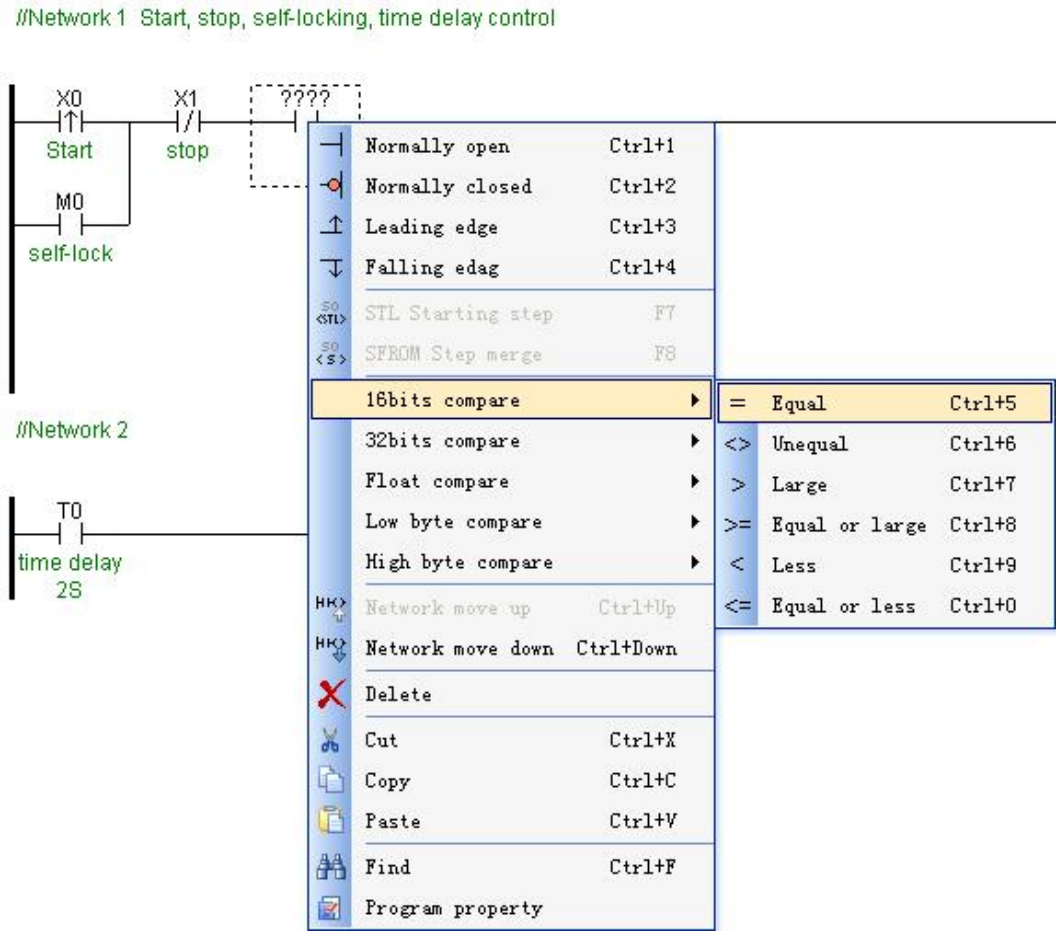




Mouse click or use keyboard move the gridlines to switch position, then may select the switch. Can input the switch component . change the switch status, can copy . cut. delete etc. operation .

Switch status change

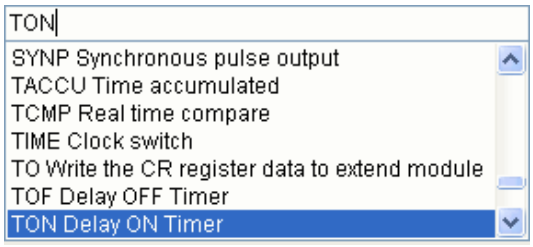
Mouse click or use keyboard move the gridlines to switch position which needed change the status, use shortcut key or mouse right button callout right button menu then may change switch status.



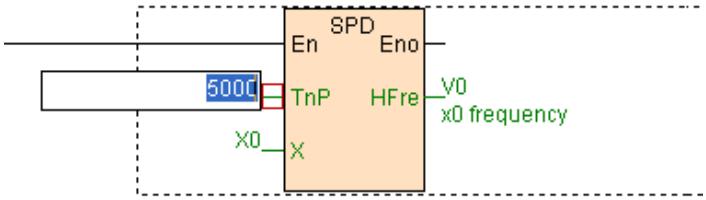
LD instruction edit

Double click project manager instruction name in the instruction tree then may add the instruction to the network, also may use drag mode pull the instruction to the relevant position .

Support keyboard input instruction , input instruction name first character will automatic popup instruction input window, input instruction name then press enter may add instruction, may press "ESC" close the window at any time.



Mouse click or use keyboard move the gridlines to instruction position, press enter may input or modify the instruction parameter.

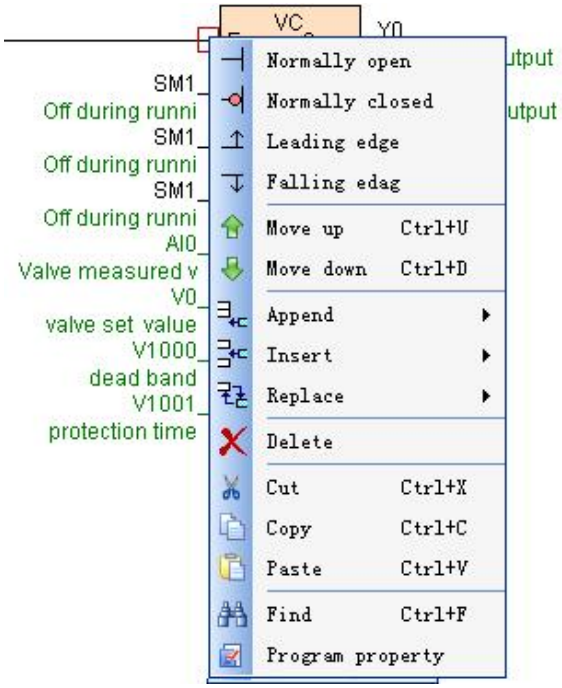


After select the instruction, may copy. cut. delete the instruction, also may move up. move down the instruction sequence.

Double click instruction may open "instruction attribute" window, complete change instruction mode. set instruction forbidden. set timer time base etc. operation .Double click input. output item may open " Item" window, via configuration mode input the item parameter.

Change status of the instruction input item

When mouse move to the instruction input item will turn up a red box, use mouse right button callout right button menu then may change instruction input item status (Normal open. Normal close. Rising edge. Failling edge).

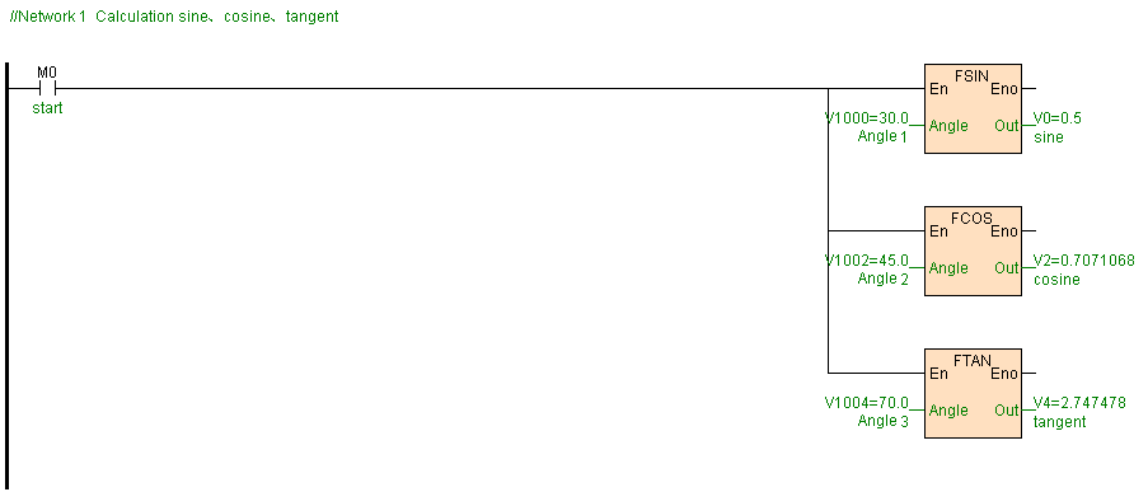


Branch edit

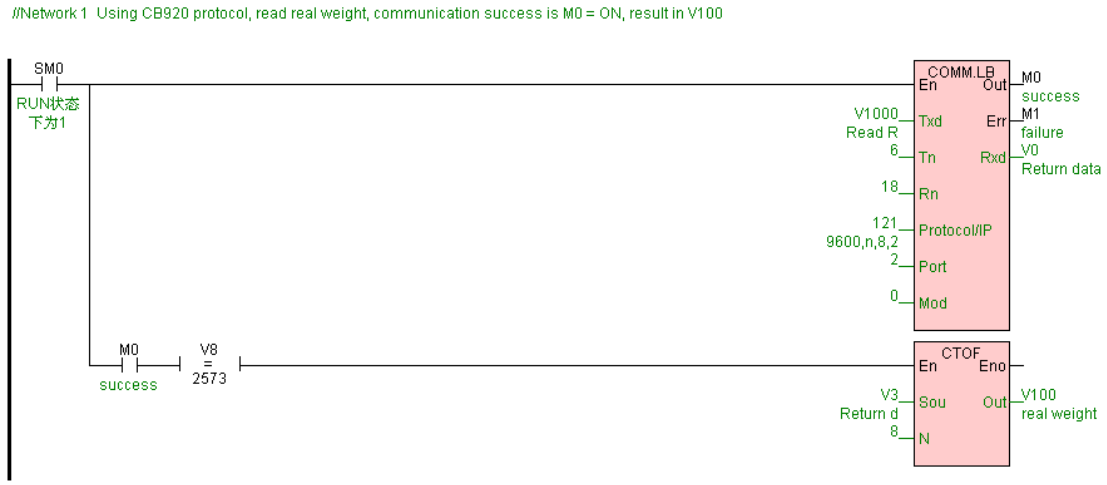
LD language support parallel output and branch output mode execute the instruction, if the same logical condition to execute the instruction then

use parallel output; if not the same logical condition to execute instruction then use branch output.

As follows, the same logical condition to execute the instructions FSIN. FCOS and FTAN then use parallel output.



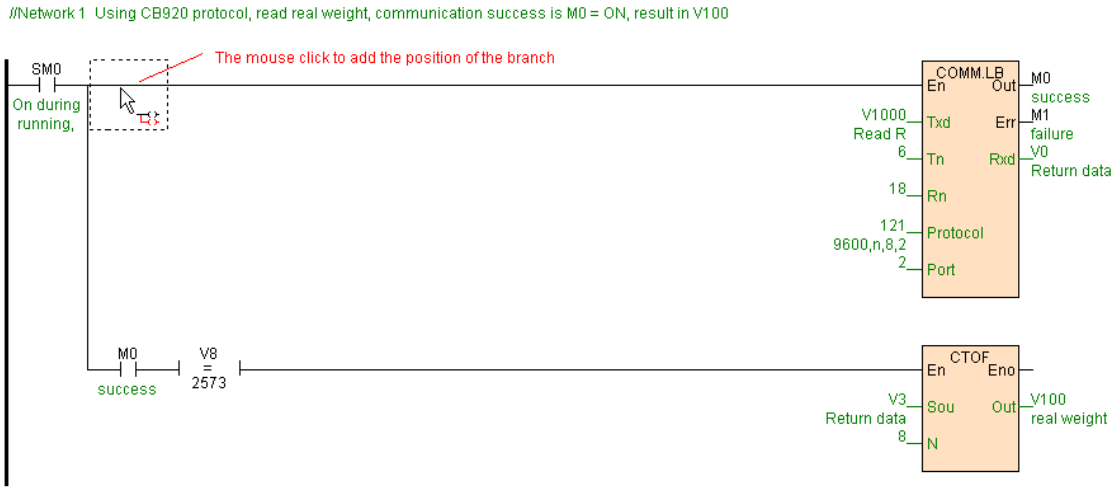
As follows, not the same logical condition to execute instruction COMM.LB and CTOF then use branch output.



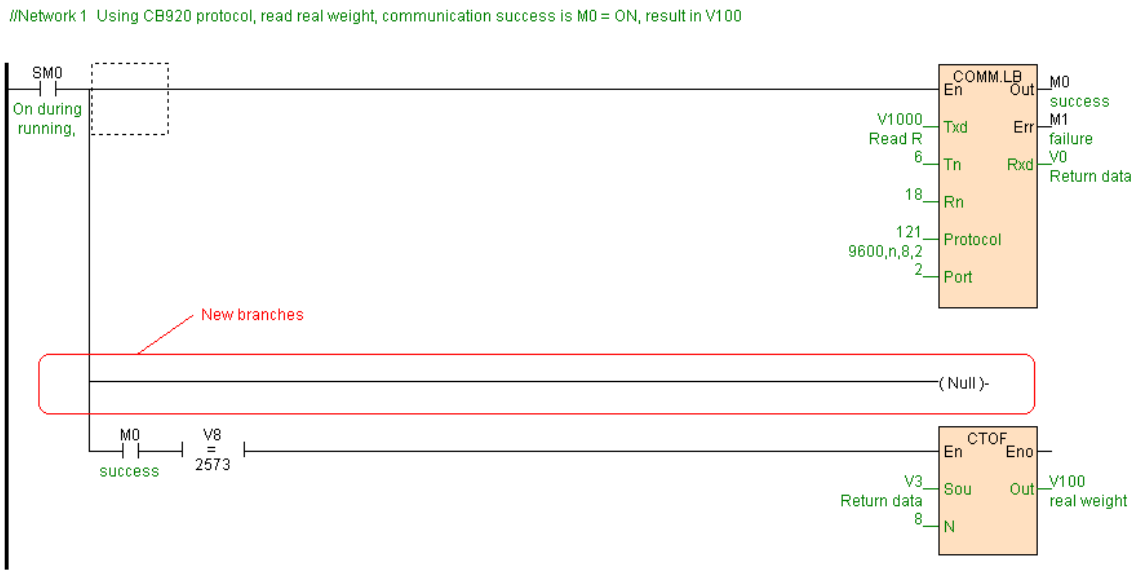
Add instruction default to parallel output, branch output need first add new branch then add instruction.

Click Tools bar button , mouse change to , mouse move to the position which will be added branch and click then may add new branch, new branch add to behind the current branch .

Before add branch as follows:

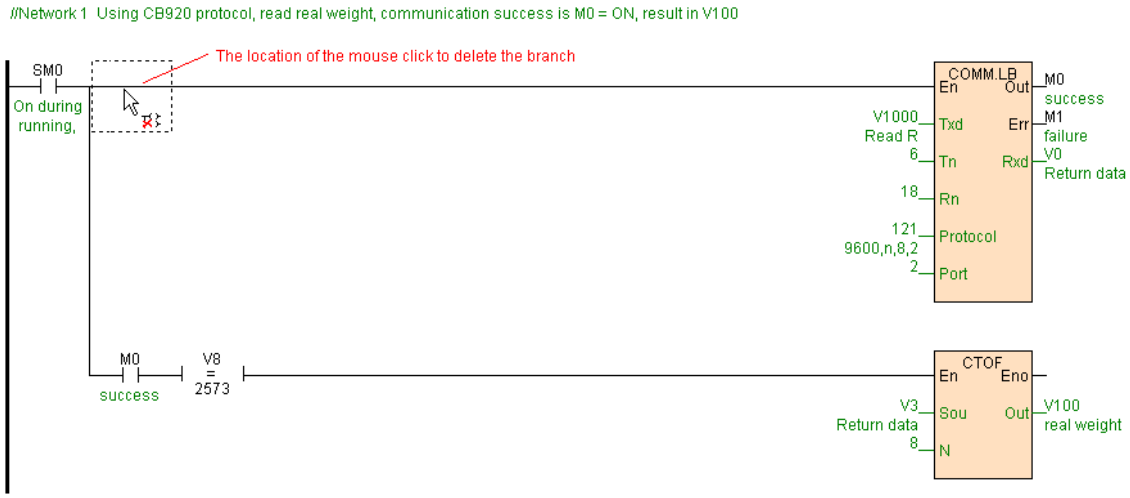


After add branch as follows:

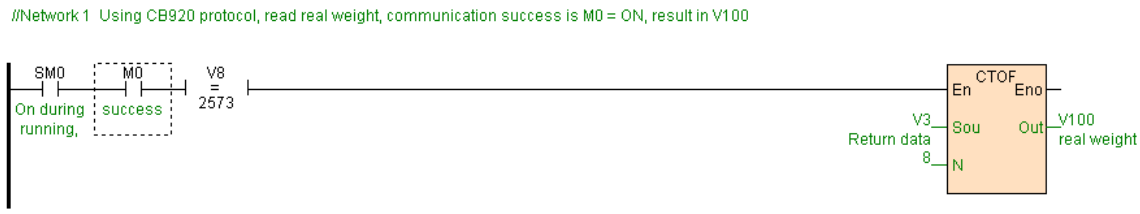


Click Tools bar button ,mouse change to ,mouse move to the position which will be deleted branch and click then may delete the branch.

Before delete the branch as follows:



After delete the branch as follows:



Network edit

Click button or use shortcut key Ctrl+L add a new network.

Click button or use shortcut key Ctrl+I insert a new network.

Mouse pull down at the network, may select current network or many networks, selected network content will display reverse, press "Del" key may delete, also may cut. copy . paste etc. operation .

Mouse double click network title may edit the network comment, when gridlines at the network title position may direct input the network comment.

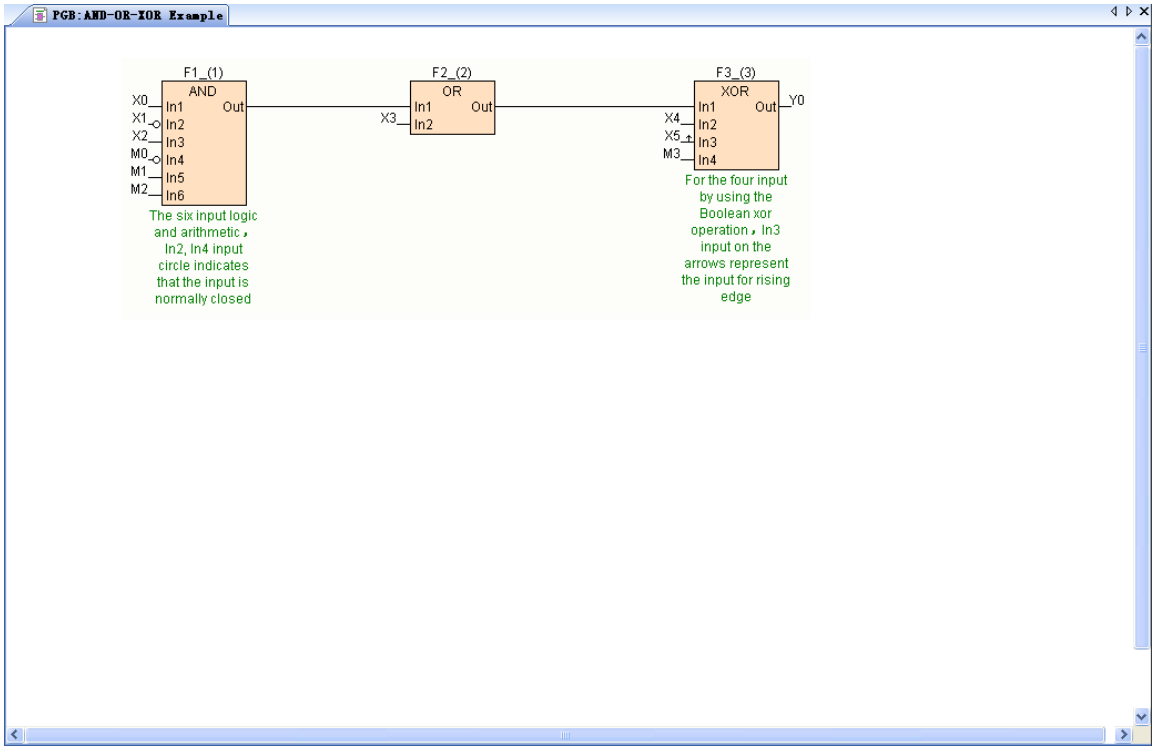
FBD function block programming

FBD work area

At FBD work area process FBD language edit, as add new page. add. delete instruction. connect instruction. change instruction execute sequence etc..

Top of FBD instruction are the function block number and it executed sequence number , as:"F3_(3)","F3" express the third function block ,"_(3)" press the function block executed sequence number is 3.

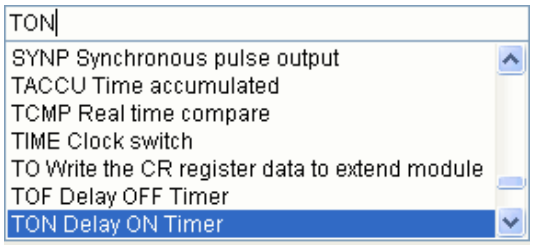
Change the connecting line between instruction system will automatic regulation the function block executed sequence, also may manual regulation the function block executed sequence.



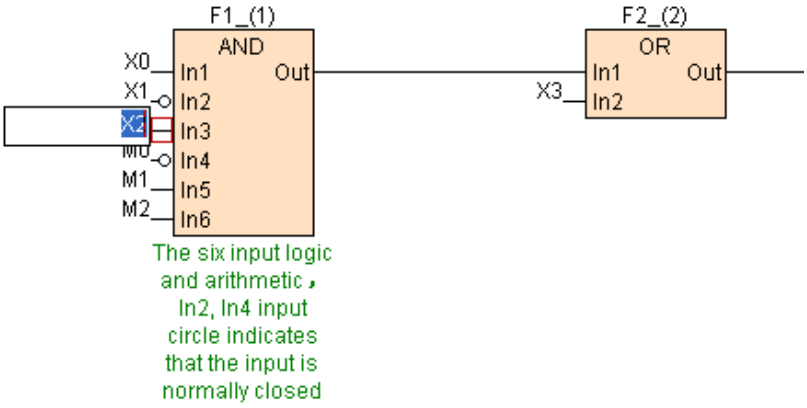
FBD instruction edit

Double click project manager instruction name in the instruction tree then may add the instruction to the network, also may use drag mode pull the instruction to the relevant position .

Support keyboard input instruction , input instruction name first character will automatic popup instruction input window, input instruction name then press enter may add instruction, may press "ESC" close the window at any time.



Mouse click or use keyboard move the gridlines to instruction position, press enter may input or modify the instruction parameter.



Via mouse drag box may select any instruction, may copy. cut. delete the instruction.

Double click instruction may open "instruction attribute" window, complete change instruction mode. set instruction forbidden. set timer time base etc.

operation .Double click input. output item may open " Item" window, via configuration mode input the item parameter.

FBD connect between instructions

FBD between instructions connected by enable flow line.

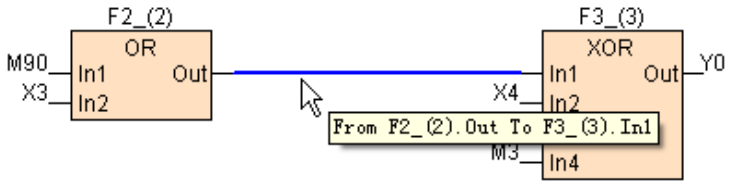
Click Tools bar  button , mouse change to  , mouse moce to instruction input or output item, if enable connect then mouse change to  , if disable connect mouse change to  .

Instruction output item can not connect to self input item, input item only a enable flow line input, output item may be output many enable flow lines .

Enable flow line connected will automatic change function block instruction executed sequence.

Between instruction may connect not use enable flow line , direct input instruction parameter.

Mouse click enable flow line will prompt the enable flow line detail connect information.

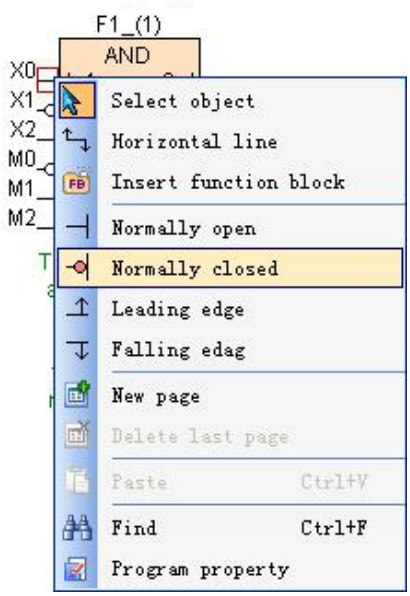


Change FB executed sequence

Mouse right button click instruction, callout right button menu click [Change FB executed sequence], input new executed sequence, press "Confirm" . If between FB connected via enable flow line , new executed sequence can not change enable flow line sequence.

Change instruction input status

When mouse move to the instruction input item will turn up a blue box, use mouse right button callout right button menu then may change instruction input item status (Normal open. Normal close. Rising edge. Failling edge).



Page edit

Click button add a new page.

Click button delete the last page, if the page have instruction then can not be deleted.

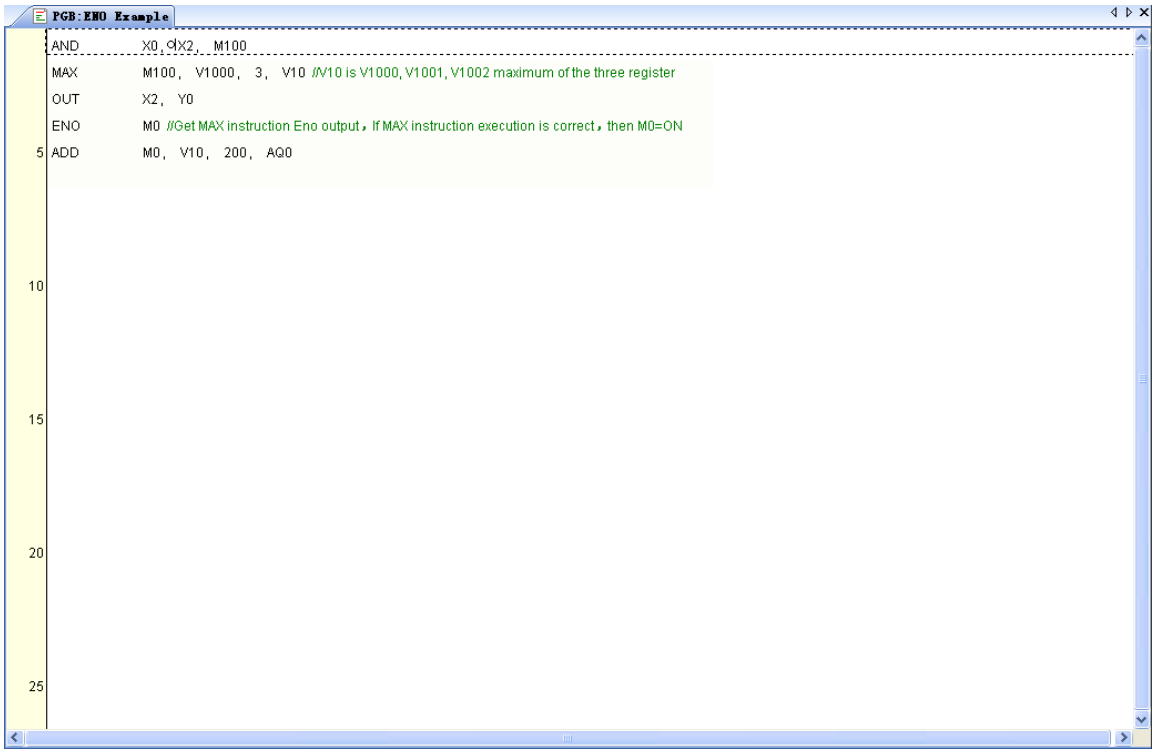
FBD comment

FBD language may write comment to each instruction, double click instruction may open "instruction attribute" window, at "Comment" box input the comment.

IL instruction list programming

IL work area

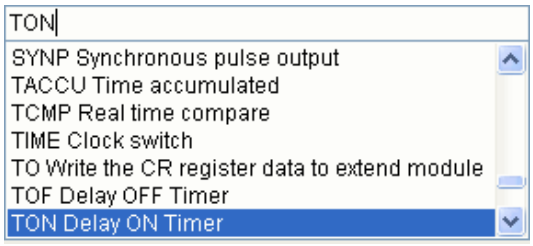
At IL work area process IL language edit, as add. delete instruction. move up or move down instruction etc..



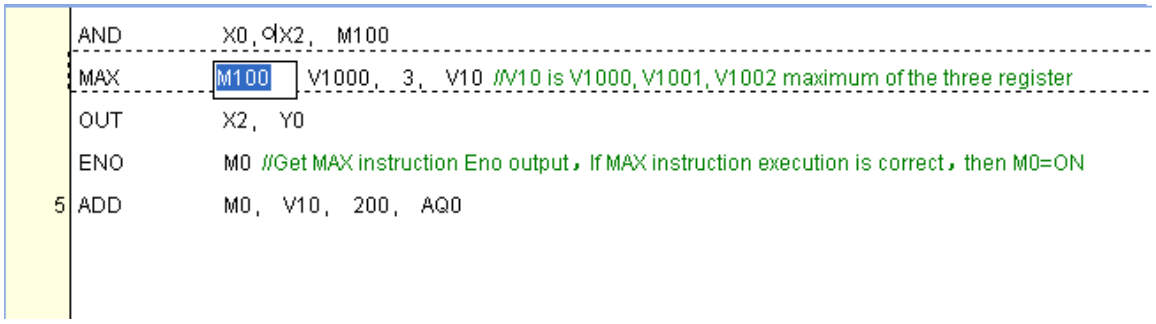
IL instruction edit

Double click Project manager instruction name in the instruction tree then may add the instruction to the tail of the program, also may use drag mode pull the instruction to the relevant position .

Support keyboard input instruction , input instruction name first character will automatic popup instruction input window, input instruction name then press enter may add instruction, may press "ESC" close the window at any time.



Mouse click or use keyboard move the gridlines to instruction position, press enter may input or modify the instruction parameter.



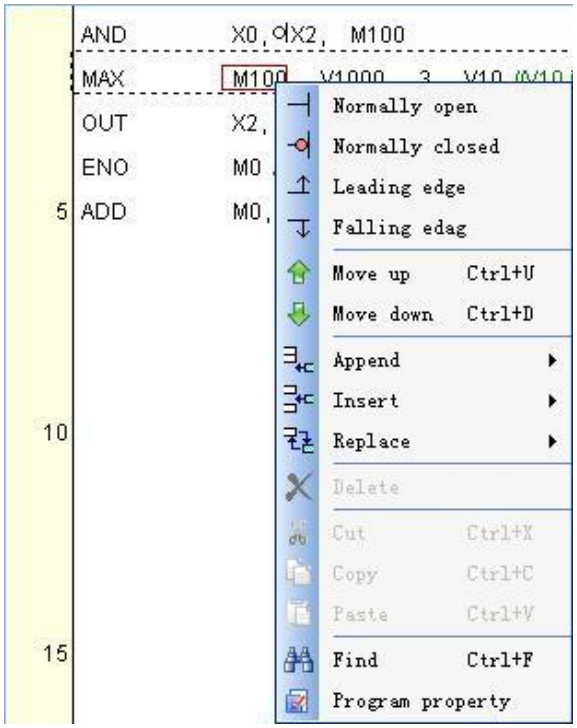
Via mouse drag box may select any instruction, may copy. cut. delete the instruction, also may move up. move down instruction executed sequence.

Double click instruction may open "instruction attribute" window, complete change instruction mode. set instruction forbidden. set timer time base etc. operation .Double click input. output item may open " Item" window, via configuration mode input the item parameter.

Change the instruction input status

When mouse move to the instruction input item, use mouse right button callout right button menu then may change instruction input item status

(Normal open. Normal close. Rising edge. Failling edge).



IL comment

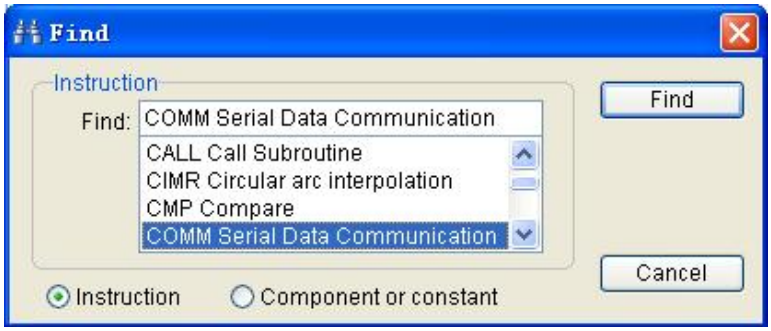
IL language may write comment to each instruction, double click instruction may open "instruction attribute" window, at "Comment" box input the comment.

Others

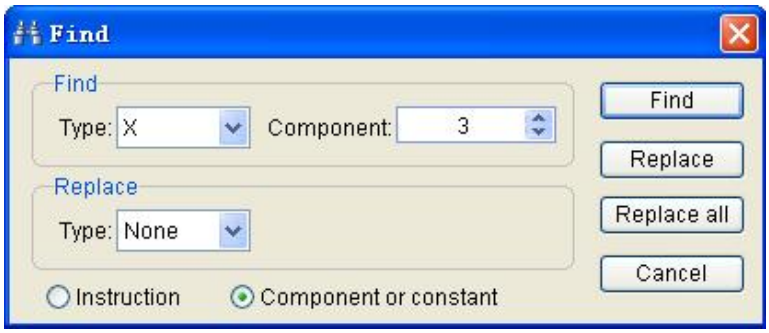
Find and replace

Via menu[Edit/Find]or shortcut key "Ctrl+F" open "Find" window, may find any instruction. component or constant , may replace component or constant.

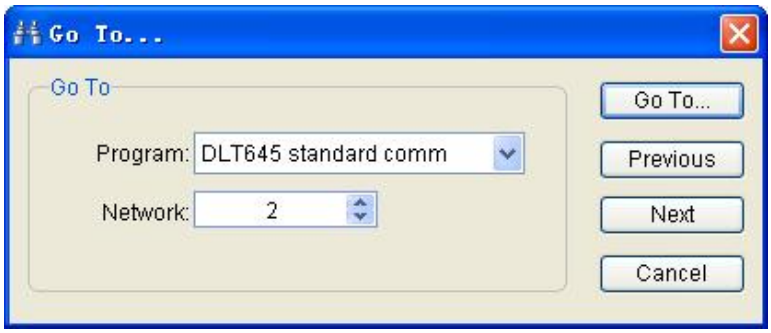
Find the instruction.



Find or replace the component or constant.

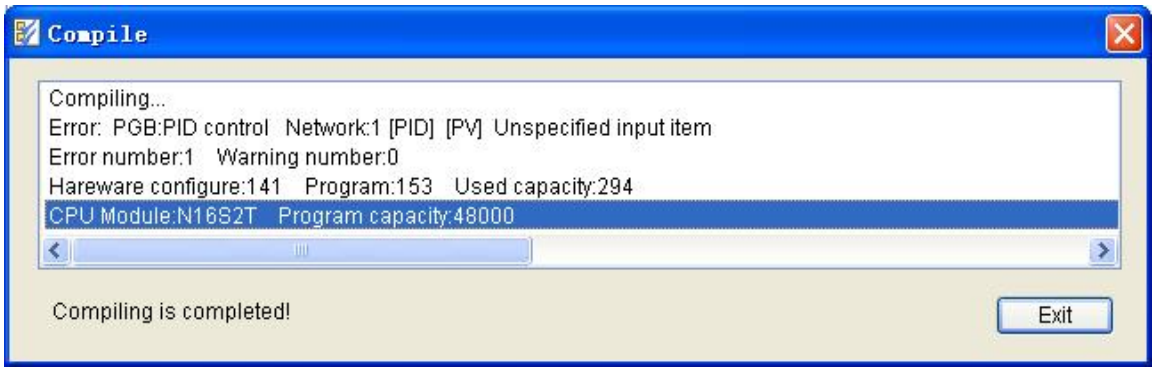


Via menu[Edit/Go To...] or shortcut key "Ctrl+G" open "Go To..." window, may realize fast position. LD language fast position to network,FBD language fast position to function block ,IL language fast position to line.



Compile program

During write the program may compiled the program at any time, to check the program whether or not have error. Via menu[Debug/Compile program] or click tools bar button start compile operation .Compile window as follows:



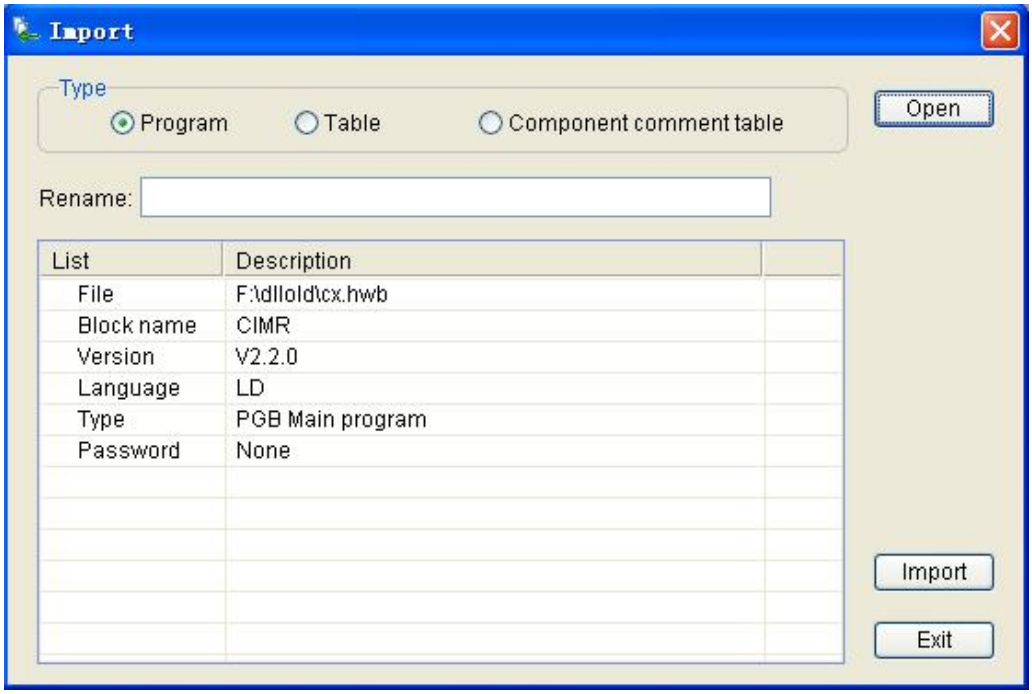
Download program or start simulator to executed program, system automatic start compile operation , only have not any error the program may be downloaded or started simulator to executed.

Compile window will list the error number and alarm number and them detail information of compile result .Double click error or alarm information may fast position the position in the program which generated the error or alarm , convenience user modify the program .

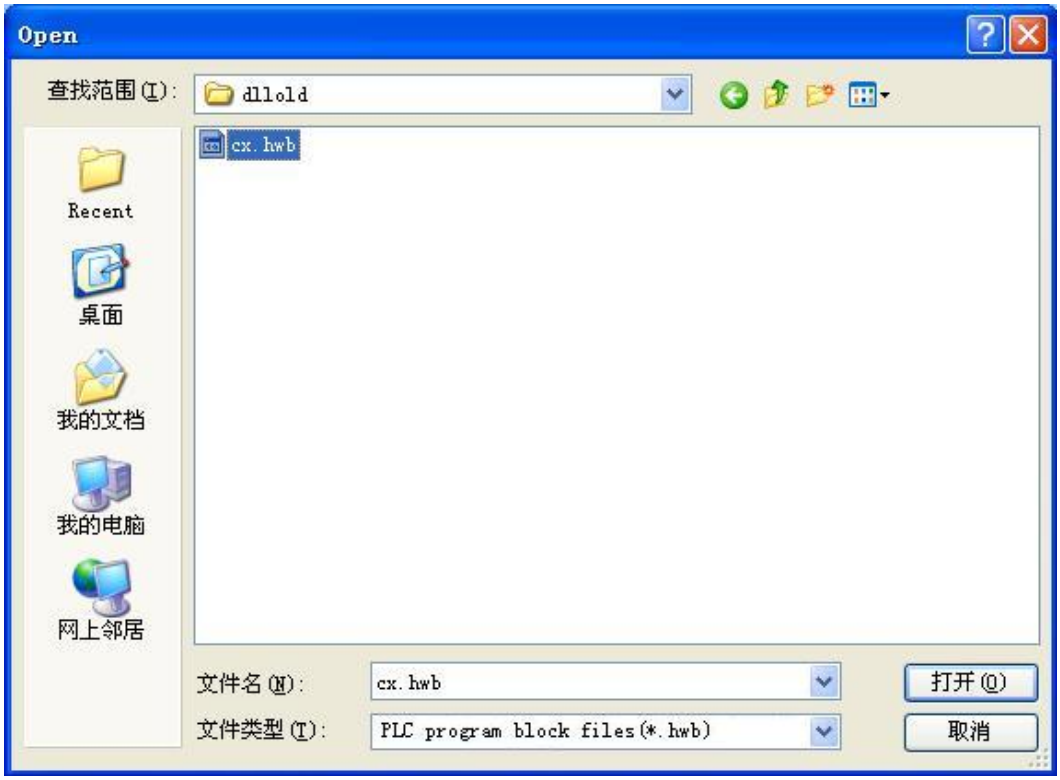
Error number is 0 express no error, alarm number is 0 express no alarm. Program error must be modified, alarm information may be ignored.

Program and table import

Via menu [File/Import] open "Import" window.



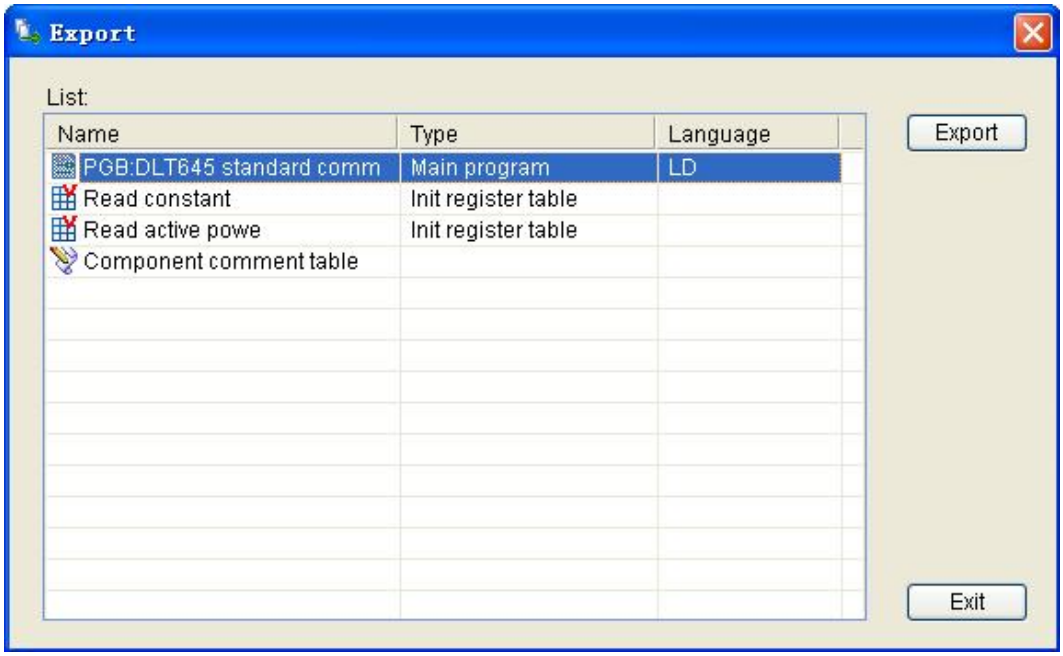
Click " program ". "Table". "Component comment table" select the imported file type ,click "Open " button select the imported file.



click "Import" button import the file to current program project.

Program and table export

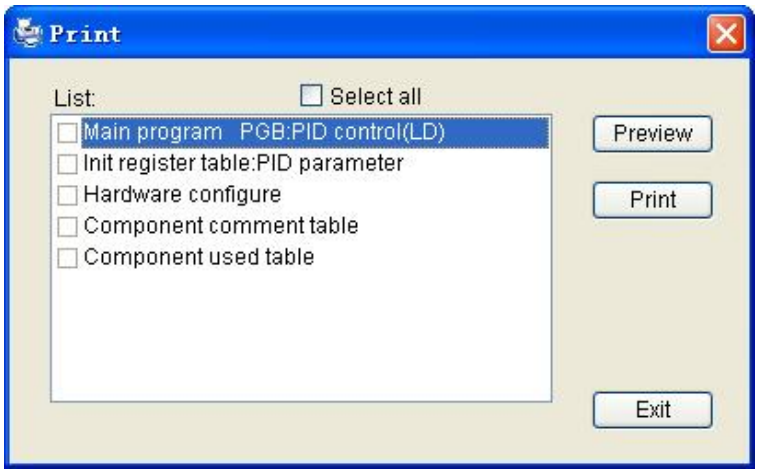
May export the current program project program block . table and component comment, convenience at other program project import to use. Via menu [File/Export] open "Export" window, select exported content and press "Export" button.



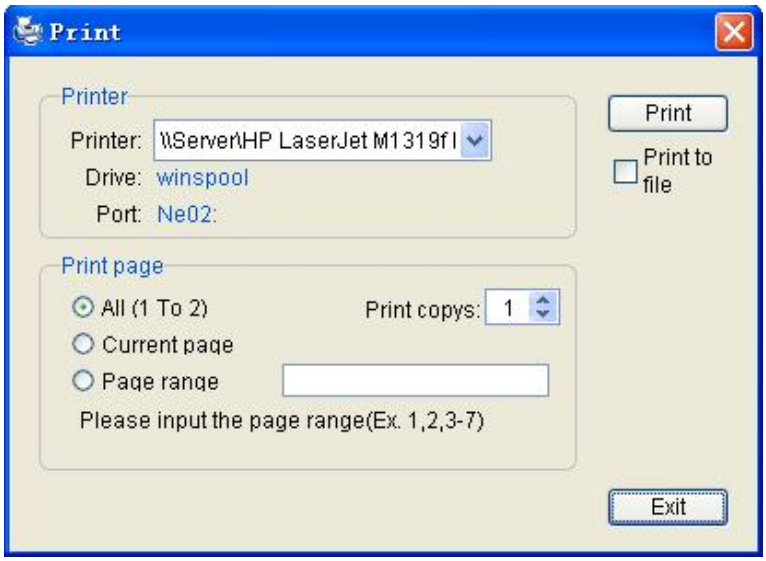
ote: if program block . table have set password protect, then exported file have the same password.

Print and preview

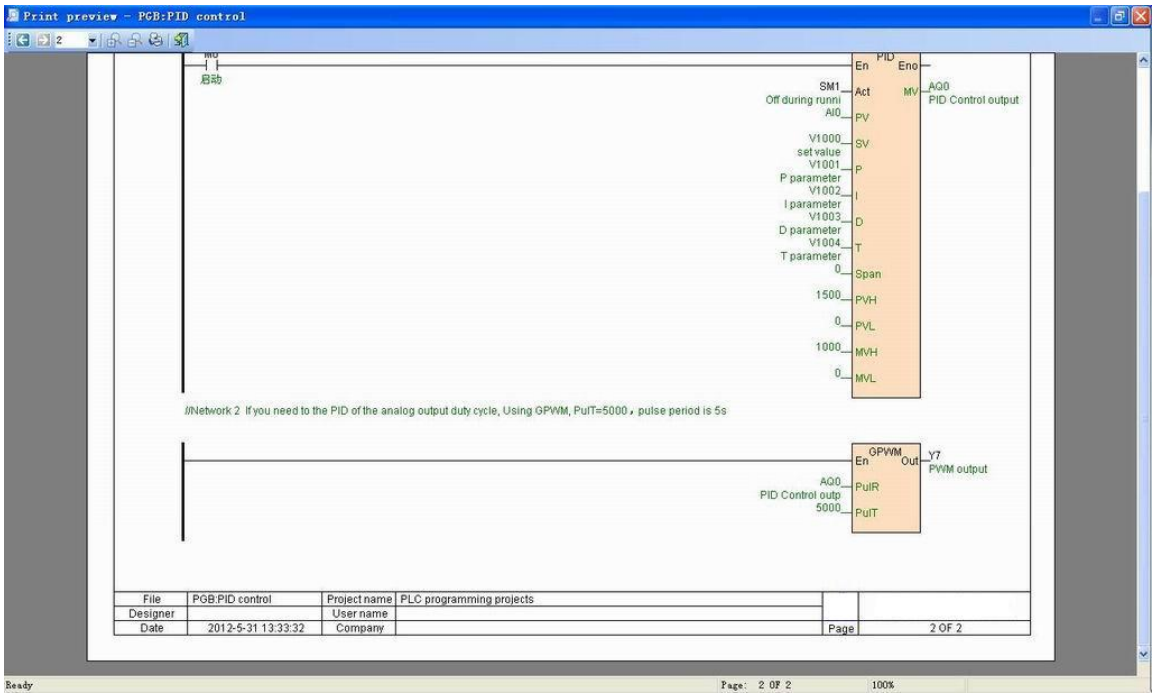
If you need print the program content via project file mode, via menu[File/Print]or click tools bar button ,open "Print" window.



Select the content will be printed,click "Print" button open " print attribute window, In the window may select print copies. pages etc. information, also may select "Print to file" output.



Via menu [File/Print preview] or click tools bar button preview the content will be printed.

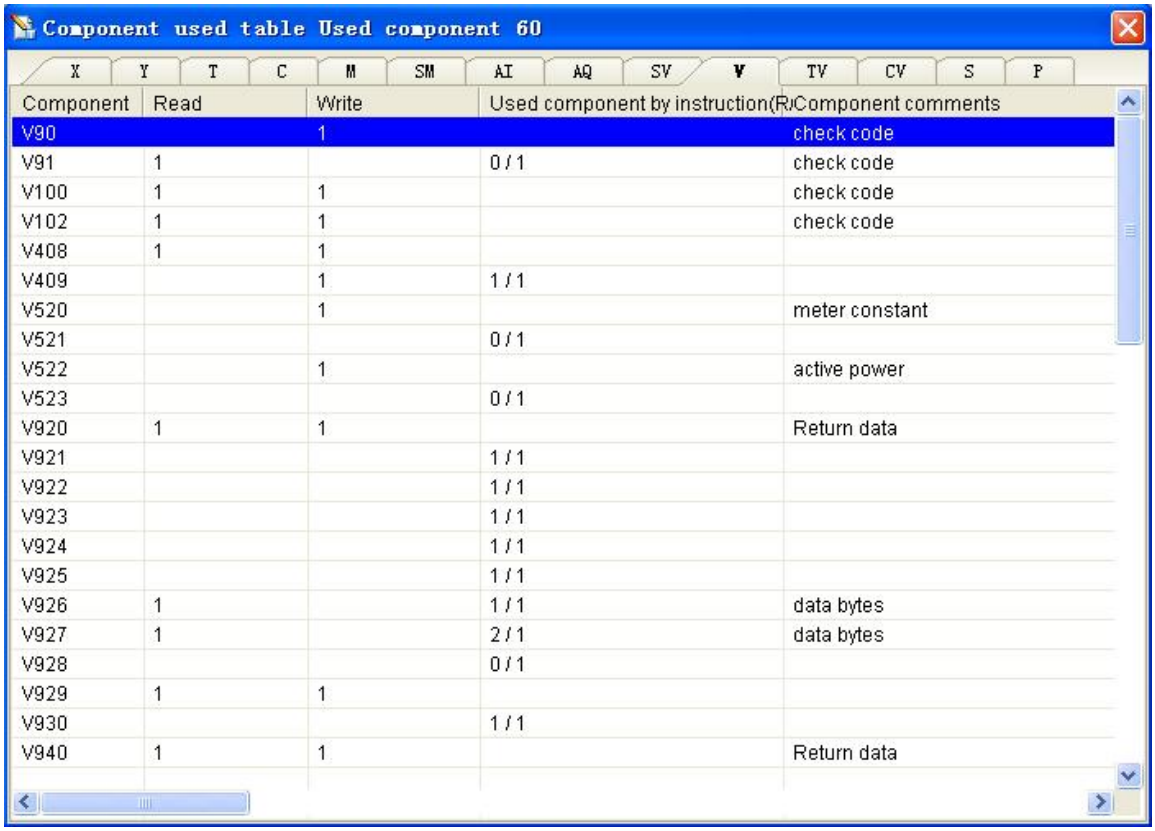


Component used table

"Component used table" list current program project all components be used, also detail statistics the component read times. write times and instruction automatic occupy continuous components (read/write) times.

Note: if "Write times" add

"instruction automatic occupy continuous components write times" greater than 1, then express the component have multiple output situation.

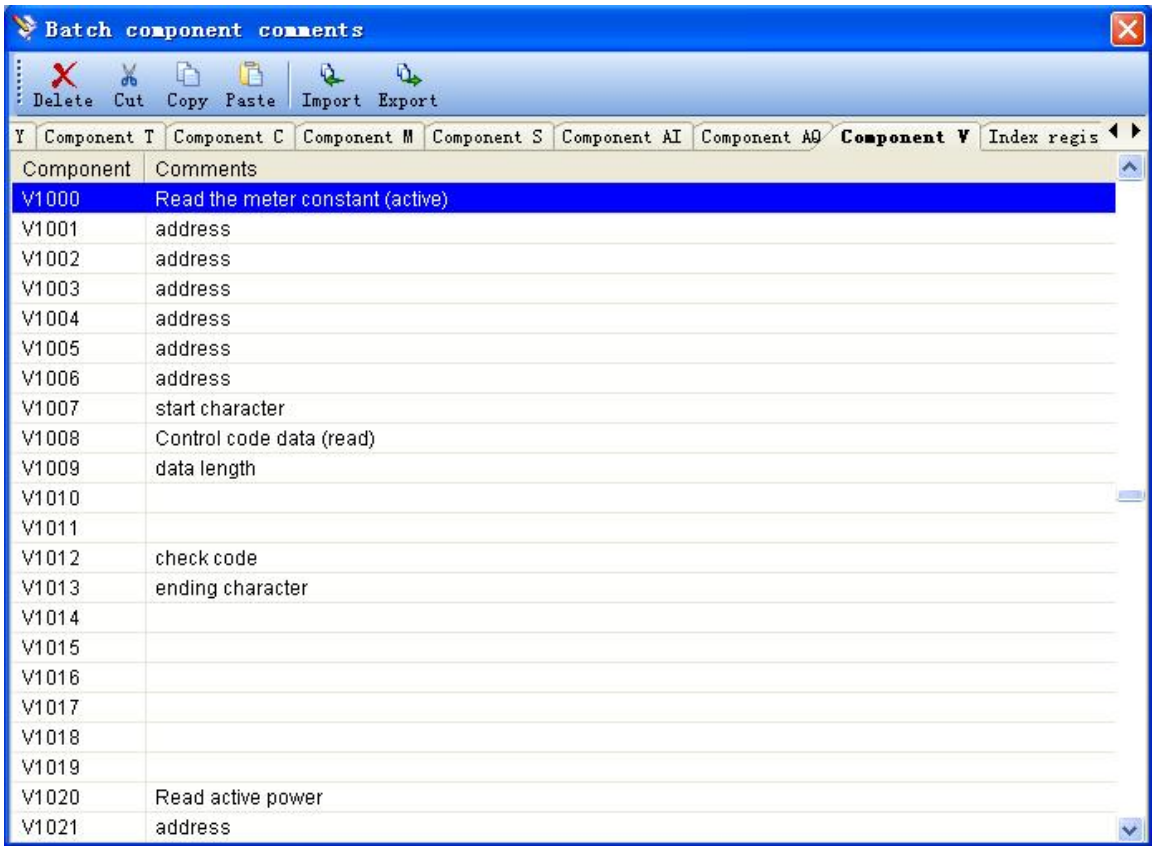


Component comment

PLC programming software supply abundant comment function, as: component comment. network comment. instruction comment. program block comment. table comment and program project comment etc., comment may be selected download to PLC, convenience in the future upload the program read and modify.

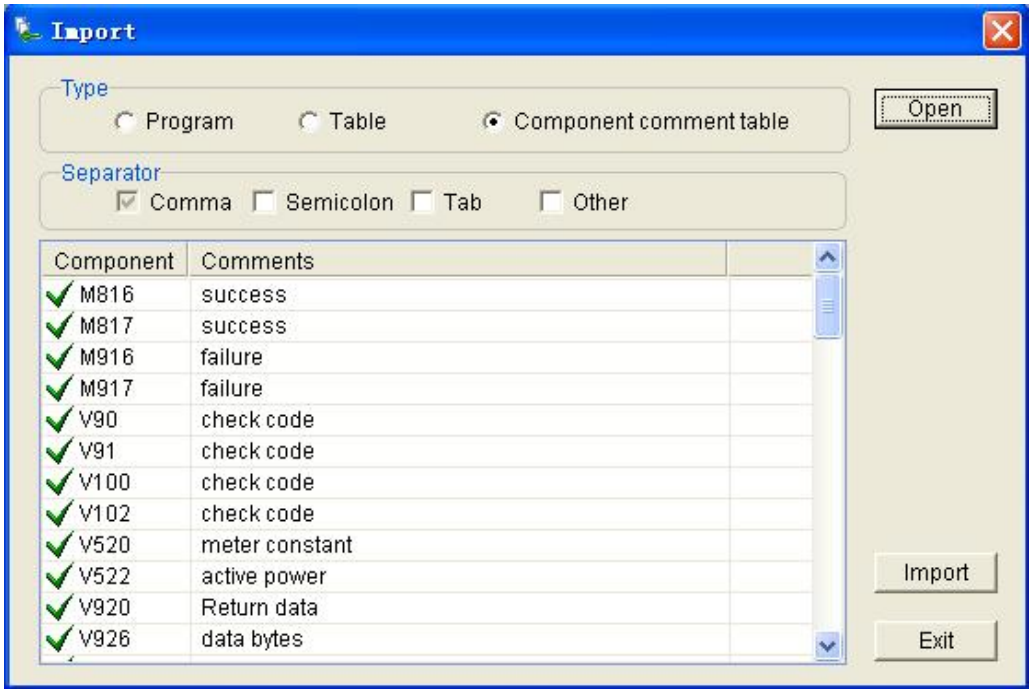
Component comment may use after the component follow "//" direct input, as: X0//motor start,X0 is component, "motor start" is the comment of X0.

Component comment may be batch edited, via menu [Tools/Batch component comment] open "Batch component comment" window.



Press "Export" button may export all component comment to file.

Press "Import" button may import component comment from external comment file.


Import
Type
Program
Table
Component comment table
Open
Separator
Comma
Semicolon
Tab
Other
Component
Comments
M816 success
M817 success
M916 failure
M917 failure
V90 check code
V91 check code
V100 check code
V102 check code
V520 meter constant
V522 active power
V920 Return data
V926 data bytes
Import
Exit

Simulate and online debugging

This section introduce the how to use the program software built-in simulator and the method of online debug the program.

Overview

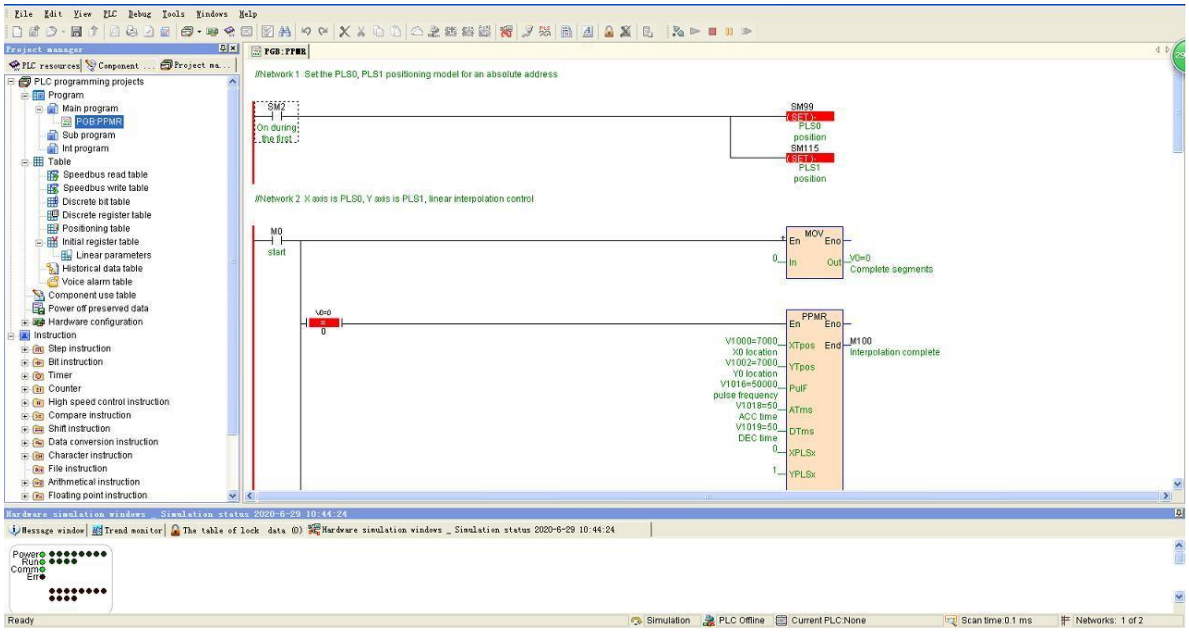
Program software built-in simulator (offline simulate), entirety realize the PLC program simulate running . When you are programming of after completed the program , can use the simulator simulate running the PLC program in entirely break away from PLC, in order to check up program executed whether or not correct, vastly reduce the debug time at local , reduce the debug difficulty , improve the debug efficiency.

Program software supply abundant debug tools , expedience online debug program, can search the whole PLC which connected the PLC, display all online PLC running status . fault status . RUN/STOP switch position. hardware configure . communication port parameter etc. detail information, can select any PLC process online monitor, look up all PLC component value or status.

Simulation environment

At simulating or online monitoring , the bottom of main windows will appear a "PLC hardware simulator windows". all divide into four page

windows : message windows. curve real time monitoring . locked data table . PLC hardware simulator windows, at the same time appear simulate tools bar at top left corner, cooperate right-hand button menu, make the simulate still more close to real environment.

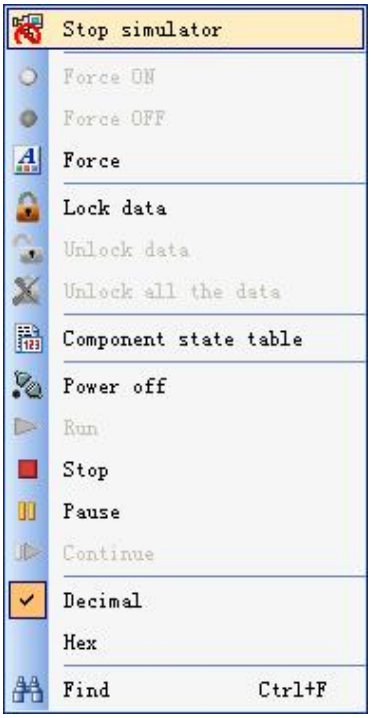


Simulate tools bar

Simulate tools bar use for control the power off. power on. start. stop. pause and continues operate of the simulator.

Right-click menu

At simulating or monitoring status, click right-hand button can callout right-click menu. Right-click menu support still further comprehensive simulate operation and others debug tools.



Stop simulator: finish current simulating status, return edit status.

Force ON: force the bit component status to ON of the mouse position.

Force OFF: force the bit component status to OFF of the mouse position.

Force: open " force " window, force component status or value.

Lock data: open " Lock " window, lock component status or value.

Unlock data: release the component locked data of the mouse position.

Unlock all the data: release all the component locked data.

Component status table : open " component status table ",can arbitrarily monitor all components status or value of PLC.

Power off/power on: simulate PLC power off/power on.

0. Start: control the simulator start running.

. Stop:control the simulator stop running.

2. Pause:control the simulator pause running.

3. Continue:control the simulator continue running.

4. Decimal: according to decimal display data value.

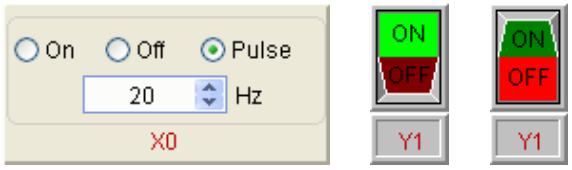
5. Hexadecimal :according to hexadecimal display data value.

6. Find: open " find " window, find component or instruction in the program.

Hardware simulate window

" Hardware simulate window " list the hardware configure of current program project , display the name of MPU and module and all X(DI)or Y(DO) channel status and AI. AQ channel value.

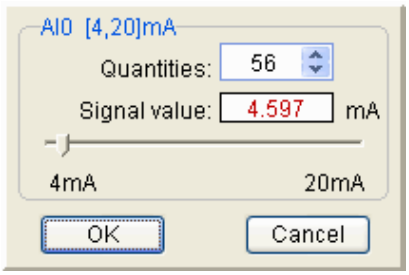
Click X or Y channel may be force the channel status.



Click AI or AQ channel may be open "AI/AO simulate window" to modify the channel value.

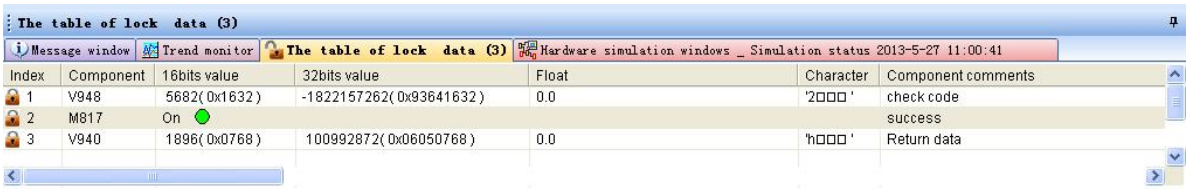
AI/AQ simulate window

Click " hardware simulate window " AI or AQ channel may be open " AI/AO simulate window ",window display the channel's signal type. signal value. signal value corresponding quantities or code value etc. , can modify the channel value(can modify AI and AQ in simulate status , only AQ can be modified in online monitor status).



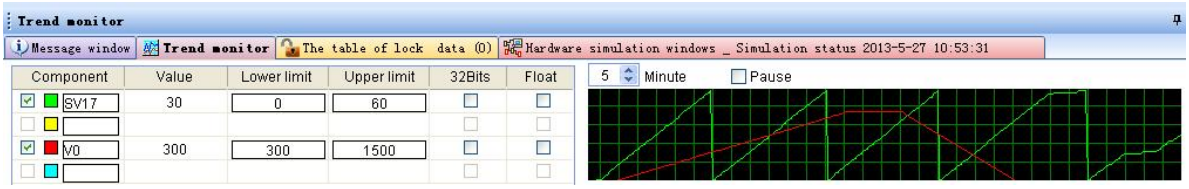
Data lock window

" data lock window " list all locked component and them status or value.



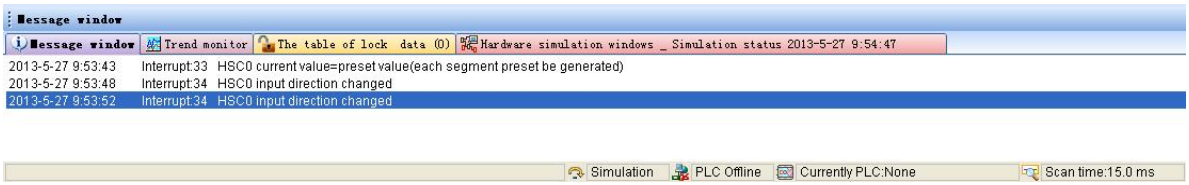
Real time curve window

" Real time curve window " according real time curve to monitor the change trend of register component value, expedience user process dynamic watch and debug some important parameter.



Message window

" Message " display the system message during simulating or debugging process, as interrupt message. system error message. communication error message etc., user can check the message at the window to be informed of current program running status.



Simulate operation

General steps of simulation

Start simulator , enter into simulate status.

In accordance with force the component, modify the component status or value, make program as far as possible executed in simulate the real local actual condition . In simulating process, can use " component status table ". " real time curve monitor " or "PLC hardware simulate " etc. numerous simulate debug tools for monitor the result of the program simulate running, to checking the result whether or not correct of the program executing .

If the program have communication instruction , then may be start " communication simulator " simulate the return data from slave device.

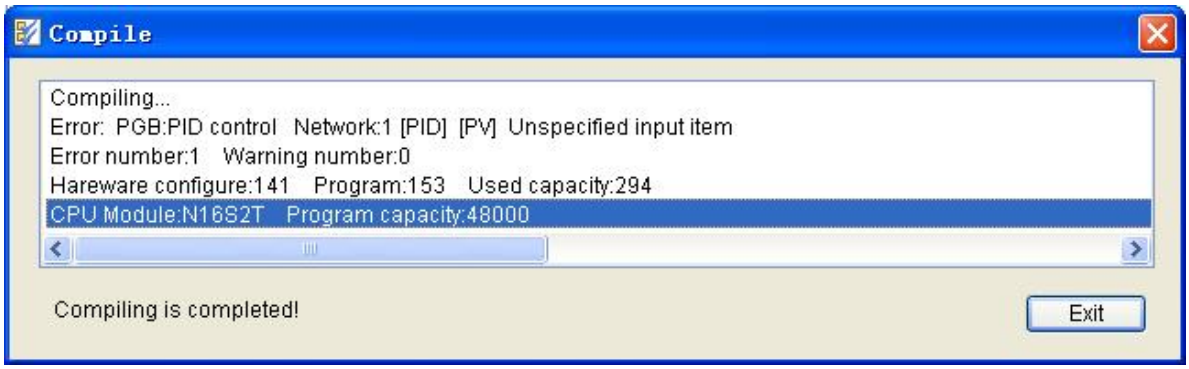
If the program have interpolation etc. motion control instructions, then may be start " interpolation simulator " according to diagram the result of simulate interpolation output.

Simulate PLC power OFF/power ON, check the program still can normal work after the power off.

Stop simulator ,return to edit status. If the result of the program not correct , then simulate executing after modify the program.

Start simulator

Click menu [debug/ start simulator] or click tools bar button to start simulator ,System auto compile the current project ,if there are error or alarm after the program compiled, if there are error or alarm after the program compiled, will popup the windows as follows.



List all error and alarm information in the listing frame , double click the error or alarm information , program will be accurate fixed position the location which generated the error or alarm , expedience user modify the program.

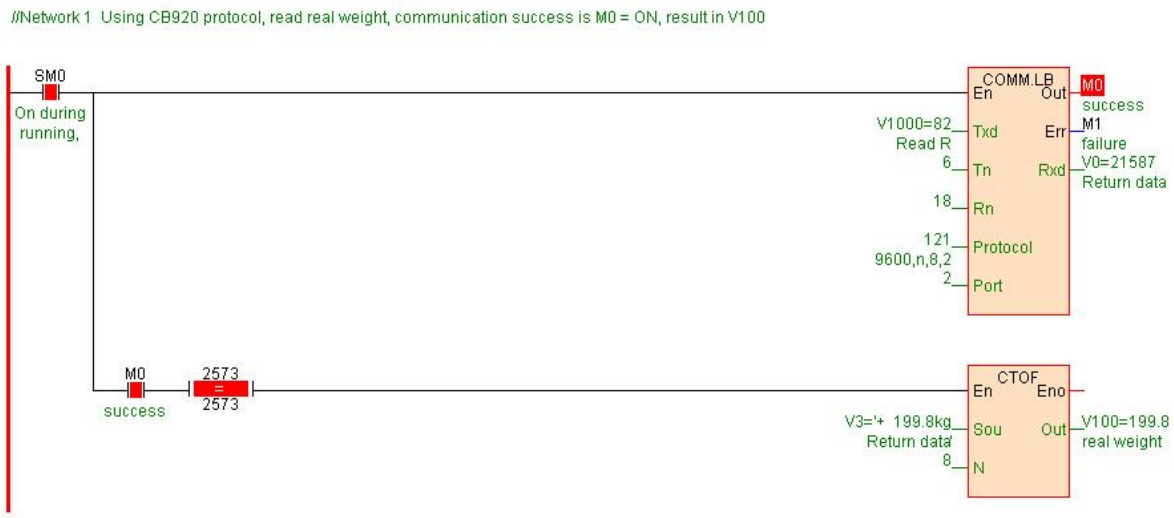
If there is error during compiling the program , simulator can not be started , until user modify the program have not any error and then the simulator can be started .

If there is no error or only alarm during compiling the program , simulator can be started . If there is alarm information , suggest you modify the program until there is no any alarm information and then start simulator .

After simulator started , programer software enter into offline simulate status . At simulate status , can not modified the program , only after stop simulator and return to edit status , just can be modified the program

omponent monitor

At simulate or online monitor status , may be through the program edit area . component status table . hardware configure windows etc. monitor the component status and value , and the instruction execute situation . Below is the program edit area which is at simulate or online monitor .



When the instruction frame is red, express the instruction is executing correct ; when is blue, express the instruction did not executed or executed error(the instruction parameter error ,as DIV instruction divisor is 0).

Register will display it current value , may be via menu [check\decimal. hexadecimal] change the register display model.

Floating point number or character format component will automatic display according to the defined of the instruction item . As up drawing :CTOF instruction, Sou input is character , so character display V3='+ 199.8kg',Out item output is floating point number , so floating point display V100=199.8.

Constant open switch and the compare switch have red diamonds , express the switch is connected , otherwise not connected. Note: the status of constant close and constant open is opposite ; rising edge and descend edge because is edge ,only generate edge connect once, rest time not connected.

The red frame of the component , express the status of component is ON, otherwise is OFF.

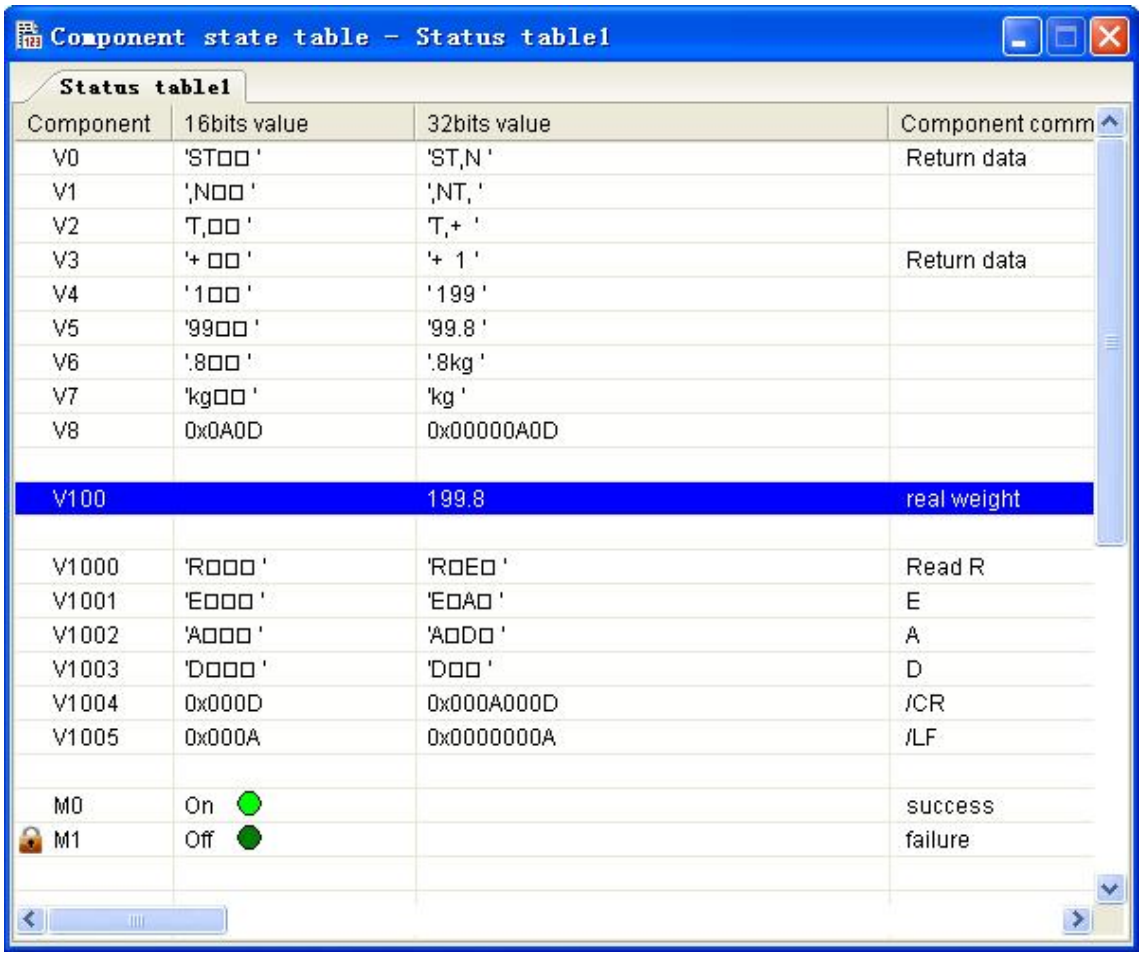
If component with  , express the component data locked.

Double clock the component can be forced the component status or value.

Component status table

" component status table " can monitor current status or value of the all component of the PLC, maximum may build 10 status tables, allow display the register value according to different data formation. Only usable in simulate or online monitor .

Click menu [debug/component status table] or click tool bar  button, open " component status table "



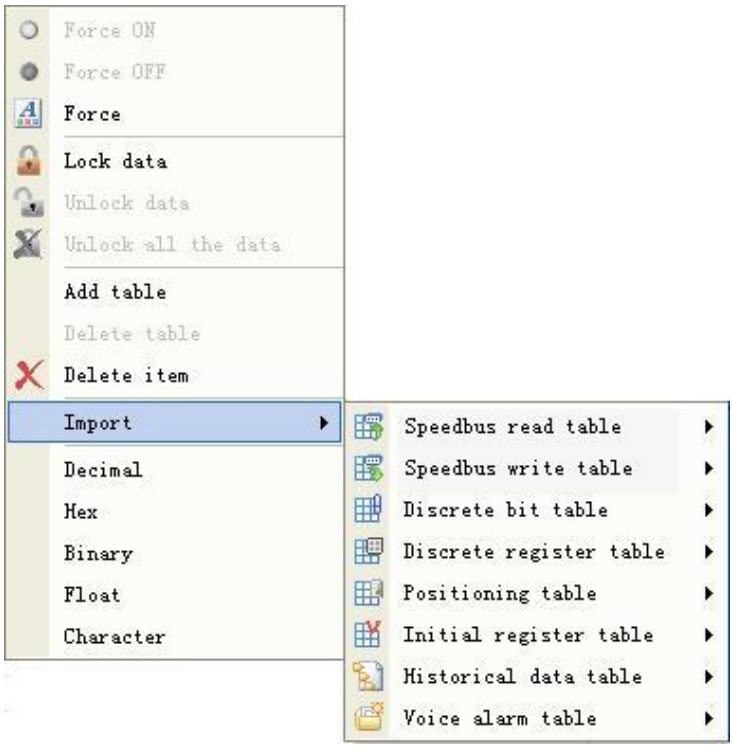
Click " component " bar nether blank space, At input frame input the component or component range, e.g. :as up drawing input V0-8. V100. V1000-1005. M0.

For register component display is "16 bit register value " and "32 bit register value ".Note:"32 bit register value" column display border upon 2 continuous registers value,as up drawing first line V0 component "32 bit register value" is V1V0 value.

If component with , express the component data locked.

Double clock the component can be forced the component status or value.

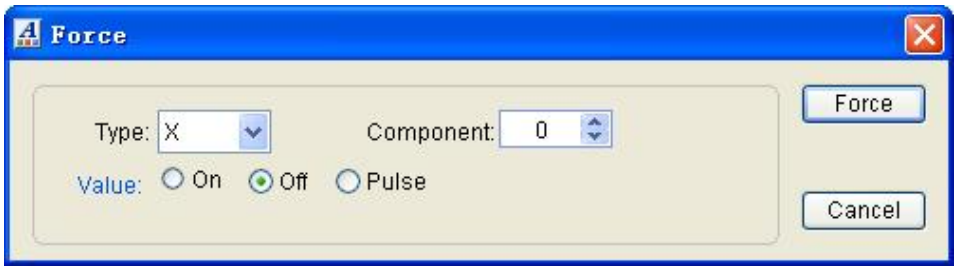
Right-hand button click may be popup the right-hand menu , via right-hand menu may be forced . locked . release locked. add or delete status table. to lead all kinds of data table component operation , may be change the display format of the register(include decimal base. floating point. character) etc. .



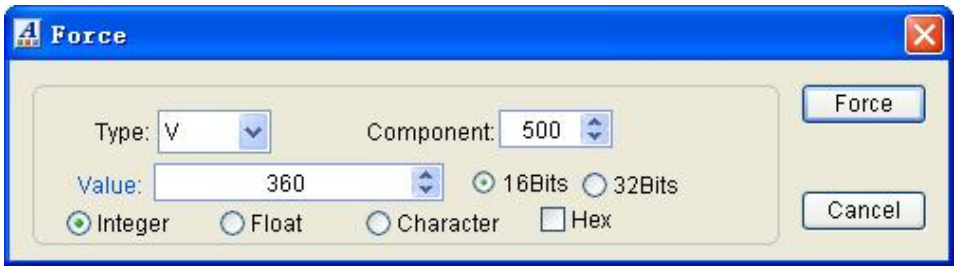
Force

Force change component status or value.

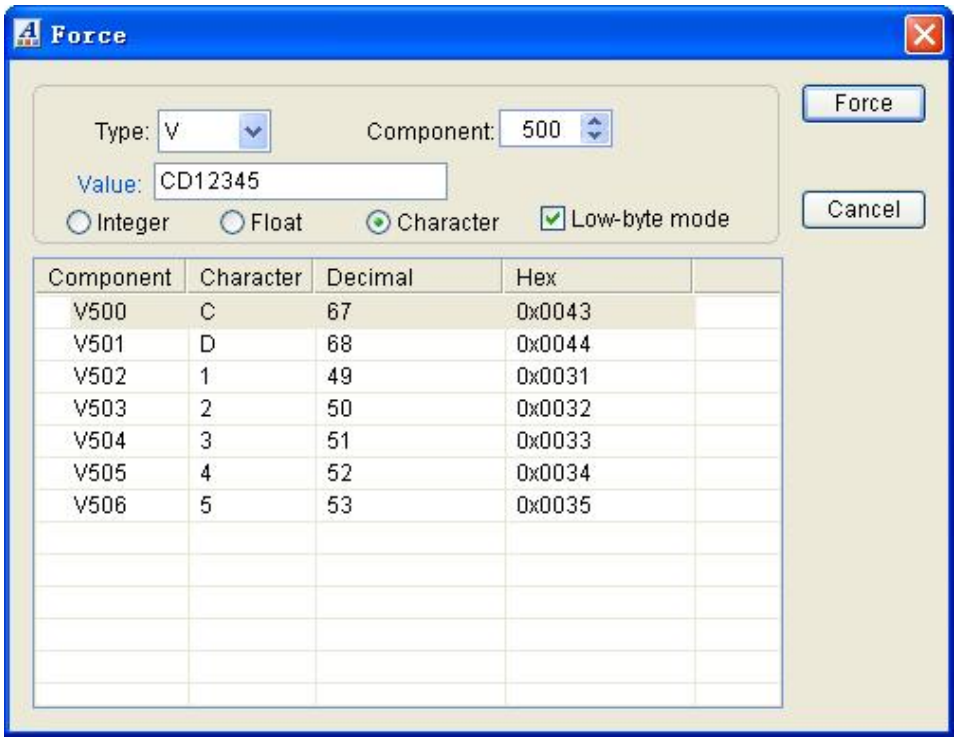
Bit component be forced: at " type " select bit component type , at " component " input component number, select ON or OFF status, if X component is external high speed pulse input point then can select input pulse frequency value, click " force " button.



Register component be forced: at " type " select register component type , at " component " input component number, input the value be forced, may be input 16 bit integer . 32 bit integer. floating point number or character,"HEX" selected express input hexadecimal integer , click " force " button .



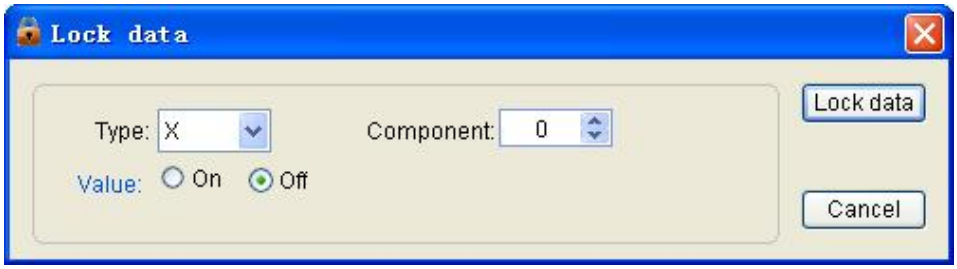
Character input as follows. each register store two character , when select " low byte mode ",each register only store one character .



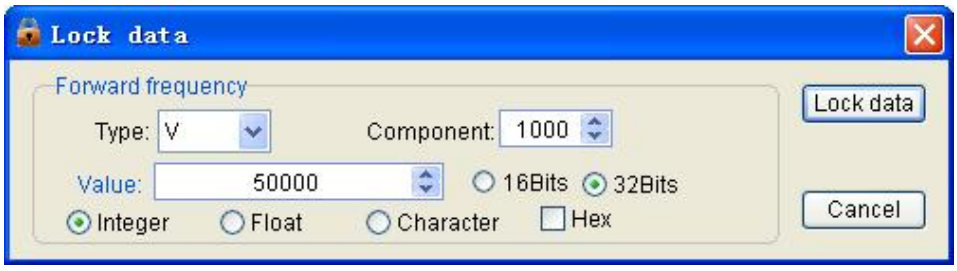
Lock data

Status or value of the component be locked, make component value not change, until release locked.

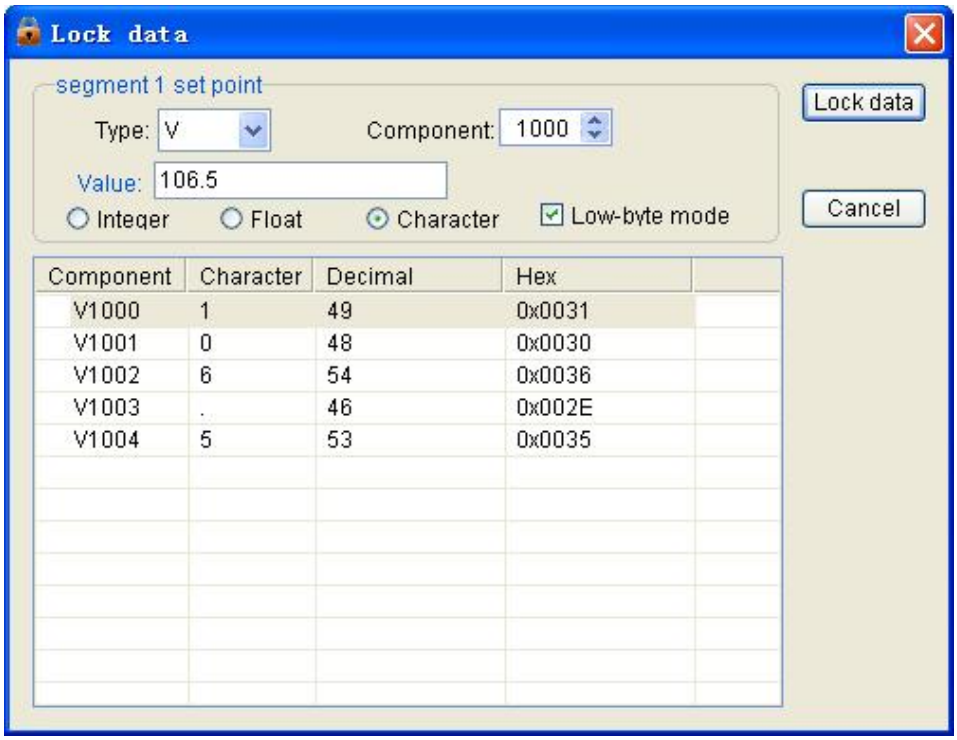
Bit component be locked: at " type " select bit component type , at " component " input component number, select ON or OFF status,click " force " button.



Register component be locked: at " type " select register component type , at " component " input component number, input the value be locked, may be input 16 bit integer . 32 bit integer. floating point number or character,"HEX" selected express input hexadecimal integer , click " force " button .



Character input as follows. each register store two character , when select " low byte mode ",each register only store one character .



Difference between force and lock:

1. Force only assign to component , program arithmetic output. external device communication input etc. will change the component status or value.

2. Lock fixed the status or value of the component , whether program arithmetic output. external device communication input etc. will not refresh it ,until release locked.

3. At on line monitor , can not force external input component status or value (such as digital input X and analog input AI). May be locked the status or value of external input component.

[Note]

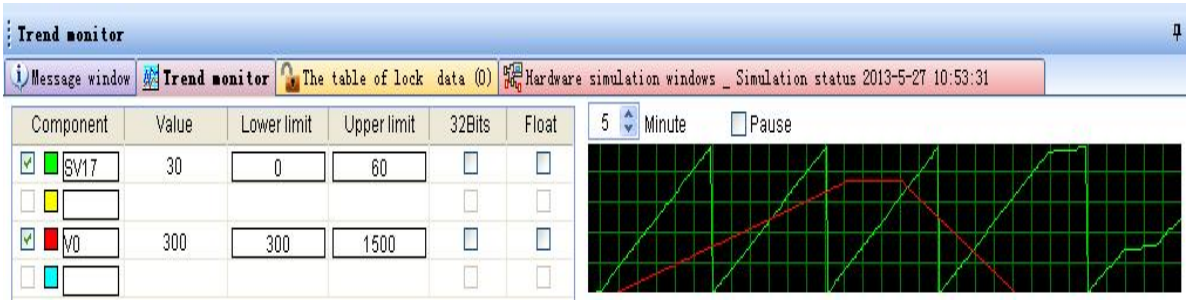
1. Please use the lock data function carefully, lock data might be produce unforeseeable effect.

2. If locked data in PLC, will generate error code 142, be "SV3=142 have locked data ", to remind user pay attention to it.

3. If really need use lock data function , expedience handle location problem, please release the locked data after the location problem handled finish.

Real time curve

At simulate or PLC on line monitor status, real time change of the register component (such as V. AI. AQ etc.) value may be use real time curve mode visualize expression it ,expedience user proceed dynamic observe and debug to some important parameters .



1. At " component " input the register component to be monitored.

2. At " upper limit value " and " lower limit value " input the register upper limit value and lower limit value .

3. Tick the leftmost select box, express draw real time curve of the component.

4. Register component default 16 bit integer ,if 32 bit integer must be tick "32 bits" select box, if floating point number must be tick " floating point number " select box.

5. " Pause " option pause the curve drawing. during pause, may be use left mouse button click the point of the curve to read the point value .

Power off simulate

All control system or device all might be power off, strong control program should consider the device power off factor fully , make sure the program still can proper functioning after power off then power on.

Click simulator tool bar  button ,simulate PLC power off , the program stop executing.

Click simulator tool bar  button ,simulate PLC again power on , the program again start executing.

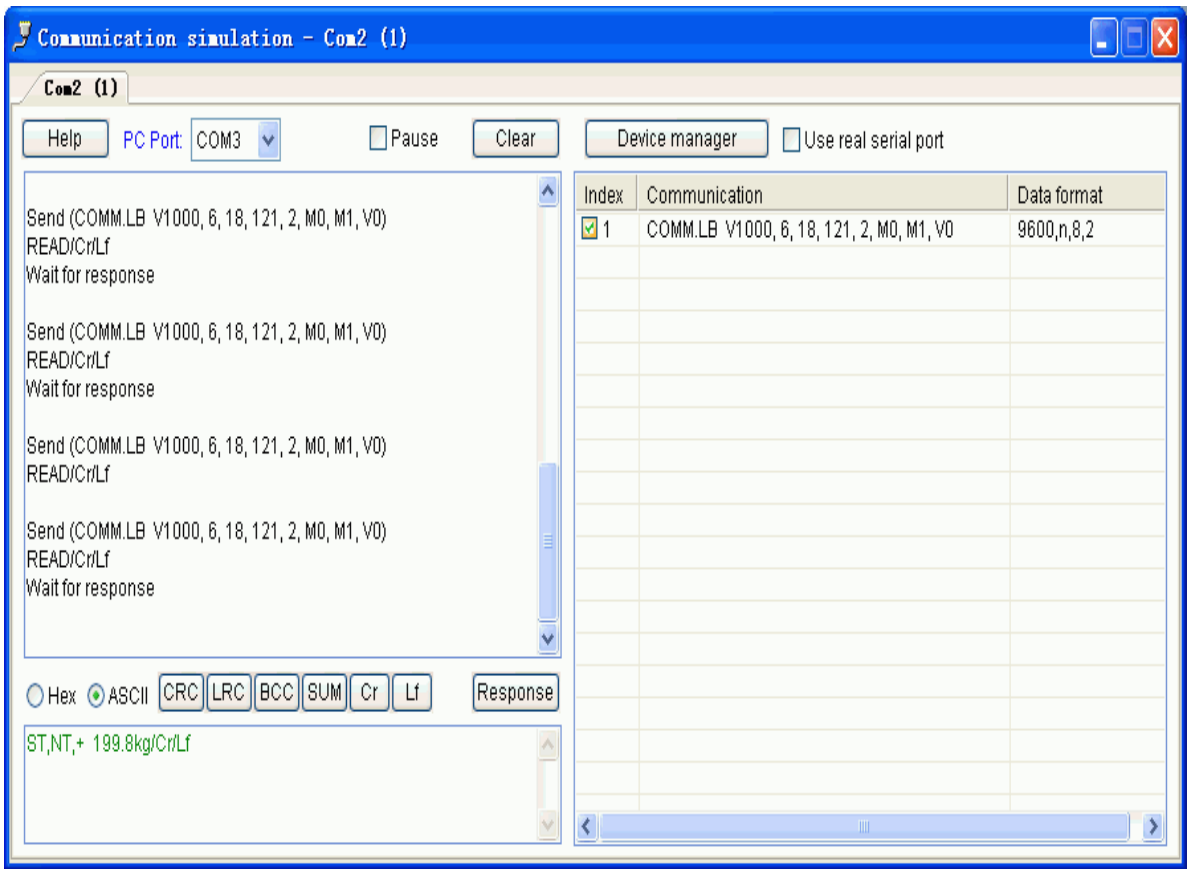
Via simulate power off/power on procedure, may be test program whether or not proper functioning after power off then power on, may be check parameter whether or not lost etc. problem.

Communication simulator

" Communication simulator " is a simulation tool be exclusively used in debug the communication instruction. It may manual operation simulate input response information from slave, also my be use the computer reality serial port communicate with salve , reality simulate PLC execute communication instruction process and handle the data return from slave.

Start communication simulator

At simulation status , click menu [debug/communication simulator] or click tools bar  button to start " communication simulator ", default " manual input response information from slave" mode.



Upper left message frame display the executing communication instruction. the instruction sent command frame data. slave response data etc. , message frame scroll display all communication instruction executed and generate the send and receive data . press " clear " button clear all message , tick " Pause " stop refresh communication instruction executed and generate the send and receive data .

Left lower frame use for manual input return response information from slave.

Right list all communication instruction using the same communication port , add ☑ identification at the executing communication instruction, double click instruction may fixed position the instruction location at the program.

Each communication port used by communication instruction will generate a alone page, page title display the communication port number and number of instructions which using the communication port.

Manual input slave response information

When upper left message frame display " wait response" , at left lower input frame input response data frame from slave (data content please according to the communication protocol specification of the slave),press " enter" ,response data will be automatic wrote to communication instruction output register .

"Hex". "ASCII" use for select Hexadecimal or ASCII code mode input and display data .

"CRC". "LRC". "BCC" and "SUM" button use for calculate the check code of the input data.

"Cr" input enter symbol,"Lf" input line break.

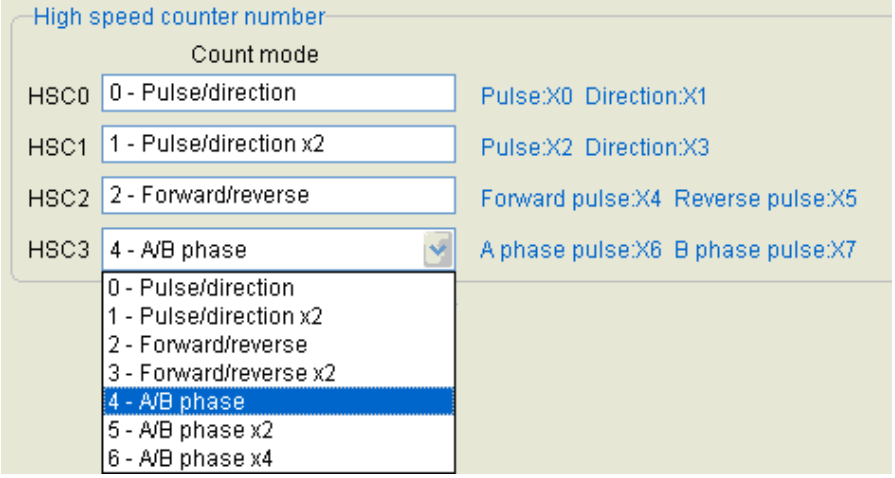
Use reality serial port communicate with slave

Tick " use reality serial port communicate with slave " enter use computer reality serial port reality communicate with slave mode, communication instruction send command frame to slave via computer serial port , also receive response data from slave, response data automatic wrote to communication instruction output register. Reality send and receive data all will be scroll displayed in the message box.

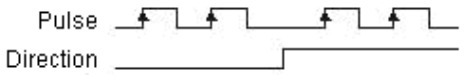
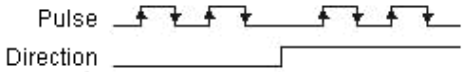
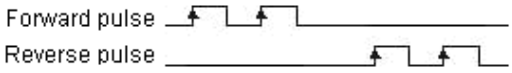
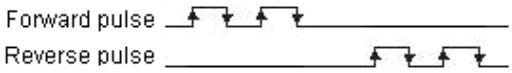
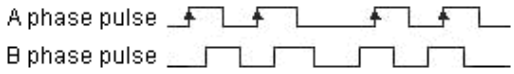
"PC port " listbox list the total serial port of the computer , out of select the reality serial port connect to the slave .

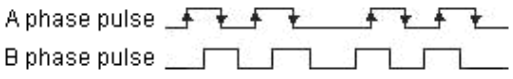
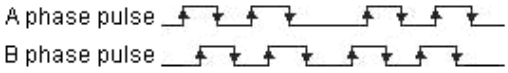
High speed counter simulate

High speed counter support:pulse/direction . positive/negative pulse . A/B phase pulse input model, support 1. 2. 4 frequency multiplication counting model .

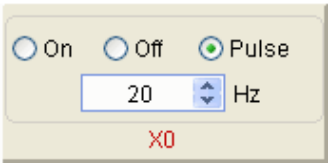


[High speed counter mode and pulse oscillogram]

Counting mode		Pulse oscillogram	
Mode	**Frequency multiplication**	**Increase count**	**Decrease count**
0 --pulse/direction	1		
1 --pulse/direction	2		
2 -- foreward/reversal	1		
3 -- foreward/reversal	2		
4 -- A/B phase	1		

Counting mode		Pulse oscillogram	
Mode	**Frequency multiplication**	**Increase count**	**Decrease count**
5 -- A/B phase	2		
6 -- A/B phase	4		

High speed counter channel express by HSCx , each channel use 2 high speed pulse input point .Can force input frequency of the high speed pulse input point.



Method of simulate pulse input, use HSC0 channel for example.

A. Pulse/direction (X0 is pulse singal,X1 is direction singal): force X0 pulse frequency to 20Hz,X1=OFF, now is increase counting

force X0 pulse frequency to 20Hz,X1=ON, now is decrease counting

B. positive/negative pulse(X0 is positive pulse,X1 is negative pulse): force X0 pulse frequency to 20Hz,X1=OFF, now is increase counting

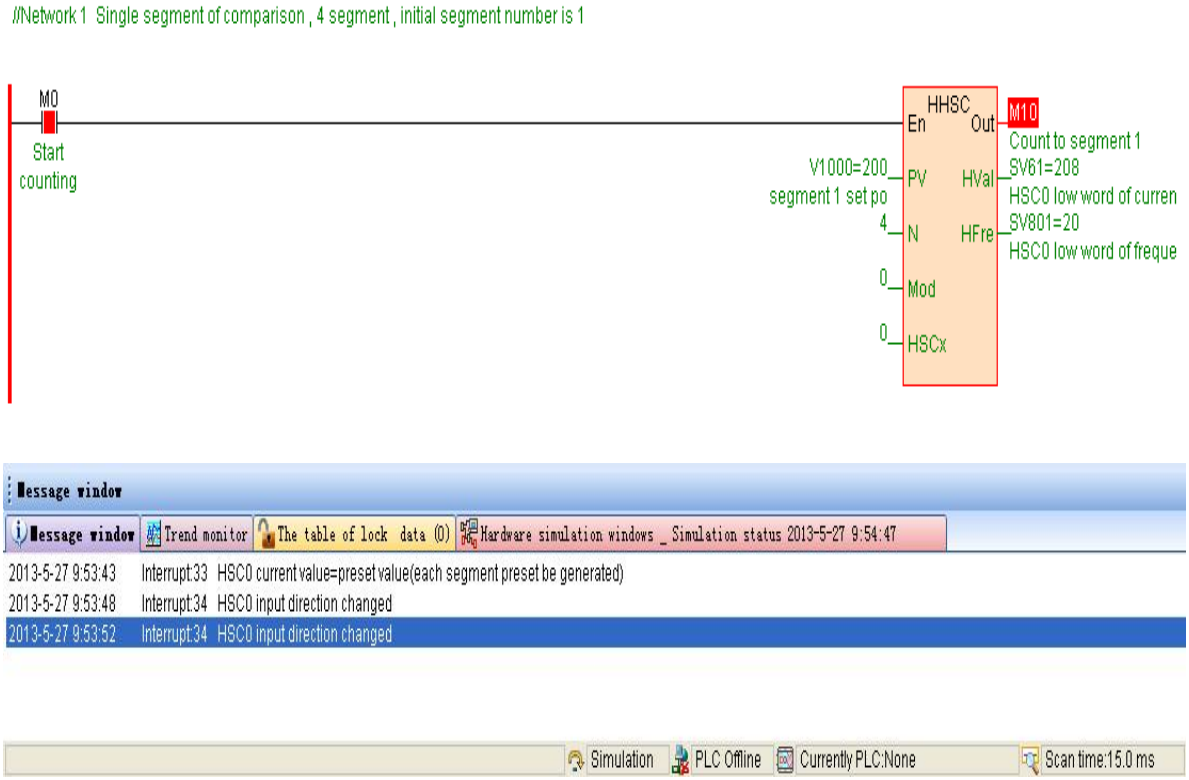
force X0=OFF,X1pulse frequency to 20Hz,now is decrease counting

C. A/B phase pulse(X0 is A phase pulse,X1 is B phase pulse): first force X0 pulse frequency to 20Hz, then force X1pulse frequency to 20Hz ,now is increase counting (A phase pulse before)

first force X1pulse frequency to 20Hz , then force X0 pulse frequency to 20Hz,now is decrease counting (B phase pulse before)

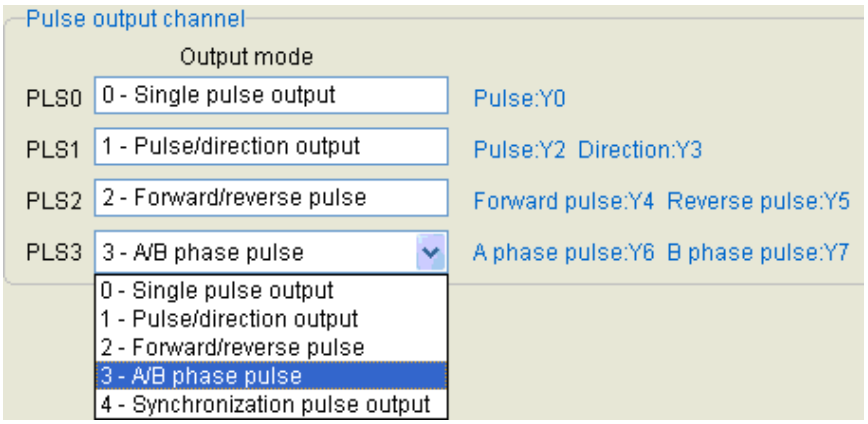
During high speed counter simulate execute process will generate corresponding system interrupt, as follows .

Program example: Download



Pulse output simulate

High speed pulse output support :single pulse. pulse/direction . positive/negative pulse . A/B phase pulse . synchronization pulse output, total 5 output mode, high speed pulse output channel express via PLSx , each channel use 2 high speed pulse output point.



Method of simulate pulse output, use PLS1 channel for example .

A. Single pulse output(Y2 is pulse signal): Y2 blink during pulse output ; no pulse output Y2=OFF.

B. Pulse/direction output (Y2 is pulse signal,Y3 is direction signal): Y2 blink during positive pulse output,Y3=OFF.

Y2 blink during negative pulse output,Y3=ON.

no pulse output Y2=OFF,Y3=OFF.

C. Positive/negative pulse output (Y2 is positive pulse signal,Y3 is negative pulse):Y2 blink during positive pulse output,Y3=OFF.

Y2=OFF during negative pulse output,Y3 blink.

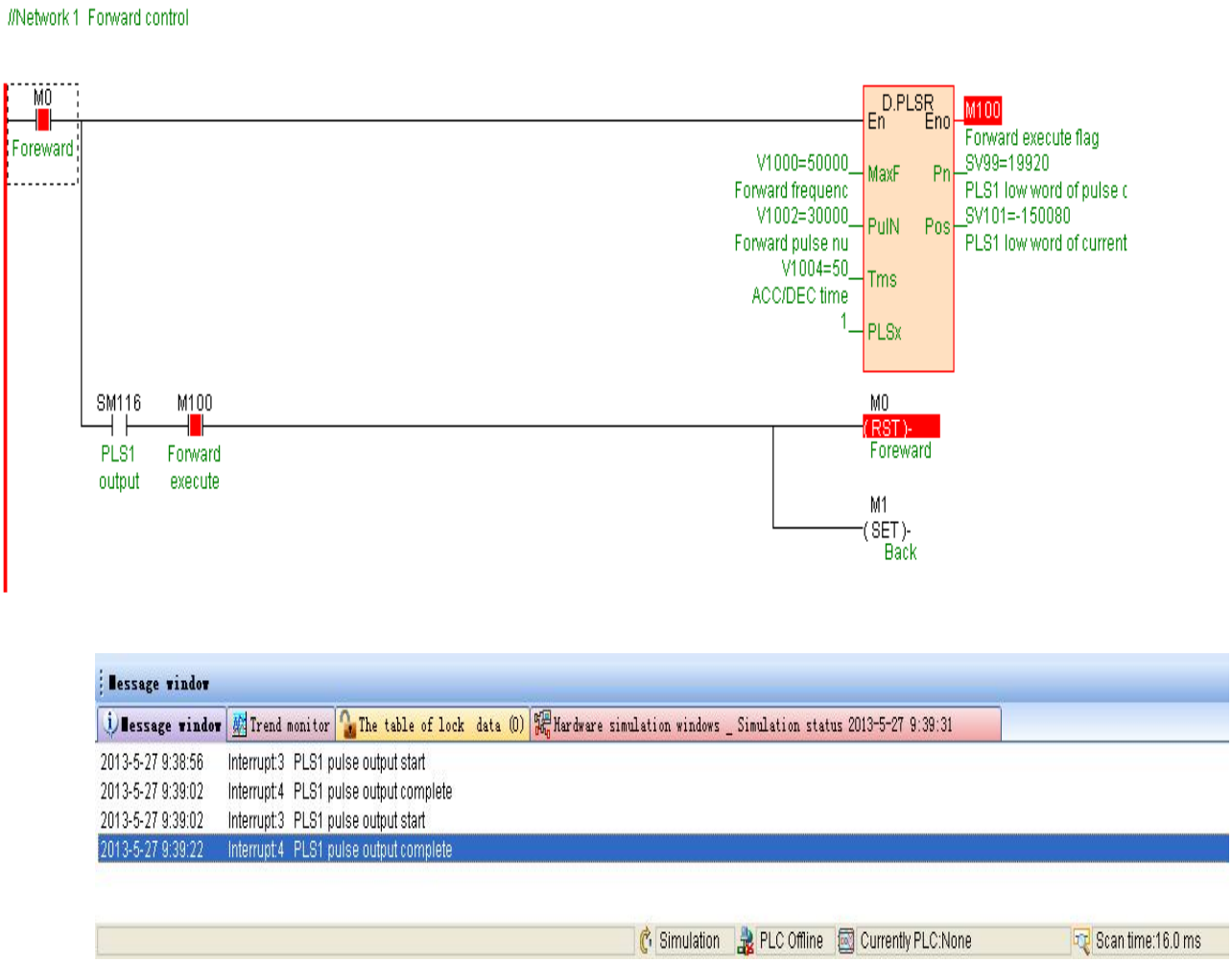
no pulse output Y2=OFF,Y3=OFF.

D. A/B phase pulse output(Y2 is A phase pulse,Y3 is B phase pulse):Y2 . Y3 blink during pulse output ;no pulse output Y2=OFF. Y3=OFF.

E. Synchronization pulse output (Y2 is pulse,Y3 is synchronization pulse): Y2 . Y3 blink during pulse output ;no pulse output Y2=OFF. Y3=OFF.

During high speed pulse output instruction simulate execute process will generate corresponding system interrupt, as follows .

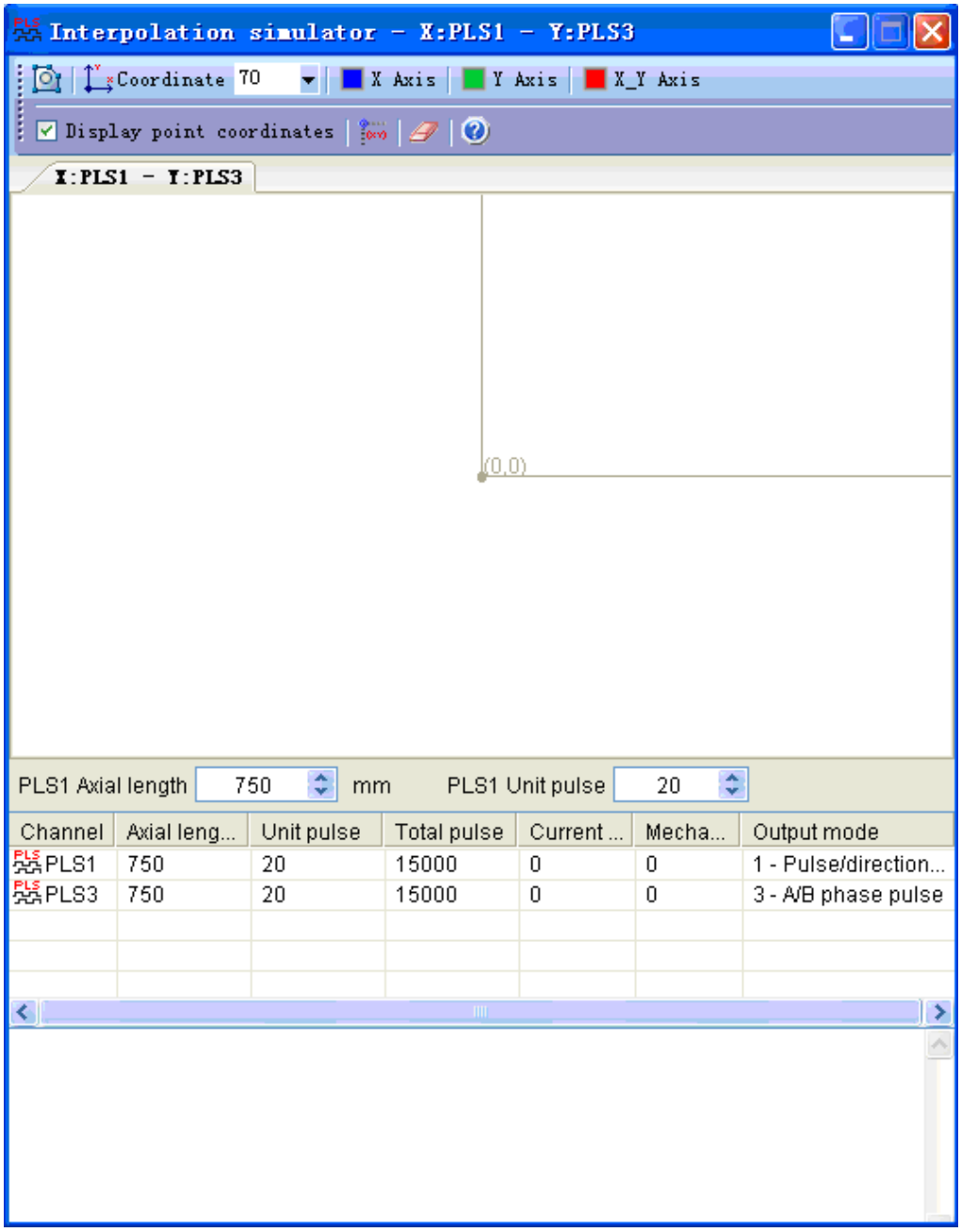
program example: Download



Interpolation simulate

There is interpolation instruction in the program ,under simulate status, may be start " interpolation simulator" to observe interpolation instruction generate motion trail .

2. Click menu [debug/interpolation simulator] or click tool bar button, start " interpolation simulator".



3. " interpolation simulator" tool bar.

A. Click, interpolation instruction executed generate motion trail will be draw. Click stop draw. Note: interpolation instruction executed is

controlled by program, if no instruction executed there is no motion trail be generated.

B. Click 

, may be select different plane coordinates .select consistent coordinate with actual motion platform , more convenience of user observe motion trail.

C. Click 

, may be select the line color of each coordinate axis .

D. Tick 

be display motion tracing point coordinate , no tick then hide point coordinate .

E. Click 
, may be pull the origin of coordinates to middle of the drawing zone .

F. Click 
, may be clear the motion trail .

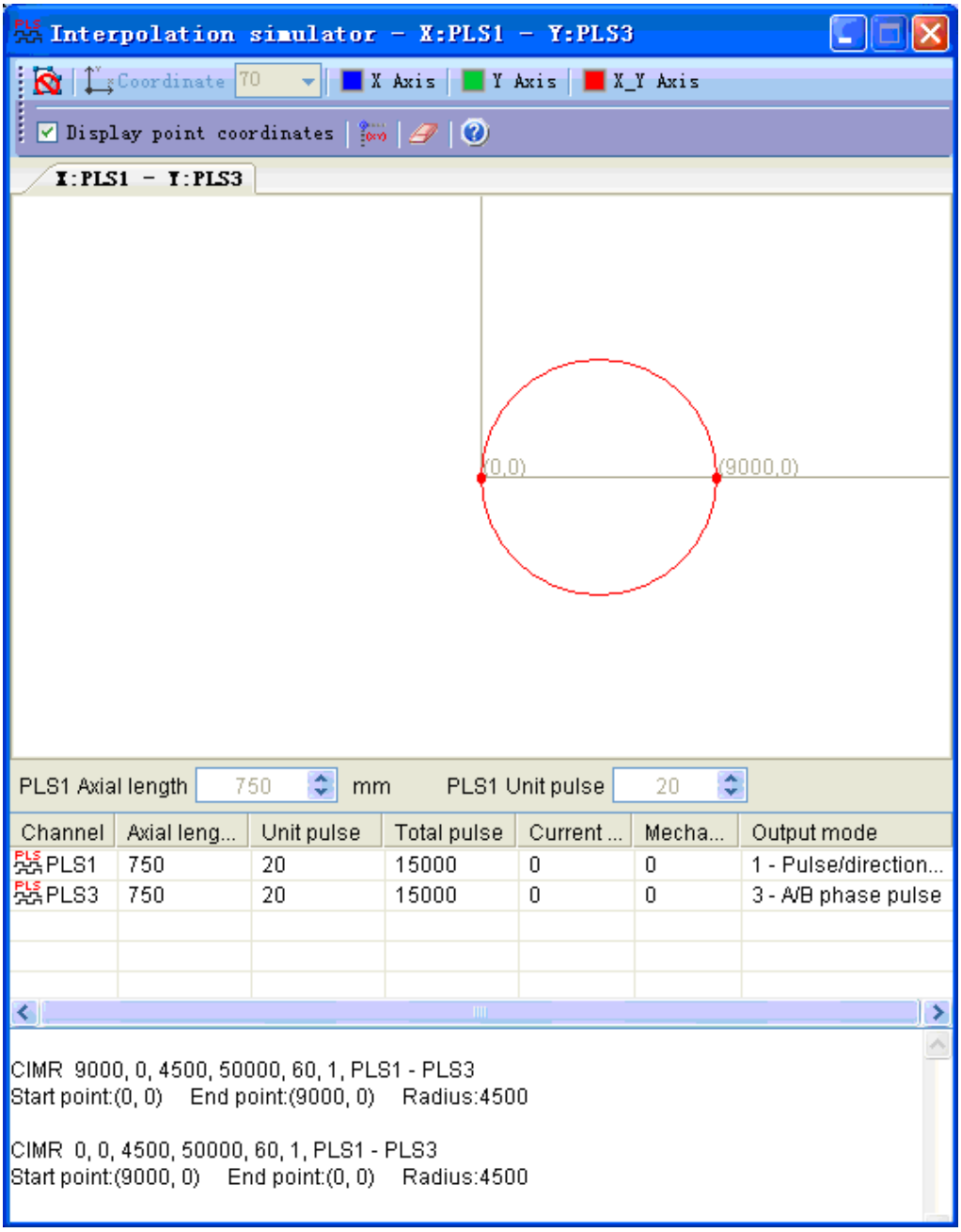
4. Interpolation instruction executed generate motion trail will be draw on plane coordinates, different motion plane will be display paging.

5. At middle part list motion plane each axis corresponding pulse output channel parameter, display the channel current position. mechanical

origin position. output mode etc. , may be set axial length. unit pulse count.

6. Bottom is message box, display executing interpolation instruction and trail describe .

Circular interpolation program example: Download



Difference between online debugging and offline simulate

1. Off line simulate : no need reality PLC, program running in simulator, modify the component status or data value and program execution result

output only happen in simulator .

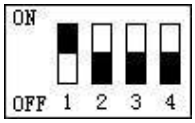
2. On line debug: programming software must on line with PLC, program download to PLC to execute ,modify the component status or data value and program execution result output happen in PLC ,real time response external input singal (DI or AI, output signal (DO or AO produce real control effect to the external device.

Online control PLC

The section introduce use programming software online control the PLC.

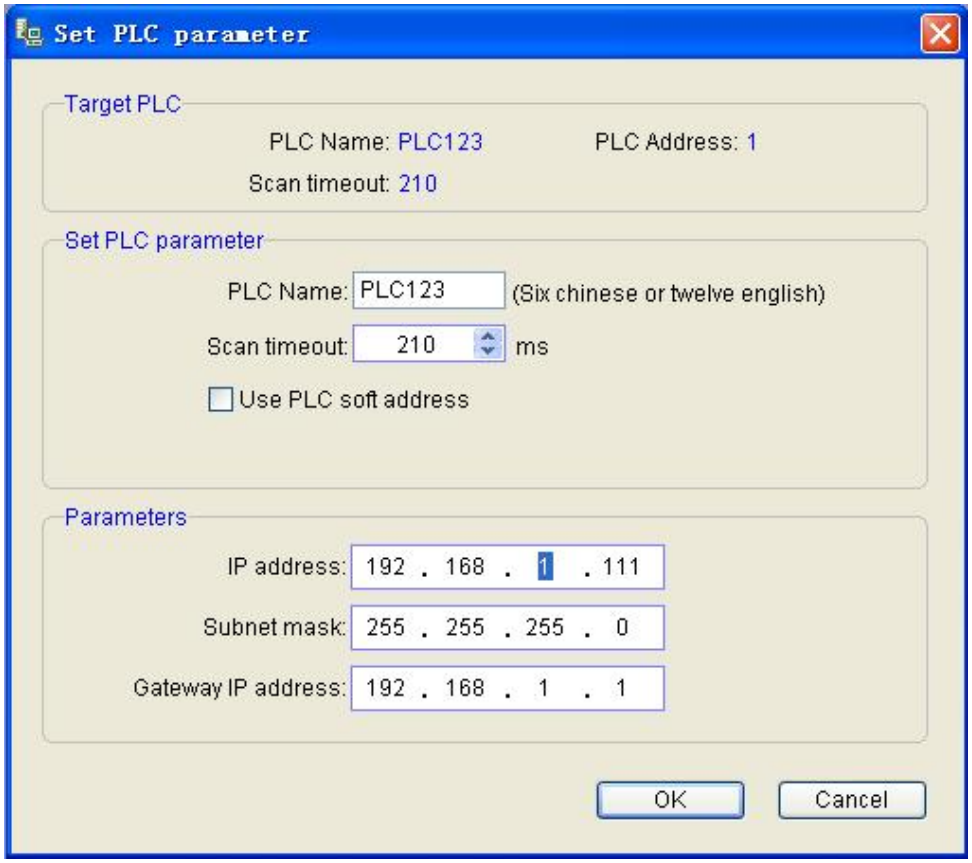
PLC station number setting up

Via hardware 4 bit DIP switch setting up PLC station number(also name station address. communication address). legal station number is 1~254(0 is broadcast address).4 bit DIP may be express decimal number 1~15(binary number from 0001 ~ 1111), method of modify the station number via DIP switch as follow:



Up picture is the 4 bit DIP switch use to set the station number of the PLC, above is ON, below is OFF, in the picture black part express the switch position, the bit set side to ON express the bit is 1, set side to OFF express the bit is 0, first bit in up picture is ON, other bits are OFF. The first bit of the DIP switch express the 0 bit of binary(b0), the fourth bit of the DIP switch express third bit of binary(b3),thus ,such as the DIP switch in up picture express 0001,also decimal data is 1, express the PLC station is 1(default is 1 when leave factory); If set the 1. 2 bits to ON side, moreover others witch set to OFF side ,then be 0011, also decimal data is 3, express PLC station number is 3.

If 4xbit DIP switch express the hardware address not enough to use, may be via menu[PLC/Set PLC parameter] setting up PLC soft address. As follows:

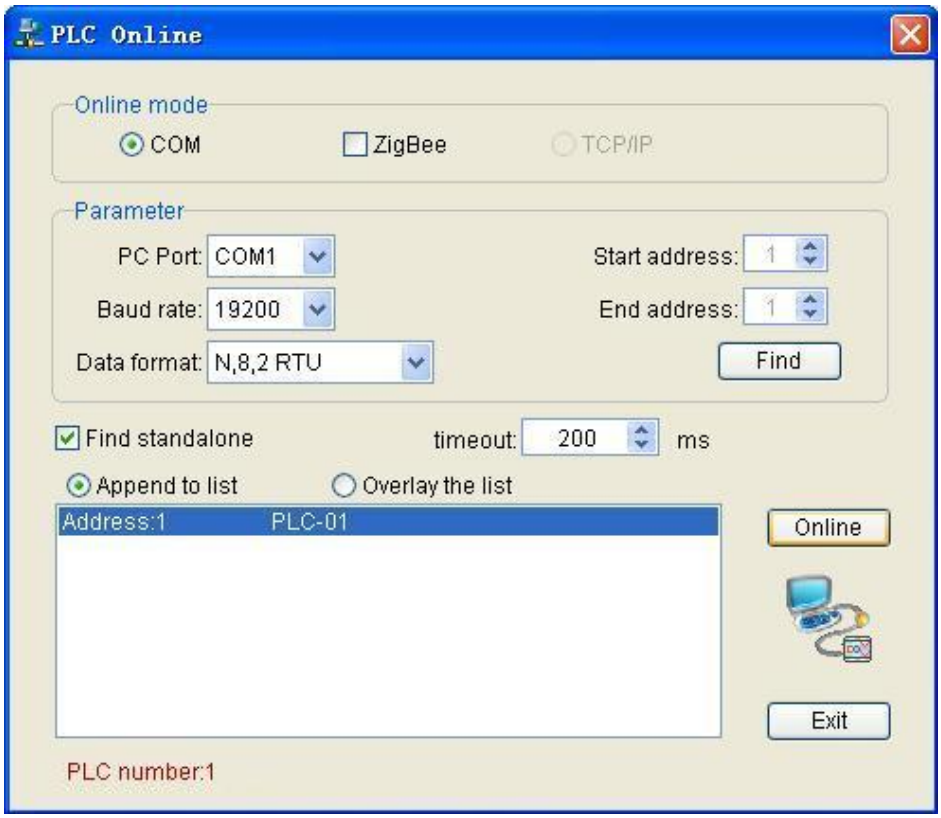


Tick "Use PLC soft address", input PLC address value, click "Confirm" . After use PLC soft address the hardware 4 bit DIP switch will be invalid.

Online with PLC

Only after online to PLC succeed, then may be control the PLC connected in the network , such as upload download program. online monitor etc..

Click menu[PLC/PLC online] or click tools bar button, open "PLC online" window.



Set the relate parameter, general condition use default parameter not need set modify.

- PC port: select the serial port for computer communicate to PLC. Useable COM serial port number in the listing different according to the computer , system will automatic search all useable COM serial port of the computer to user for select by user.

- Baud rate: select communication baud rate, system default is 19200.
- Data format: select communication data format, system default is "N,8,2 RTU".
- Start address. end address: if communicate to single PLC, use "Stand-alone search" option ; if communicate to many PLCs, then "Start address" input the minimum station number of the PLC, "End address" input the maximum station number of the PLC.
- Append to listing. cover listing: select the already online PLCs use append or cover to the listing.
- Overtime : set the overtime of build communicate between computer and PLC. RS232 or RS485 etc. wired mode online default 200ms , wireless mode online (ex. via GPRS) must according to the wireless communicate delay condition then set a biggish value, general suggest around 5000ms.
- Stand-alone search: if communicate to single PLC, use "Stand-alone search" option ; if communicate to many PLCs, then must cancel "Stand-alone search" option moreover set "Start address" and "End address".

Online operation

- Online: direct click "Online" button, searched PLC (online succeed) will be display in the listing.
- Search: if forget about the baud rate etc. communication parameters, may be click "Find" button to search, search function will try all baud rate . all data format to communicate with the PLC, so need spend longer time for waitting for search the PLC, searched PLC (online succeed) will be display in the listing.

Click "Exit", close "PLC online" window.

If online unsuccessful ,chances are below cause, look over to exclusion the problem, if can not exclusion please connect the technical support .

A. Selected PC serial port not correct.

B. Selected communication parameter and the communication parameter of the PLC are different .

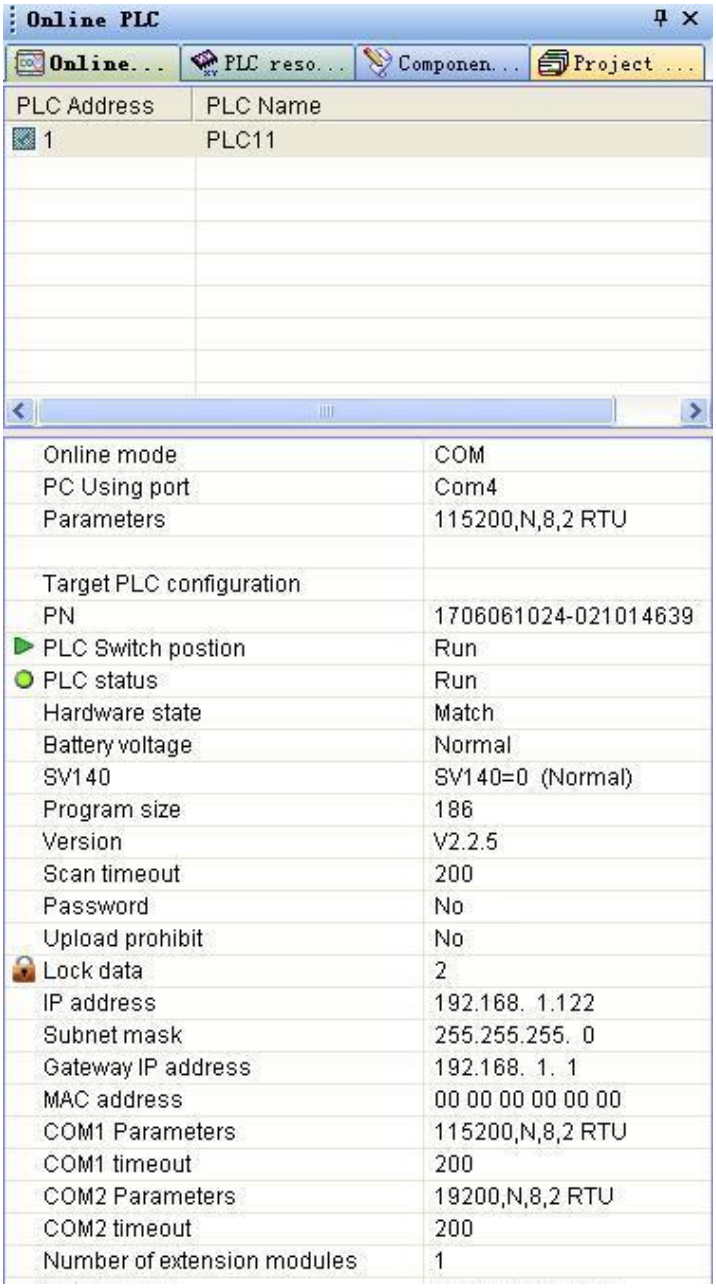
C. PLC power off.

D. Communication cable connected wrong or not connected well.

E. Use "USB convert to RS232 serial port data line" not correct installed the driver.

Online PLC window

At "PLC online" windows if have searched PLC (online succeed), after exit "PLC online" window will automatic open "Online PLC window", as follows.



On the left side of the upper is PLC listing , here display the already connected PLCs. May be double click any PLC in the listing to select be current PLC, all online operation of the PLC as: upload download

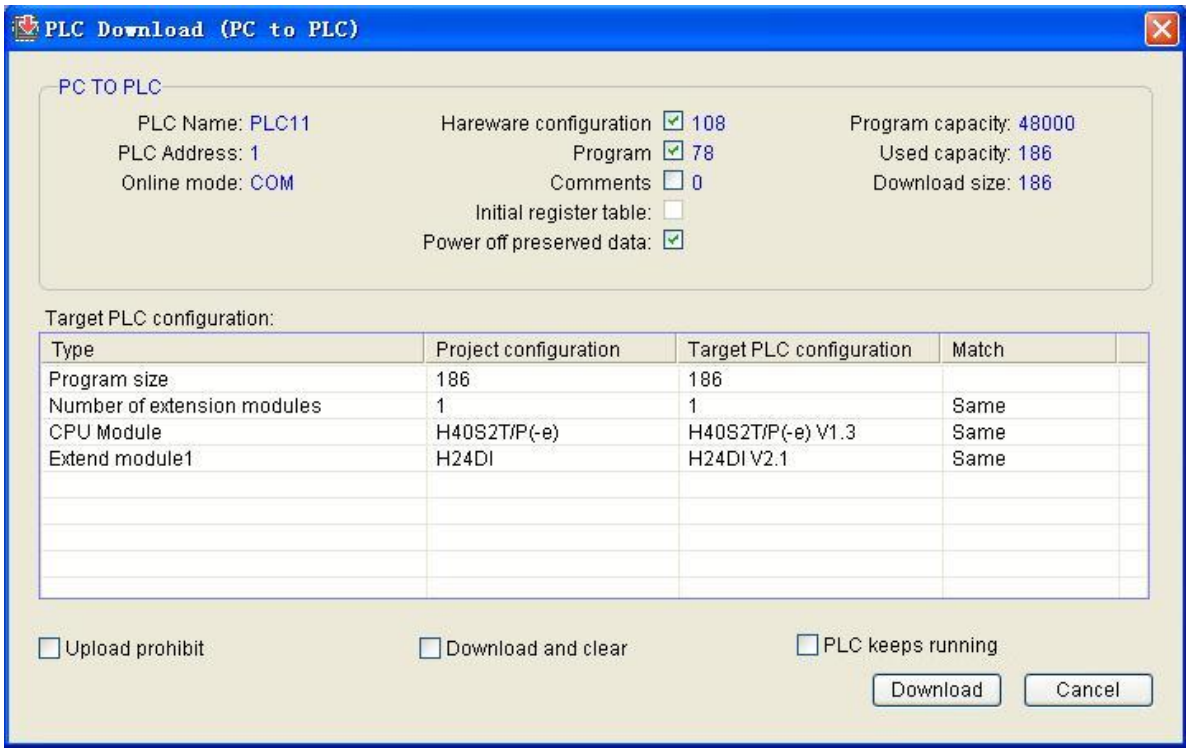
parameter. firmware upgrade etc. only for current PLC, don't influence others PLC.

On the left side of the bottom is monitor zone , online monitoring use for display current PLC detail status ,as: PLC running status. program big or small. version. whether password. series port parameter. master and extend modules type version etc. information.

Download program

Download current program project (hardware configuration. program. comment) to target PLC. Before download system automatic compile the current program project, if the program compiled error, then list all error, after user modify the program until have not any error then can be downloaded . Alarm information only prompt the user pay attention to them , if no error only alarm information , then may be downloaded.

Click menu[PLC/PLC program download] or click tools bar  button, open "PLC program download" window.



Can choose to download content: Hardware configuration, Program, Comments, Initial register table.

Forbid upload: if tick the option, downloaded program can not be uploaded. This way protect the intellectual property of the user program .

With eliminate function download: if tick the option, during download automatic initialize the PLC, include clear program. power of latched area etc., then download the program.

Without stop download: if tick the option, during download program the PLC will not stop executing, thus online modify program. **Please careful**

use the function, did not debugged program maybe produce cannot foresee result.

Hardware match: listing detail list the match status between "Program project configuration" and "Target PLC configuration", hardware configuration information must the same. If hardware configuration not the same, program downloaded to target PLC will produce error code 140, thus "SV3=140 hardware configuration not matched". User need according to the listing prompt modify the different hardware configuration to the same, then again download the program to guarantee PLC proper functioning.

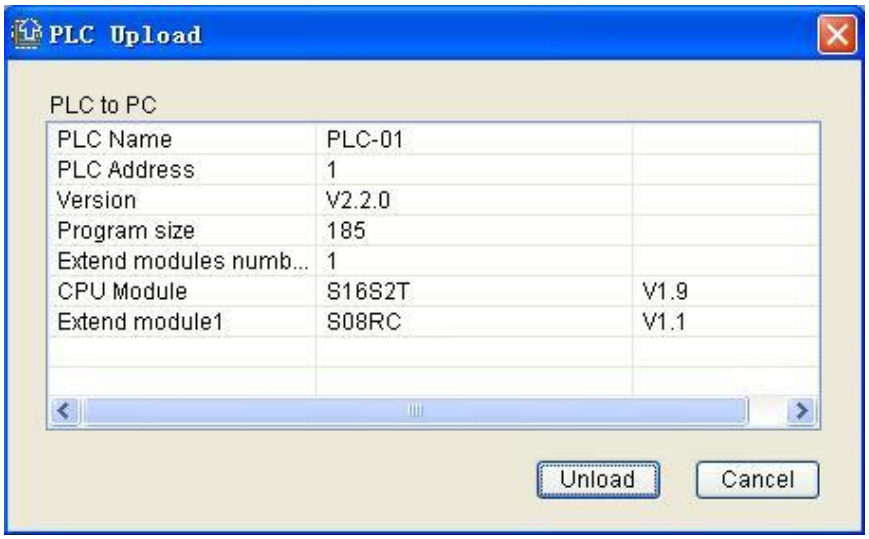
Modify hardware configuration information: if there is "CPU module" different, please via menu [File/Program project attribute] modify CPU module type; if there is "Extend module number or type" different, please via menu [Check/PLC hardware configuration] open "PLC hardware configuration" window and then add subtract or modify the module.

Click "Download" button download the program to PLC.

Upload program

Upload the target PLC program to the computer. If the program selected "Forbid upload" during downloaded, then the program can not be uploaded.

Click menu [PLC/PLC program upload] or click tools bar button, open "PLC program upload " window.

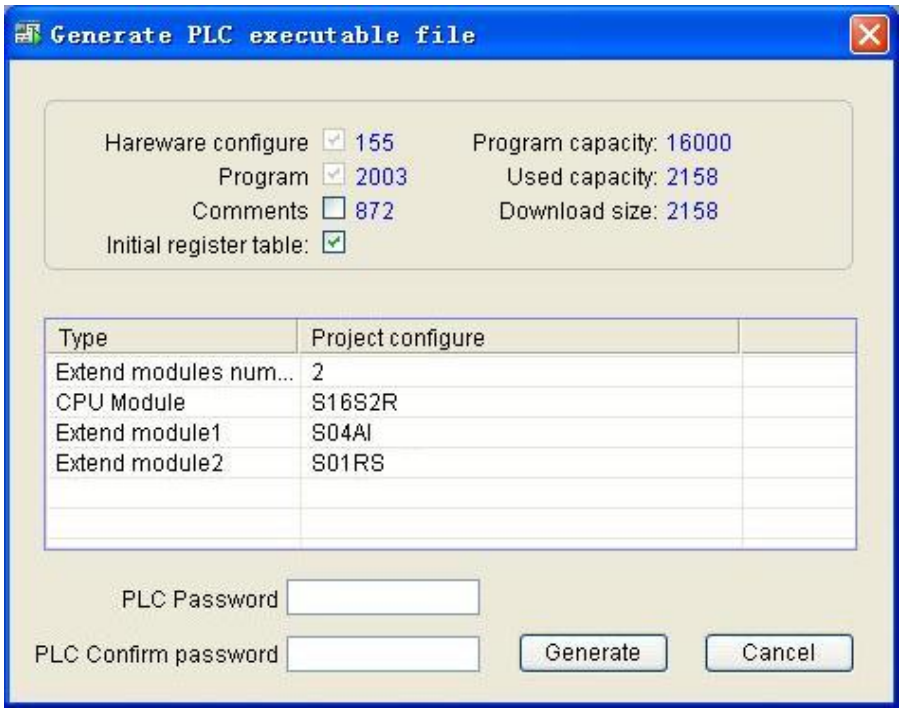


Click "Upload" button, upload the target PLC program to computer.

Generate PLC executable file

The programmable software generate PLC executable file from the PLC project, and the file can be released and downloaded to the PLC lonely,but it cannot be edited.

Clicking menu[File/Generate PLC executable file],open the window that is called "Generate PLC executable file".



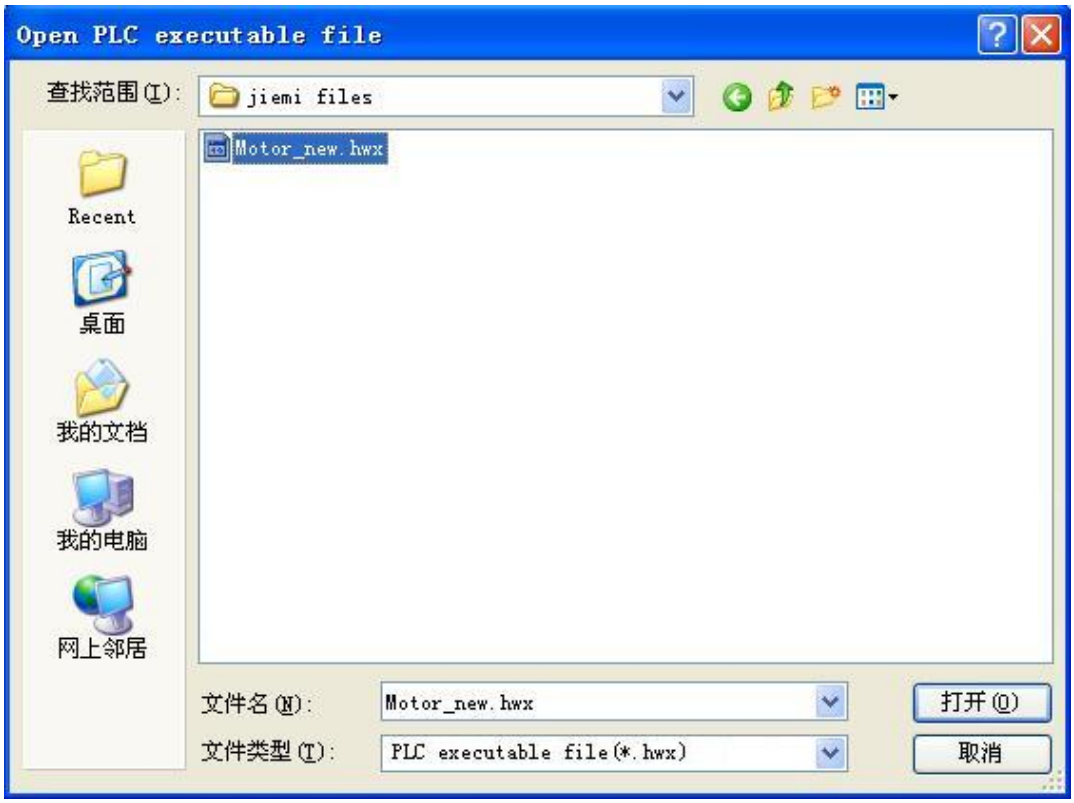
"PLC password". "PLC comfirm password": if PLC is setuped password,it should be input password to download the PLC executable file to the PLC.

Clicking "generate" button,that will generate the executable file.

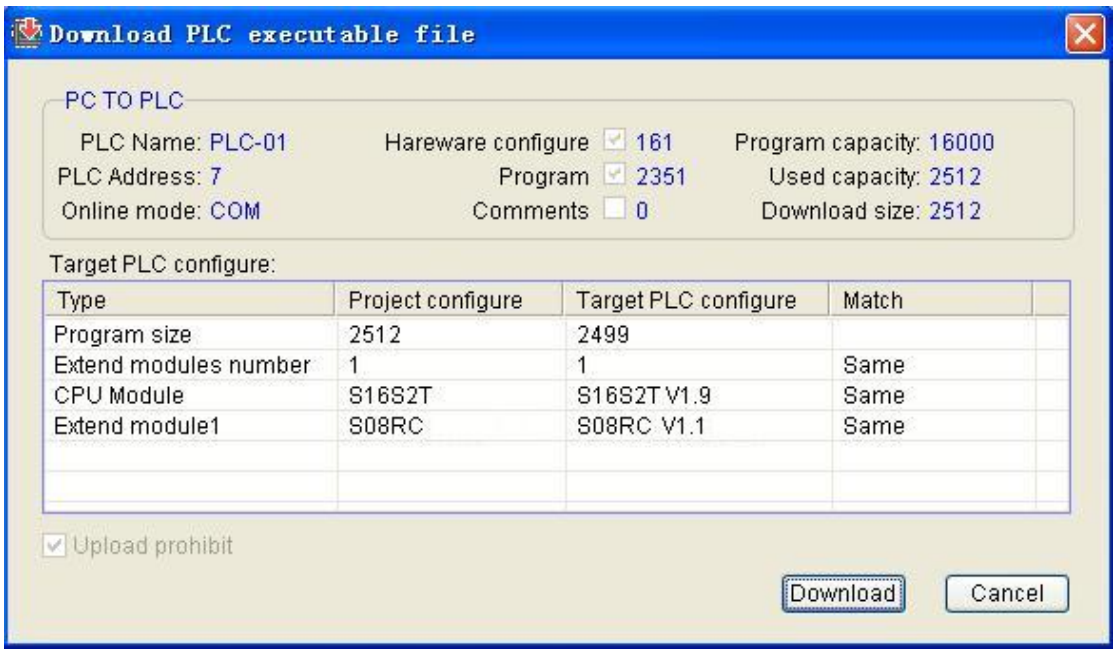
Download PLC executable file

Downloading the executable file to PLC,whose program whill not be uploaded..

Clicking menu[PLC/Download PLC executable file]or clicking button of toolbar,open the window called "Open PLC executable file".



Chosing PLC executable file,opening the window called"Download PLC executable file".

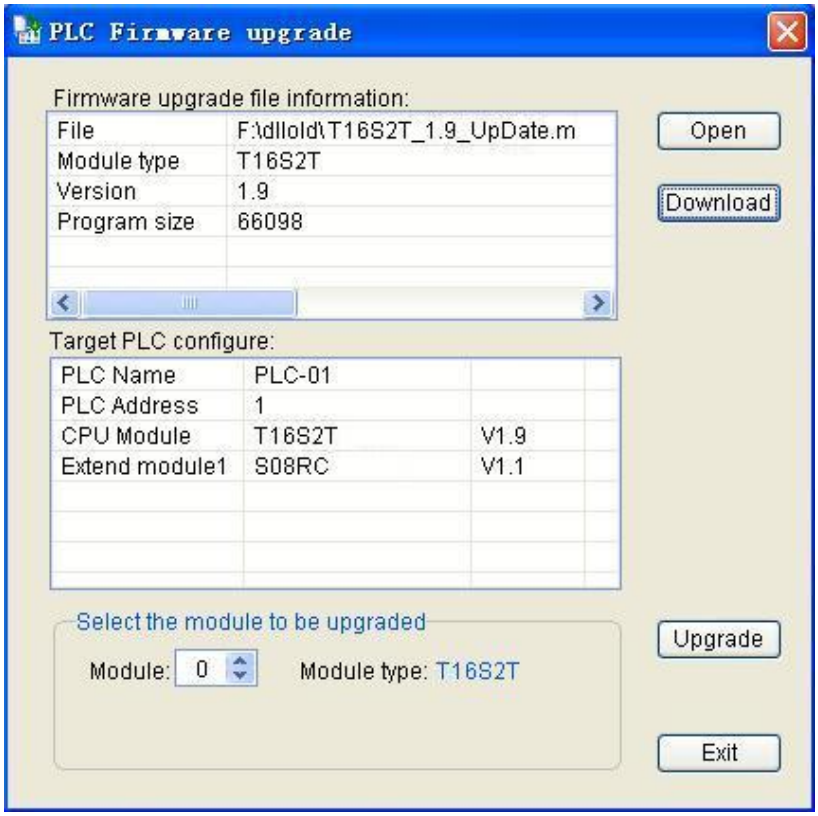


Clicking "Download" button,so that downloaded PLC executable file to PLC.if the hardware is not compatible with target or the password is not correct ,the file will be not downloaded.

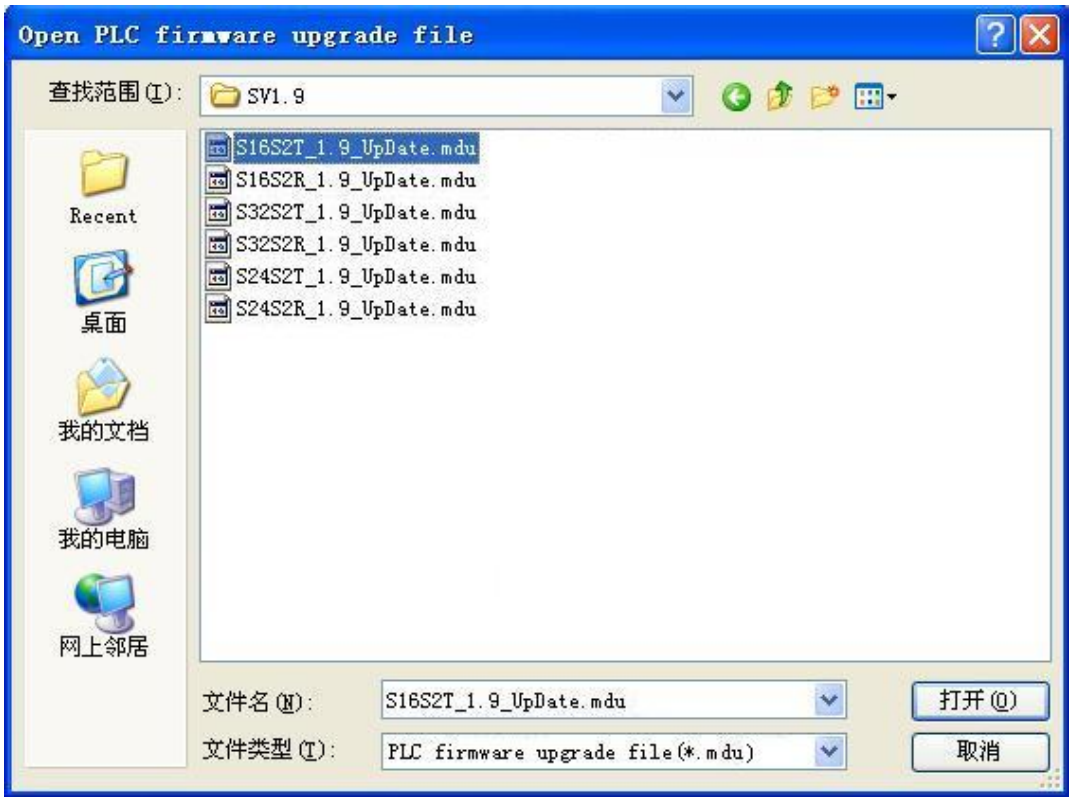
PLC firmware upgrade

Update the firmware program of the PLC CPU MPU or extend module, make CPU MPU oe extend module support the new function.

Click menu [PLC/PLC firmware upgrade], open "PLC firmware upgrade " window



Click "Open" button, select the upgrade file according to the upgraded module.



At "PLC firmware upgrade " window, up table listed firmware upgrade file version. module type etc. information, bottom listed target PLC name. CPU module and extend module etc. hardware configuration information, at "Select upgrade module" box select upgraded module number, module number 0 express CPU module, extend module from 1 begin according to from left to right sequence arrange.

Click "Upgrade" button, upgrade the firmware program of PLC CPU MPU or extend module.

Note: if interrupt during upgrading because power off or other reasons ,must rerun firmware upgrade until upgrade successful.

Start or stop PLC

Via programming software control PLC start or stop, make it RUN/STOP status, realize remote control PLC start or stop.

Click menu [PLC/Start PLC running] start PLC running.

Click menu [PLC/Stop PLC running] stop PLC running.

[Note]

1. At PLC CPU MPU set a start stop switch, when switch at "STOP" position, PLC be stop status; when switch at "RUN" position, PLC be run status.

2. The relation between start stop switch and programming software "Start or stop PLC running" function: according to start stop switch prior, when PLC power off again power on, if start stop switch at STOP position, then now PLC will be stop status; if start stop switch at RUN position, then now PLC will automatic change to run status. only PLC at power on status , programming software maybe proceed "Start PLC

running" or "Stop PLC running" control, no matter start stop switch at any position.

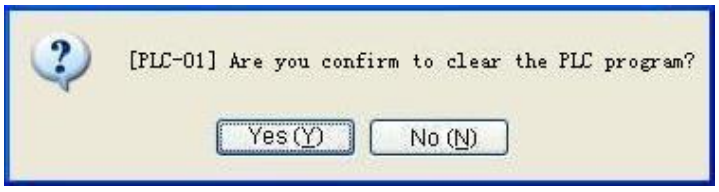
3. If start stop switch at STOP position, after download program or module firmware upgrade complete, PLC will be STOP status; if start stop switch at RUN position, after download program or module firmware upgrade complete, PLC will be RUN status.

4. User control "Start PLC running" or "Stop PLC running" via the programming software, must guarantee locate safety, because of avoid harm for person and machine.

Clear PLC program

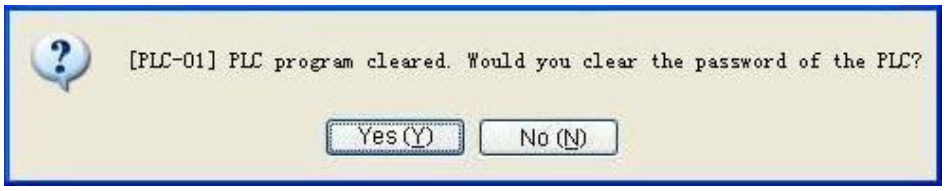
Initialize target PLC, clear include program. hardware configuration. power off latched zone etc..

Click menu [PLC/Clear PLC program]



Affirm whether or not clear PLC program, if select "Yes", will initialize the PLC. If select "No", no any operation to the PLC.

If PLC set password protect, will prompt user input the password, only the password correct, maybe clear the target PLC program. Also prompt user whether or not clear the PLC password after clear the PLC program, if want clear the PLC password, select yes. Then PLC password will be retained.

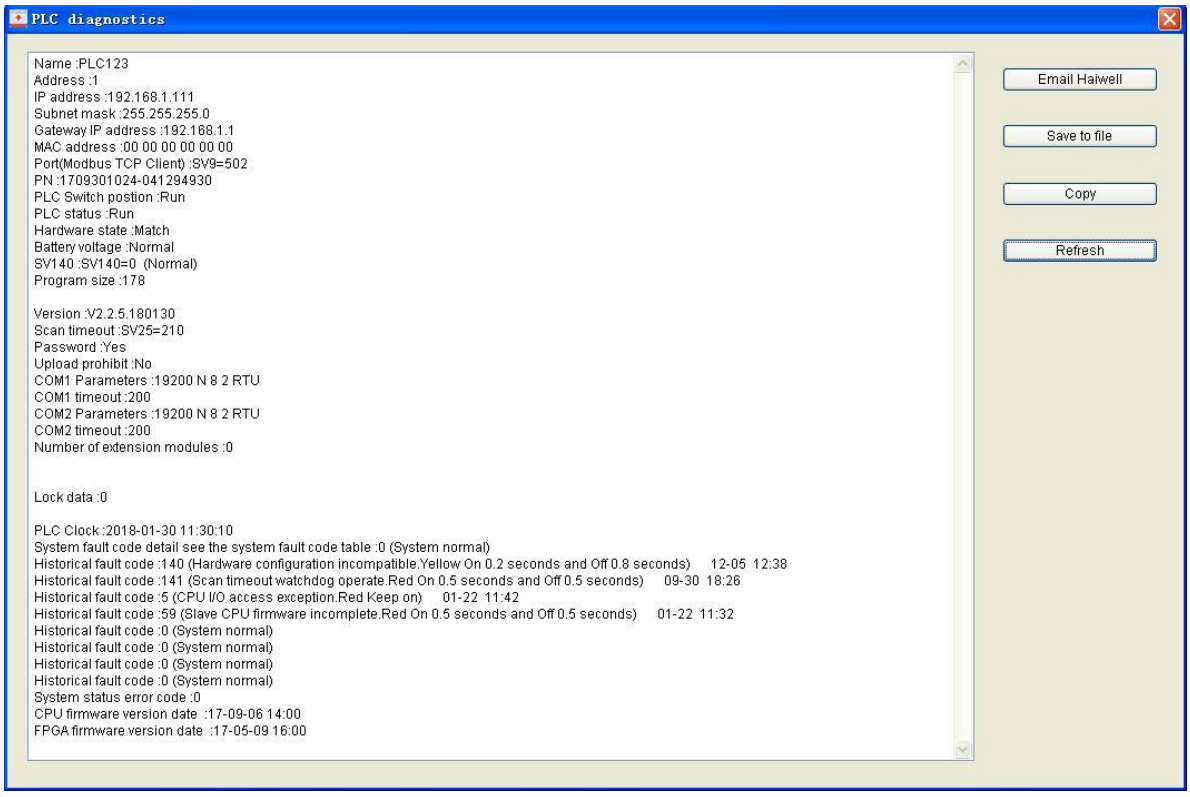


Program compare

Compare between the current program project and the target PLC. distinguish list "Program content" and "Hardware configuration" whether or not the same.

PLC Diagnosis

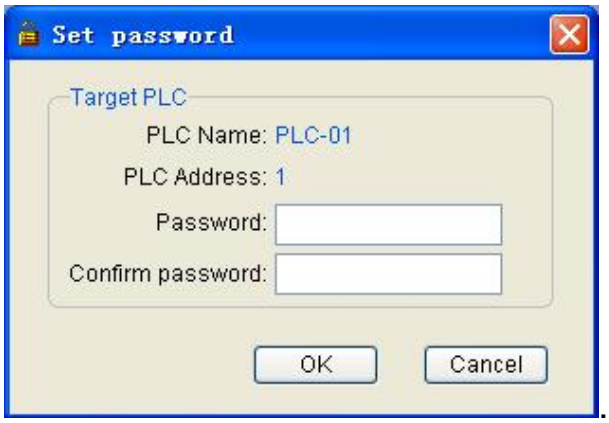
To comprehensive diagnosis PLC of online, PLC listed all of the information system, convenient for the user can quickly find the problem in exceptional cases.



Set PLC password

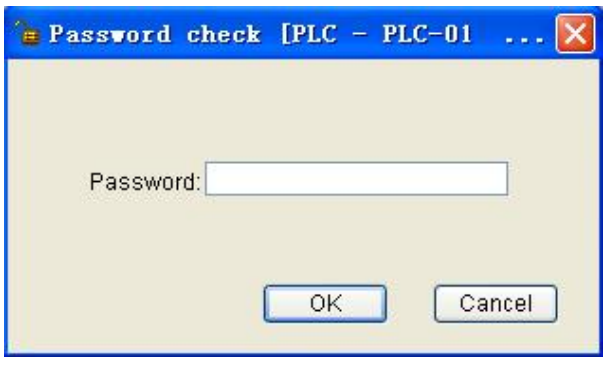
Set target PLC password, use for protect the PLC program and hardware configuration etc. information. After set the password, include program upload . program download. program clear etc. must after input correct password then be executed.

Click menu [PLC/Set PLC password], open "Set PLC password " window.



At password and confirm password field, input the same password , click "Confirm" complete set the password.

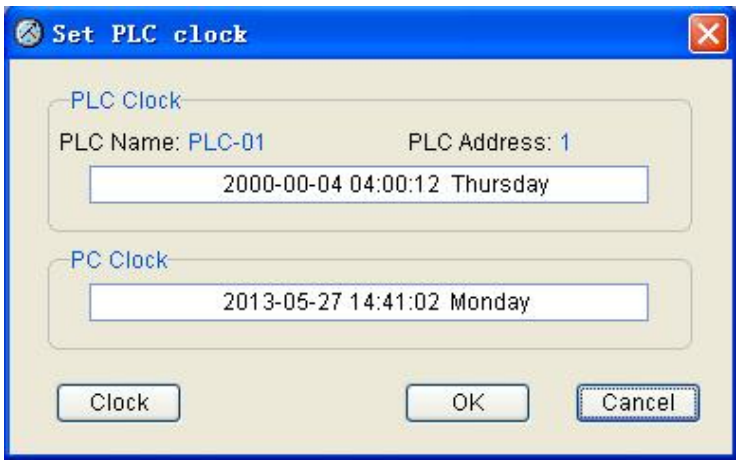
If PLC already set password , user want modify or cancel the password ,click menu [PLC/Set PLC password], open " password checking" window, input "Original password ", after password checked correct will open "Set PLC password " window, again set new password, password is empty express no password be set (thus clear password).



Set PLC clock

Set target PLC real time clock.

Click menu [PLC/Set PLC clock], open "Set PLC clock " window.



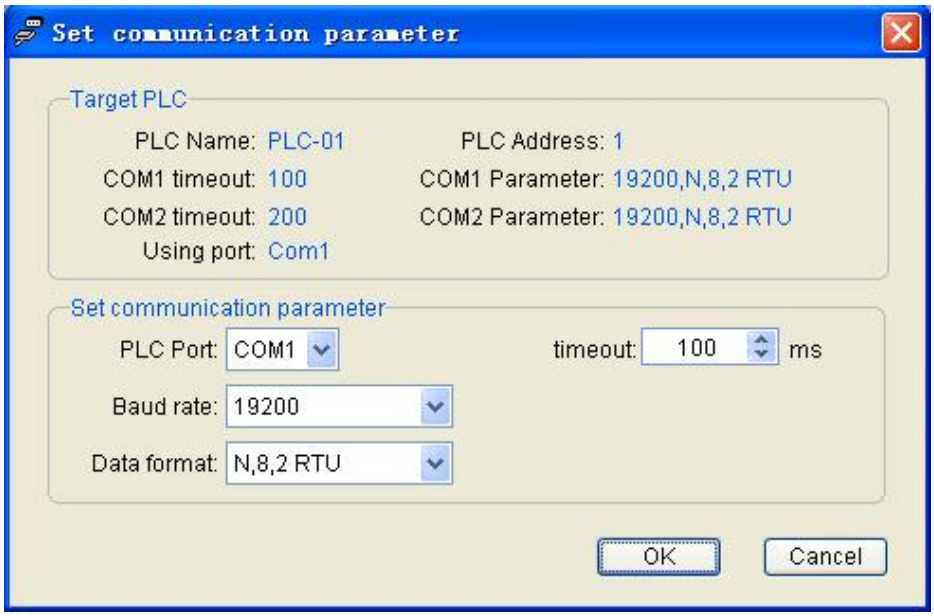
. Click "Confirm" button maybe set the computer current data and time to the PLC real time clock.

If want to regulation computer current clock , click 《 clock 》 button, maybe modify the computer data and time.

Set PLC communication parameter

Set target PLC each communication parameter.

Click menu [PLC/Set PLC communication parameter], open "Set PLC communication parameter" window.



PLC port: select the PLC communication port be set. CPU MPU built-in 2 communication ports(RS232 is COM1,RS485 is COM2). Extend communication port according to reality sequence distinguish are COM3. COM4. COM5.

Baud rate: select communication baud rate.

Data format: select communication data format.

Overtime time: set communication overtime time.

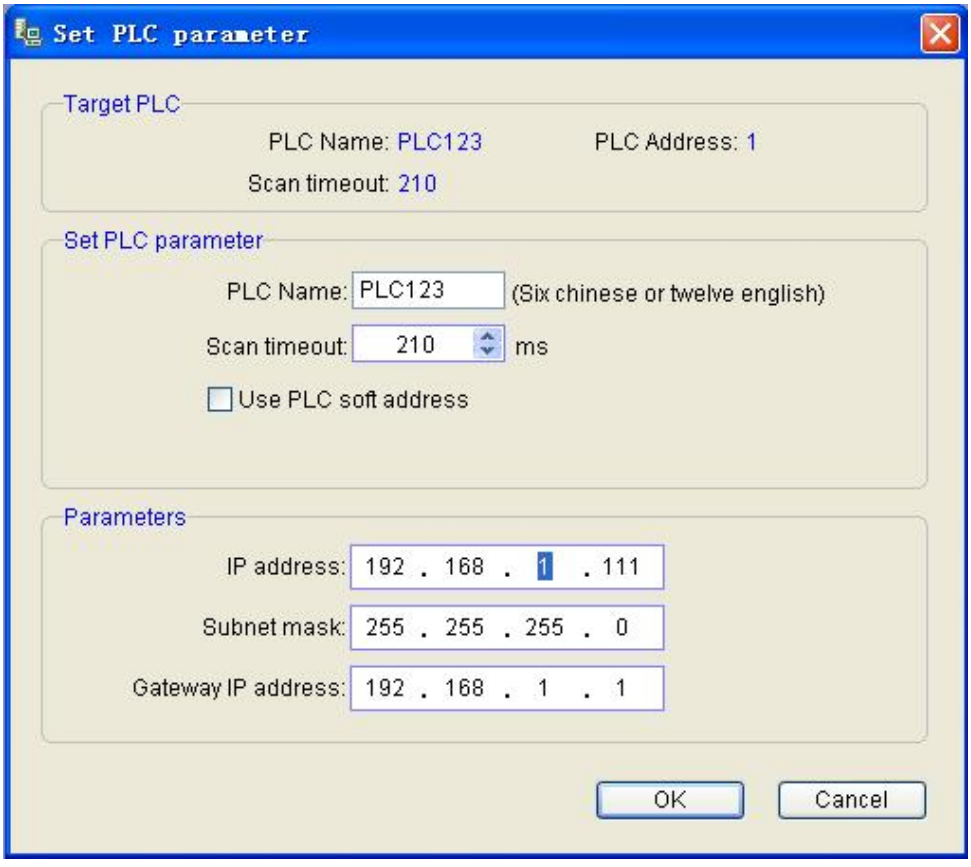
Click " Confirm" button, maybe set PLC communication port communication parameter.

Note: all the communication port default parameter is 19200 N,8,2 RTU, suggest use the default parameter.

Set PLC parameter

Set target PLC name. program scan overtime time and PLC soft address.

Click menu [PLC/Set PLC parameter], open "Set PLC parameter" window.



Set PLC name maybe convenience distinguish many PLCs on the network, PLC name support maximum 6 chinese characters or 12 english characters. numeral.

PLC scan overtime time: thus watch dog time, unit is milliscond (ms). When PLC program running reality scan time greater than the value, will generate error code 141, thus "SV3=141 scan overtime, watch dog action". PLC stop running.

Tick "Use PLC soft address", input PLC address value , click " Confirm". After use PLC soft address the hardware 4 DIP switch will be invalid.

Networking communicate function

This section introduce PLC networking communicate function.

Feature

Support many communication protocol: built-in Modbus RTU/ASCII protocol . freedom communication protocol and Speedbus high speed communication protocol .

Support 5 communication ports: all type MPU built-in 2 communication ports (RS-232 + RS-485), may be extended to 5 communication ports , each communication port may be mutual independence or simultaneously working, the same function of all communication port , all can use for program . upload download program. monitor . networking, all communication port support master or slave communicate mode .

Networking flexible: support 1:N. N:1. N:N networking mode, support all kinds of human-machine interface and configuration software, can be networking with the third part device which has communication function (such as inverter . instrument. bar code scanner etc.).

Highly convenient communication instruction: make you whether use any communication protocol only need one communication instruction can complete complex communication function,programming simpleness, many communication instruction may be executed at the same time, need not be worried about communication port conflict . send receive control. communication interrupt handle etc. problem , may be mix use all kinds of protocol easy complete you need communication function.

Total communication instruction bring OUT output: may be express the communication instruction executed succeed and fail, may be explicit pointing communicate fail with the slave, expedience location debug and fault judge.

May be convenience netwoking with the third part communication instruction:support different baud rate . different protocol format . different manufacturer equipment networking in the the same 485 networking.

Extend module with communication port may be action a remote IO module: PLC extend module built-in a RS485 communication port ,already support parallel bus (use extend bus connect to the parallel

interface of the PLC MPU) also support serial bus (use communication instruction control remote module via RS485 communication port of MPU),When expand via serial bus (remote IO module), don't limit by system expand point ,may distributed installation .This very important for a large number of disperse digital or analog signal(temperature. humidity. differential pressure. blowing rate. flow. fan rotate speed. valve open degree etc.) need be sampled and monitored in the system, easy realize distributed installation moreover no expand point limited , vastly improve control system configuration flexible and in the future expand capacity, reduce wire and work load of all kinds of signal , meanwhile reduce disturb because analog signal wire too long, save project invest cost.

Upper computer (HMI . configuration software etc.)use Modbus protocol access :PLC action slave need no any communication program, each component corresponding Modbus communication address code refer to "communication address code table "

Modbus communication

PLC built-in Modbus RTU/ASCII protocol ,when the third part device (as inverter . servo controller. instrument etc.) support Modbus protocol , may be use Modbus protocol communicate with it , use MODR (Modbus

read) instruction read data from slave and use MODW (Modbus write) instruction write data to slave .only 2 instructions, need no any checking program, it according to communication mode (RTU or ASCII) automatic verify the return data (CRC or LRC).

[E.X.]

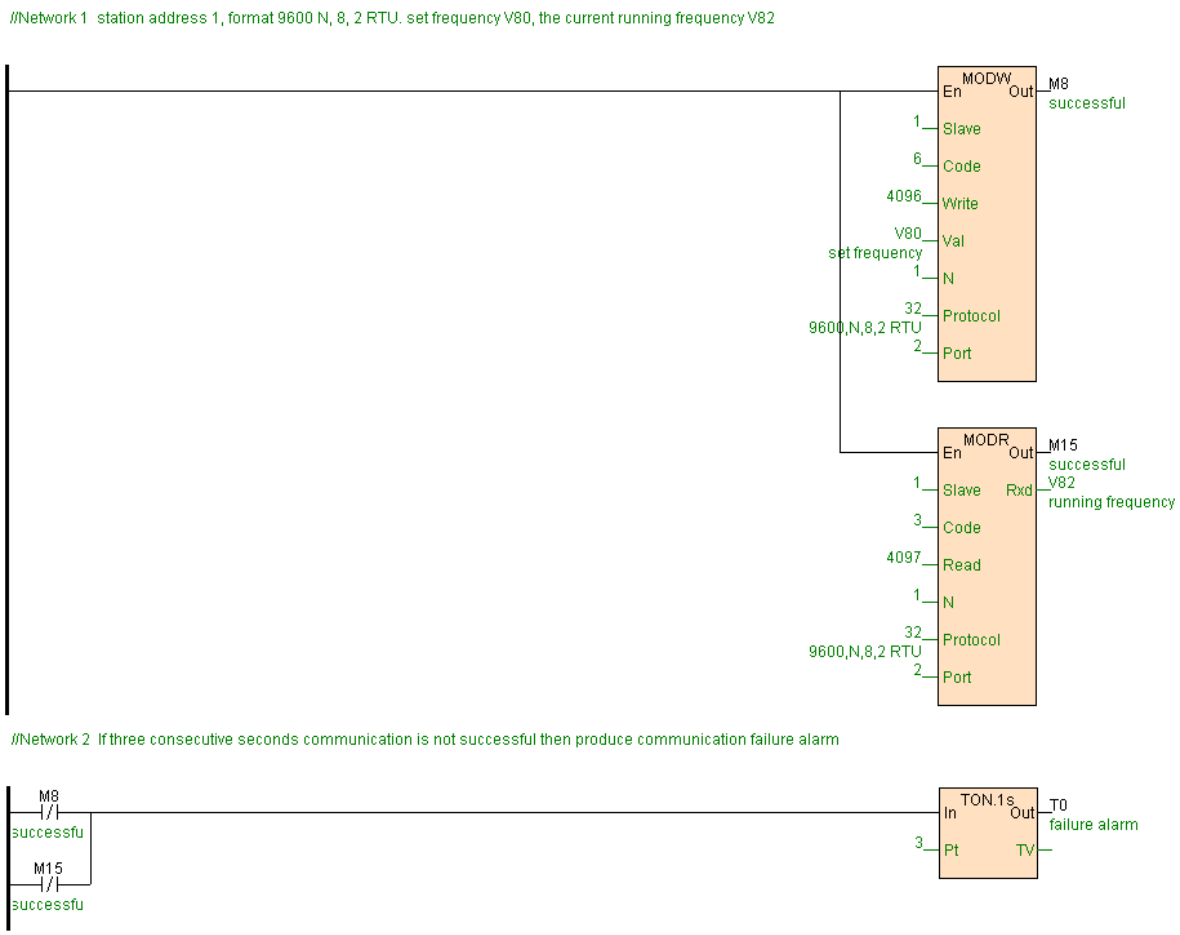
PLC via 485 communication port networking with INOVANCE MD320 series inverter ,use communication mode set frequency to inverter and read running frequency from inverter.

[Example program] Download

1. According to INOVANCE MD320 series inverter communication protocol (please refer to INOVANCE MD320 series inverter manual communication section), preset frequency Modbus address is 4096,MODW instruction real time write V80 value to inverter.

2. Running frequency Modbus address is 4097,MODR instruction read current frequency from inverter and store to V82.

3. MODW. MODR instructions get electricity from busbar and all along executing, if continuous 3 second communication unsuccessful then generate communication fail alarm.



Speedbus communication

Speedbus is a efficient high speed communication protocol , have disperse or continuous . blended data (bit component. register component) transmittability, have high speed communication and communication efficiency, once communication maximum complete 30 pen data interactive, there is 2 Speedbus protocol communication instructions , are the HWRD (Speedbus read instruction ,must define "Speedbus read communication table ") and HWWR (Speedbus write instruction, must define "Speedbus write

communication table").When single PLC control ability not enough or control equipment distance disperse , often need use many PLC proceed substation control, data interaction between each PLC according to the need .

PLC have powerful networking function , data interaction between station may be use Speedbus protocol ,also may be use standard Modbus protocol , whether use any protocol , slave station need no any program , only need read or write instruction at the master PLC.

[E.X.]

2 PLC via 485 communication port networking, use Speedbus protocol to exchange data .

[Example program] Download

1. Only 1#PLC write communication program,2#PLC need no any communication program.if continuous 3 second communication unsuccessful then generate communication fail alarm.

2. Define Speedbus read communication table "read 2#PLC data" as follows:

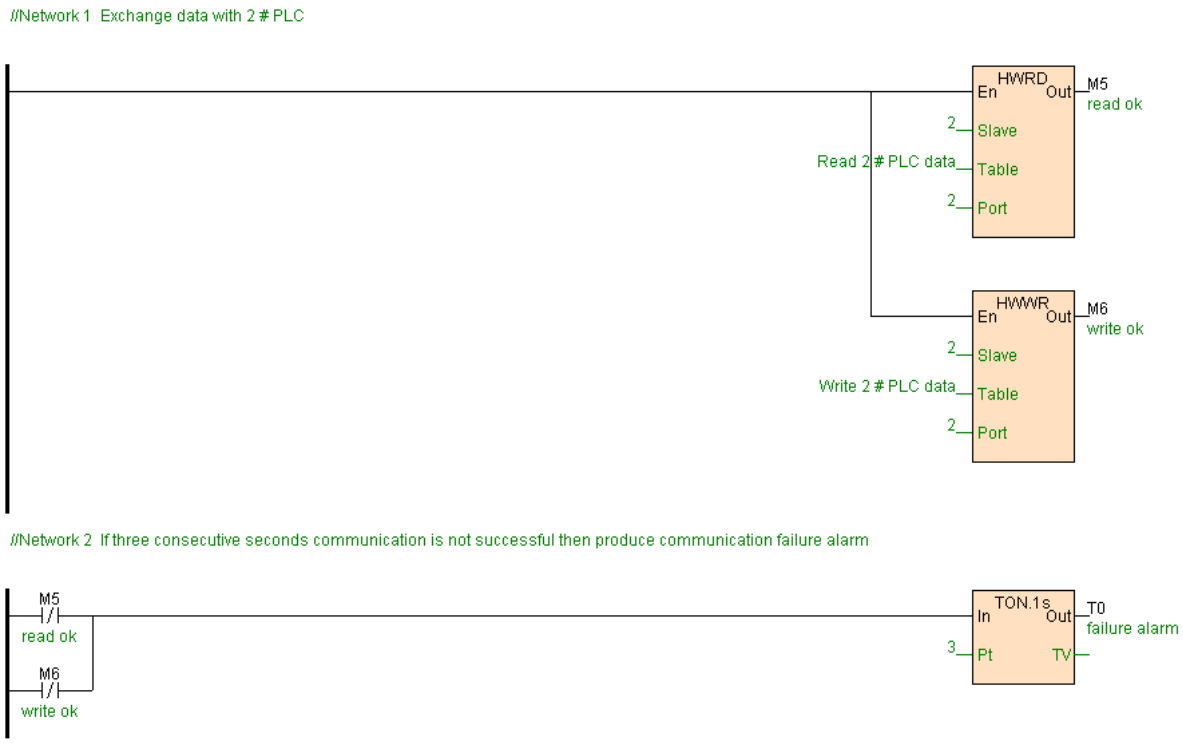
Number	Read data component from slave	D ata component wrote to master
1	X0	M10
2	X3	M11
3	V11	V80

4	V12	V81
5	AI0	V20
6	AI1	V21

3. Define Speedbus write communication table " write 2#PLC data" as follows:

Number	Read data component from master	Data component wrote to slave
1	X0	M100
2	X1	M101
3	V0	V100
4	V50	V102
5	Y4	M0
6	Y5	Y0
7	V60	V200
8	V61	V201

4. HWRD. HWWR instructions get electricity from busbar and all along executing, according to above defined Speedbus read/write communication table, automatic data exchange with 2#PLC.



Freedom communication

PLC except support standard Modbus protocol and Speedbus protocol , also support flexible freedom protocol communication ,whether slave equipment use any protocol ,so long as know it protocol text may be communicate with it.

Freedom communication kernel is read slave communication protocol text, according to slave protocol demand send correct content (byte string) to slave, and receive response data from slave then according to the protocol proceed verify. decompose. account for correct result, specific refer to

COMM. RCV. XMT instruction.

Below 2 reality communication example explain how to use the third part communication protocol realize freedom communication.

[Example 一]

PLC networking communicate with multifunction electric energy meter which accord with DLT-645 standard ,read the meter constant (active power) and current active power total electric energy data value.

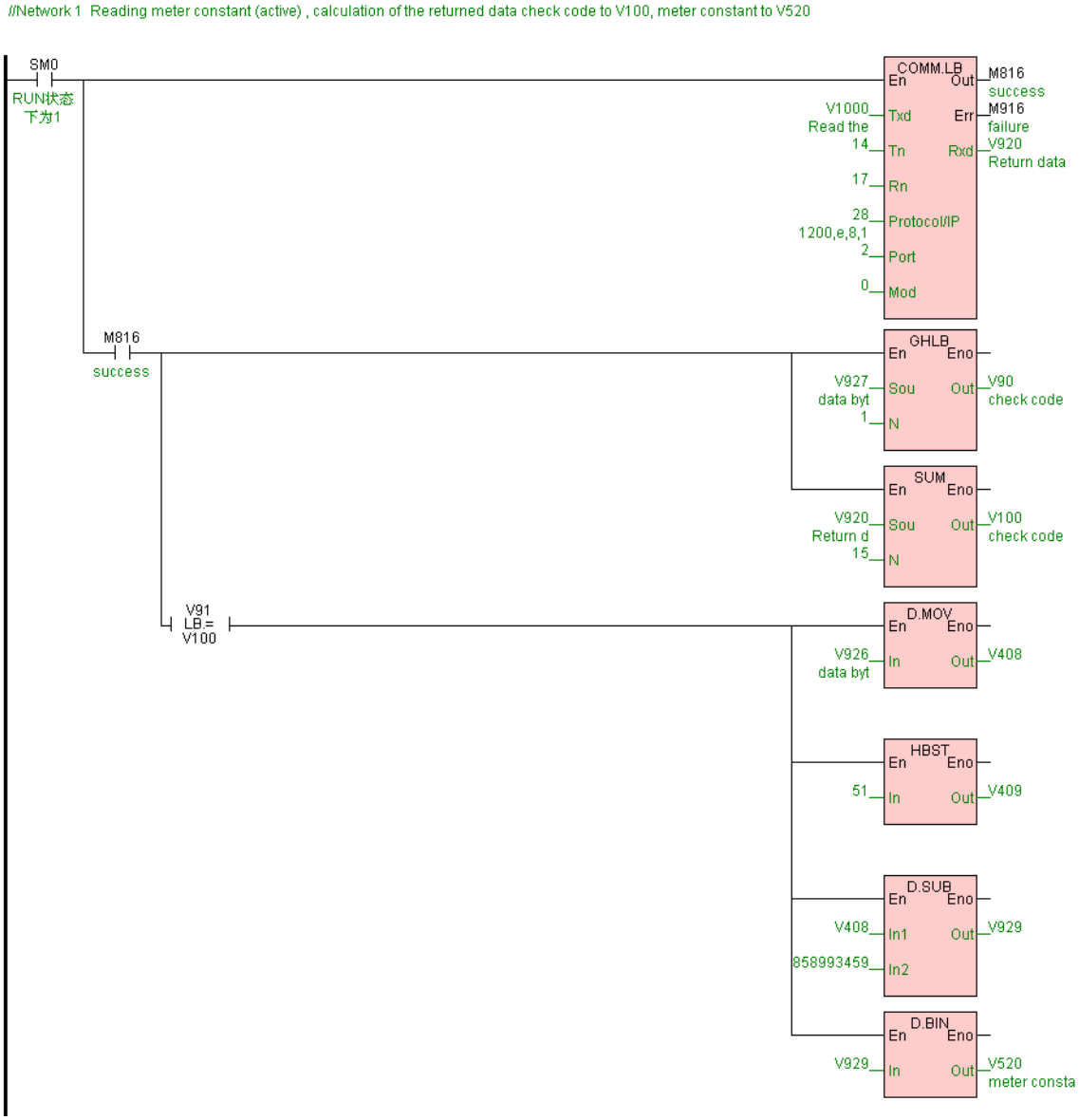
[Example 一 program] Download

1. According to DLT-645 protocol rule, read the meter constant (active power) send command frame:68 07 05 06 00 00 00 68 01 02 63 F3 3B 16 total 14 bytes, meter constant (active power) identification coding :0xC030 + 0x3333 = 0xF363, command store at " read meter constant (active power)" table.

Number	Component default value	Declare
1	V1000=0x0068	Start symbol
2	V1001=0x0007	Address
3	V1002=0x0005	Address
4	V1003=0x0006	Address
5	V1004=0x0000	Address
6	V1005=0x0000	Address
7	V1006=0x0000	Address
8	V1007=0x0068	Start symbol
9	V1008=0x0001	Control code
10	V1009=0x0002	Data lngth
11	V1010=0x0063	Identification code
12	V1011=0x00F3	Identification code
13	V1012=0x003B	Verify code

14	V1013=0x0016	End symbol

Program use COMM.LB instruction(low byte mode) send low byte of V1000~V1013 component receive, 17 bytes return data received store to V920~V928. If 17 bytes data which returned are :68 07 05 06 00 00 00 68 81 05 63 F3 97 33 33 BB 16,then meter constant (active power) V520=64.

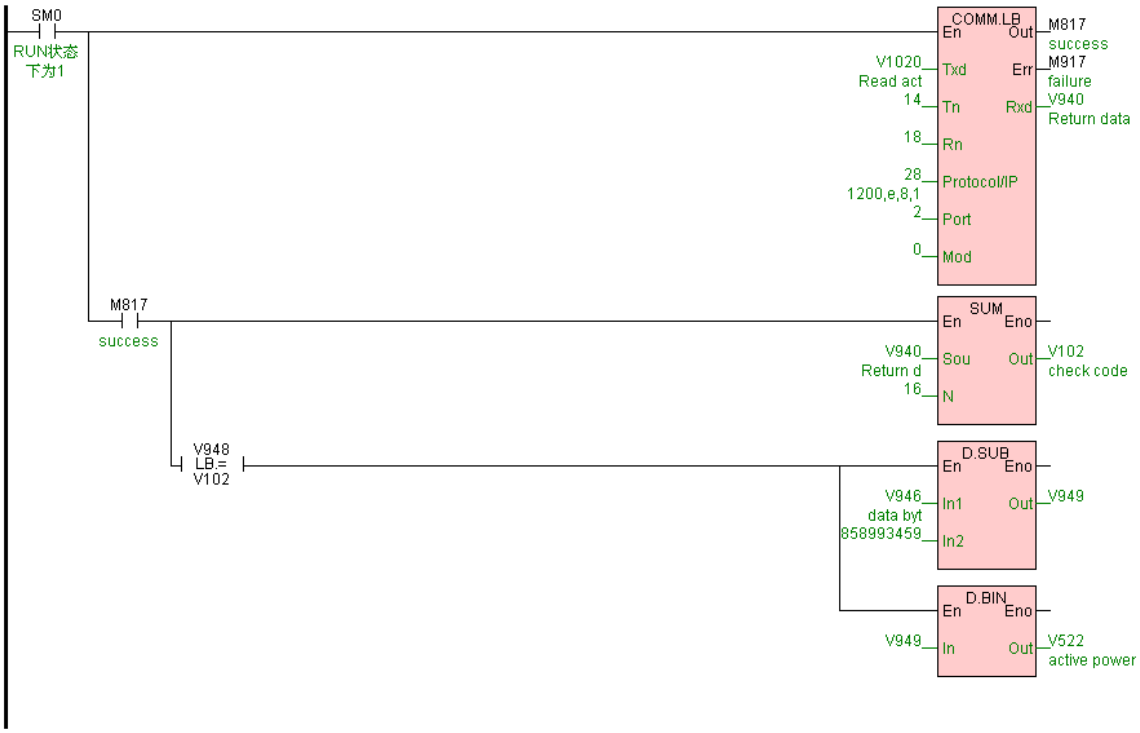


2. According to DLT-645 protocol rule, read current active power total electric energy send command frame:68 07 05 06 00 00 00 68 01 02 43 C3 EB 16 total 14 bytes, current active power total electric energy identification code:0x9010 + 0x3333 = 0xC343,store to " read current active power total meter constant " table.

Number	Component default value	Declare
1	V1020=0x0068	Start symbol
2	V1021=0x0007	Address
3	V1022=0x0005	Address
4	V1023=0x0006	Address
5	V1024=0x0000	Address
6	V1025=0x0000	Address
7	V1026=0x0000	Address
8	V1027=0x0068	Start symbol
9	V1028=0x0001	Control code
10	V1029=0x0002	Data lngth
11	V1030=0x0043	Identification code
12	V1031=0x00C3	Identification code
13	V1032=0x00EB	Verify code
14	V1033=0x0016	End symbol

Program use COMM.LB instruction (low byte mode)send V1020~V1033 component low byte, 18 bytes return data received store toV940~V948. If 18 bytes data which returned are :68 07 05 06 00 00 00 68 81 06 43 C3 97 C6 33 33 32 16,then current active power V522=9364.

//Network 2 Read active power, calculation of the returned data check code to V102, active power to V522



[Example 二]

PLC communicate with METTLER TOLEDO T600 weigh terminal ,use it CB920 communication protocol read real time weight value.

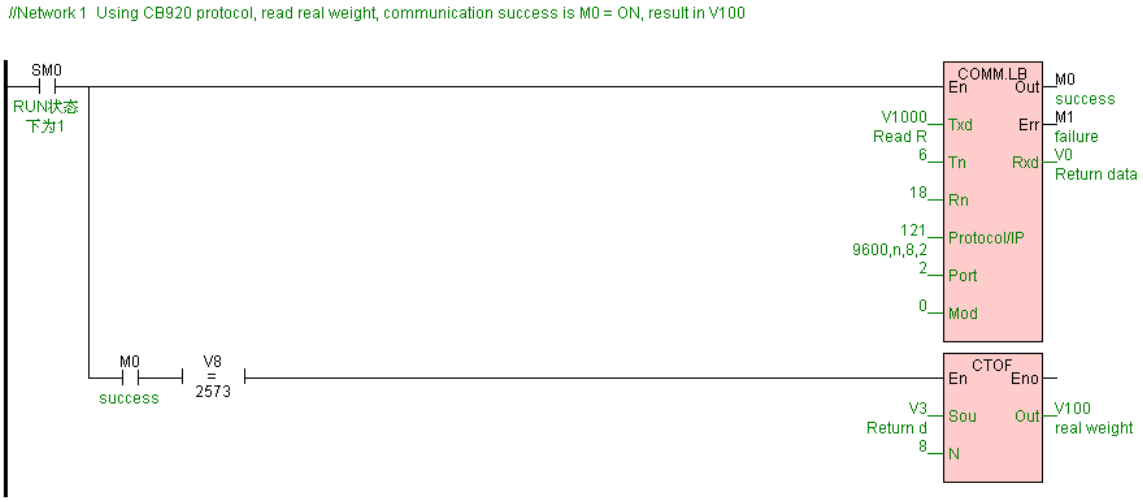
[Example 二 program] Download

1. According to CB920 communication protocol , read real time weight and it ASCII code command:READ/CR/LF, thus 6 bytes command frame:52 45 41 44 0D 0A, command store to " read real time weight " table.

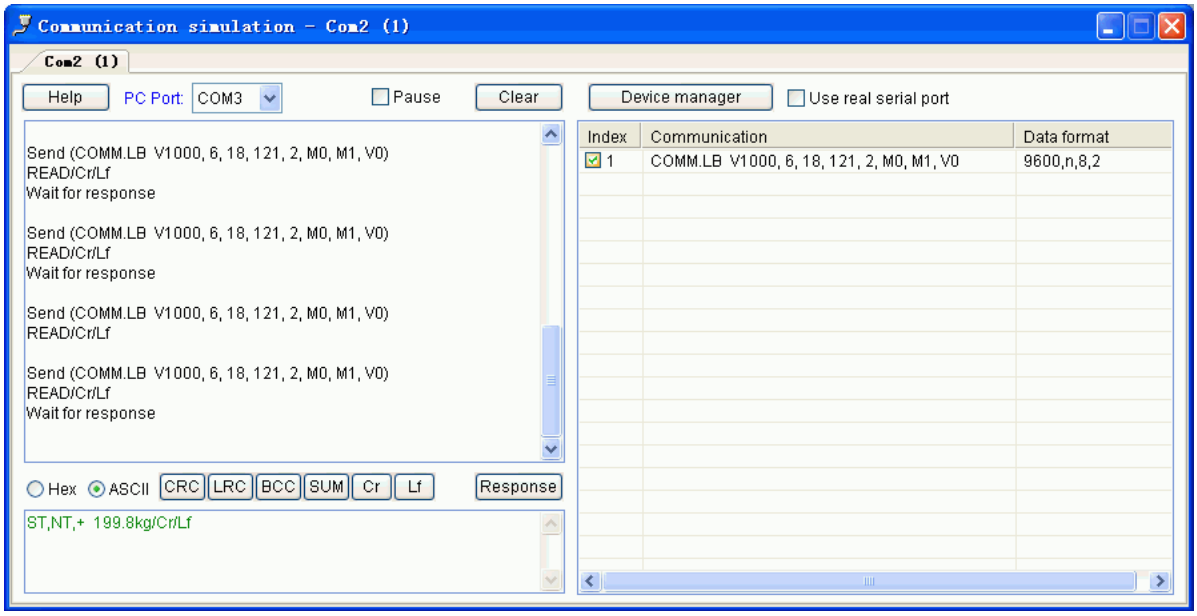
Number	Component default value	ASCII code
1	V1000=0x0052	R
2	V1001=0x0045	E
3	V1002=0x0041	A

4	V1003=0x0044	D
5	V1004=0x000D	/CR
6	V1005=0x000A	/LF

Program use COMM.LB instruction (low byte mode) send low byte of V1000~V1006 component,18 bytes data received store to V0~V8. If 18 bytes data returned as:53 54 2C 4E 54 2C 2B 20 20 31 39 39 2E 38 6B 67 0D 0A(return ASCII code :ST,NT,+199.8kg/Cr/Lf), then real time weight V0=199.8.



2. communication simulator .


Communication simulation - Com2 (1)
Com2 (1)
Help
PC Port: COM3
Pause
Clear
Device manager
Use real serial port
Send (COMM.LB V1000, 6, 18, 121, 2, M0, M1, V0)
READ/Cr/Lf
Wait for response
Send (COMM.LB V1000, 6, 18, 121, 2, M0, M1, V0)
READ/Cr/Lf
Wait for response
Send (COMM.LB V1000, 6, 18, 121, 2, M0, M1, V0)
READ/Cr/Lf
Send (COMM.LB V1000, 6, 18, 121, 2, M0, M1, V0)
READ/Cr/Lf
Wait for response
Index
Communication
Data format
1
COMM.LB V1000, 6, 18, 121, 2, M0, M1, V0
9600,n,8,2
Hex
ASCII
CRC
LRC
BCC
SUM
Cr
Lf
Response
ST,NT,+ 199.8kg/Cr/Lf

Hardware manual

Here are resume of PLC hardware, include type. specification. parameter. install guide. wire drawing. etc.

PLC specification

Item		Specification	Declare
Program control model		Cycle scan model	
Input/output (I/O) control model		Refresh once each cycle scan,support immediately refresh instruction (MPU and extend module)	
Execution speed of instruction		0.05us/basic instruction	
Program language		LD(ladder) + FBD(function block) + IL(instruction list)	Accord with IEC 61131-
Program capacity		48K	
Storage way		Flash ROM permanent storage,dispense with backup battery	
X	External input	X0~X1023 1024 point	Support edge catch and signal filtering set
Y	External output	Y0~Y1023 1024 point	Power-off preserve outp can be configured
M	Auxiliary relay	M0~ M12287 12288 point (default power-off preserve)M1536~M2047 512 point	Power-off preserve area can be set freedom
T	Timer(output coil)	T0~T1023 1024 point (default power-off preserve)T96~T127 32 point	Power-off preserve area can be set freedom,time base:10ms. 100ms. 1s b set freedom,T252~T255 1ms

Item		Specification	Declare
C	Counter(output coil)	C0~C255 256 point (default power-off preserve)C64~C127 64 point	Power-off preserve area can be set freedom
S	Step state bits	S0~S2047 2048 point (default power-off preserve)S156~S255 100 point	Power-off preserve area can be set Freedom
SM	System state bits	SM0~SM215 216 point	
LM	Local relay	LM0~LM31 32 point	
AI	Analog input register	AI0~AI255 256 point	Support quantities convert. sample times a zero point correct
AQ	Analog output register	AQ0~AQ255 256 point	Support quantities convert,power-off preserve output can be configured
V	Internal data register	V0~V14847 14848 point (default power-off preserve)V1000~V2047 1048 point	power-off preserve area can be set freedom
TV	Timer(Current value register)	TV0~TV1023 1024 point (default power-off preserve)TV96~TV127 32 point	Power-off preserve area can be set freedom,time base:10ms. 100ms. 1s can be set freedom,T252~T255 1m
CV	Counter(Current value register)	CV0~CV255 256 point (default power-off preserve)CV64~CV127 64 point	Power-off preserve area can be set freedom,CV48~CV79 ar 32 bits,Other are 16 bits
SV	System register	SV0~SV900 901 point	
LV	Local register	LV0~LV31 32 point	

Item		Specification	Declare
P	Indexed addressing point	P0~P29 30 point,use for indirect addressing	
I	Interrupt	I1-I52 52 point	
LBL	Lable	255point,use for program skip	
Constant	10 Decimal	-32768~+32767(16 bits),-2147483648~+2147483647(32 bits)	
	16 Hexadecimal	0000~FFFF(16 bits),00000000~FFFFFFFF(32 bits)	
Communication port		MPU built-in 2 communication port(RS232/RS485) ,maximum 5 communication port (RS232/RS485) be extended	can be program or networking(master/slave
Communication protocol		Modbus RTU/ASCII protocol. freedom communication protocol. Speedbus speed communication protocol,Baud rate 1200~115200bps	
PLC network capacity		PLC communication address can be set external set,maximum 254,support 1:N. N:1. N:N network	
Real time clock(RTC)		Display:year/month/day/hour/minute/second/week	(H/N series built-in battery)
Hardware extended capacity		7 module can be extended	
High speed counter		8 Channel 200K	Have teaching function,7 counting model:1 - pulse/direction 1 times,2 pulse/direction 2 times,3 positive/reversal pulse 1 times,4 - positive/reversa pulse 2 times,5 - A/B phase pulse 1 times,6 - A/B phase pulse 2 times - A/B phase pulse 4 time

Item	Specification	Declare
High speed pulse output	8 Channel 200K	5 output models:1 - sing pulse output,2 - pulse/direction output,3 · positive/reversal pulse output,4 - A/B phase pulse output,5 - Synchronism pulse outp
Float point arithmetic instruction	support within 32 bits float point arithmetic,integer/float point convert arithmetic	
password protect	Support three level password protect function(program file passord. program block password. PLC hardword password) and upload prohibited function	

Other specification

1. Power specification

Item	AC supply	DC supply
Input power supply	100~240VAC	24VDC -15%~+20%
Power supply frequency	50~60Hz	---
Instant surge	MAX 20A 1.5ms @220VAC	MAX 20A 1.5ms @24VDC
Power output	MAX 25VA	---
Permit Power supply lost	20ms within @220VAC	10ms within
Fuse capacity	2A,250V	2A,250V
Action (working) specification	When input power's volatge rise to 95~100VAC,PLC will be run,when input power volatge drop down to 70VAC,PLC will be stoped.	---

Item		AC supply	DC supply
Output power supply	5VDC for CPU	5V,-2%~+2%,1.2A(maximum)	5V,-2%~+2%,1.2A(maximum)
	24VDC power supply for output and extend modules	24V,-15%~+15%,500 mA(maximum)	24V,-15%~+15%,500mA(maximum)
	24VDC power supply for input and external device	24V,-15%~+15%,200mA(maximum)	Direct use the 24VDC input power supply
Isolation model		Transformer/photoelectricity isolation,1500VAC/1 minute	No electrical isolation
Protect the power supply		24VDC output over the limit of the current	DC power input polar against. over volatge

2. Product environment specification

Item	Environment specification
Temperature/Humidity	Working temperature: 0 ~ + 55 ℃ storage temperature: - 25 ~ + 70 ℃ and humidity: 5 ~ 95% RH, no condensation
Anti vibration	10~57Hz range 0.075mm,57Hz~150Hz acceleration 1G,X. Y. Z three axis 10 times each direction
Anti shock	15G,contiune 11ms,X. Y. Z three axis 6 times each direction

Item	Environment specification
Anti jamming	AC EFT: ± 2500V, surge: ± 2500V, DC EFT: ±2500V, surge: ±1000V
Over volatge capability	Between AC terminal and PE terminal 1500VAC,1min,Between DC terminal and PE terminal 500VAC,1min
Insulation impedance	Between AC terminal and PE terminal@500VDC,>=5MΩ(Between all input/output terminal and PE terminal@500VDC)
Earth	The third grounding(Can not connect to the strong power system's earth)
Operation environment	Operated where no dust, moisture, corrosion, electrical shock and physical shock ,etc.

3. Digital input (DI)specification

Item	Digital input DI
Input signal	Non-voltage contact or NPN/PNP contact
Action driving	ON: 3.5 mA above OFF: below 1.5 mA
Input impedance	About 4.3KΩ
Input maximum current	10mA
Response time	Default 6.4ms,Configurable 0.8~51.2ms
Isolation mode	Each Channel optical isolation
Input indication	LED light means ON,dark means OFF

Item	Digital input DI
Power supply	PLC internal power supply:DC power(sink or source)5.3mA@24VDC

4. Digital output (DO)specification

Item		Relay output-R	NPN or PNP transistor output -T/P
maximum load	Resistance load	2A/1 point,8A/4 point per COM	0.5A/1 point,2A/4 point per COM
	Inductive load	50VA	5W/24VDC
	Light load	100W	12W/24VDC
Min. load		10mA	2mA
Voltage specification		Below 250VAC,30VDC	30VDC
Drive capability		Maximum 5A/250VAC	MAX 1A 10S
Response time		Off-on 10ms,On-off 5ms	Off-On 10us,On-Off 120us
Leakage current when route opened		---	Below 0.1mA
Isolation mode		Mechanical isolation	Each Channel optical isolation
Output indication		LED light means ON ,dark means OFF	
Power supply		PLC internal power supply 24VDC	

5. Analog input (AI)specification

Item	Voltage input	Current input	RTD input	Thermocou Input

Item	Voltage input				Current input		RTD input	Thermocou Input
Input range	-10V~+10V	0V~+10V	0V~+5V	1V~+5V	0~20mA	4~20mA	Pt100. Pt1000. Cu50. Cu100	S. K. T. E. J N. R. Wre3 Wre5/26. 20]mV. [(50]mV. [(100]mV
Resolution	5mV	2.5mV	1.25mV	1.25mV	5uA	5uA	0.1℃	0.1℃
Input impedance	6MΩ				250Ω		6MΩ	6MΩ
Maximum input range	±13V				±30mA			±5V
Input indication	LED light means normal ,dark means break OFF							
Response time	5ms/4 Channel						560ms/4 Channel ,880ms/8 Channel	
Digital input range	12 bits,Code range:0~32000(H series module 16 bits A/D convert)						16 bits,Code range:0~32000	
Precision	0.2% F.S						0.1% F.S	
Power supply	MPU use internal power supply, extend module use external power supply 24VDC ±10 5VA							
Isolation mode	Opto-electric isolation,Non-isolation between Channel ,between analog and digital is o electricisolation							
Power consumption	24VDC ±20%,100mA(maximum)						24VDC ±20%,50mA(maximu	

6. Analog output(AO)specification

Item	Voltage output	Current output

Item	Voltage output				Current output	
Output range	-10V~+10V	0V~ +10V	0V~+5V	1V~+5V	0~20mA	4~20mA
Resolution	5mV	2.5mV	1.25mV	1.25mV	5uA	5uA
Output load impedance	1KΩ@10V		≥500Ω@ 5V		≤500Ω	
Output indication	LED light means normal					
Drive capabilit	10mA					
Response time	3ms					
Digital output range	12 bits,Code range:0~32000(H series module 16 bits D/A convert)					
Precision	0.2% F.S					
Power supply	MPU use internal power supply, extend module use external power supply 24VDC ±10% 5VA					
Isolation mode	Opto-electric isolation,Non-isolation between Channel ,between analog and digital is opto-electricisolation					
Power consumption	24VDC ±20%,100mA(maximum)					

Indicator declare

1. CPU indicator declare

1. POW:power indicator .green,constant light - power normal;Not light - Power abnormal.

2. RUN:Running indicator .green,constant light - PLC is running;Not light - PLC is stopping.

3. COM:communication indicator .green,flicker - communicating,flicker frequency signify the speed of the communication;Not loght - No communication.

4. ERR:Erroe indicator .double(red. yellow),as follows:

Consult manage	Signify information type	ERR the state of the indicator
Normal	Without error	Not light
Normal,just prompt take attention to the locked data	PLC have the component which locked	Yellow flicker:On 0.2 seconds and Off 0.8 seconds
Modificate the PLC hardward configure	Problem in the soft seting ,permit user keep on operate the user program	Yellow flicker:On 0.2 seconds and Off 0.8 seconds
Check module parallel bus (check the RTC battery; check extension module power supply)	Communication abnormal between modul,auto dislodge the abnormal module,permit user keep on operate the user program	Yellow flicker:On 0.8 seconds and Off 0.2 seconds
Upgrade the fireware or modify the user program	Fireware abnormal or user program abnormal,can not operate the user program	Red slowly flicker:indicator light 0.5s not light 0.5s
Maintain	Hardware error,user program con not running	yellow constant light

Note:the specific error code please check the system register SV3,error code corresponding the content see detail the " system error code table".

2. Extend module ndicator declare

1. POW:power indicator .green,constant light -Power normal;Not light - Power error.

2. LINK:many state indicator .three colors(Red. Yellow. Green),as follow:

Consult manage	Module bus state	LINK the state of the indicator
Normal	Module no communication	Not light
	MPU identification the module but have not communication	Green constant light
	Serial or parallel communicating	Green flicker:indicator light 30ms not light 30ms
parallel power supply not enough,must connect to external power supply	Without serial or parallel communicate	Yellow flicker:indicator light 0.5s not light0.5s
	With serial or parallel communicate	Yellow dark and shake alternately:indicator not light 0.5s shark 0.5s
Upgrade the fireware fail,reupgrade the fireware of the module	Without serial or parallel communicate	Red flicker:indicator light 0.5s not light 0.5s
	With serial or parallel communicate	Red dark and shake alternately:indicator not dark 0.5s shark 0.5s
Maintain	Without serial or parallel communicate	Red constant light
	With serial or parallel communicate	Red shark quickly:indicator light 30ms not light 30ms

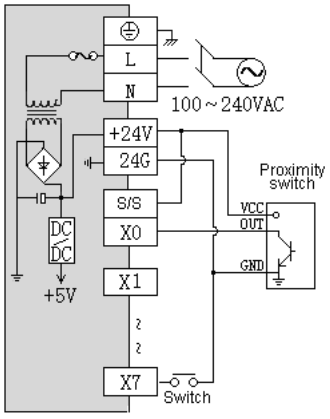
Note:the specific error code please check the module parameter register CR15,error code corresponding the content see detail the " CR parameter table".

3. I/O indicator declare

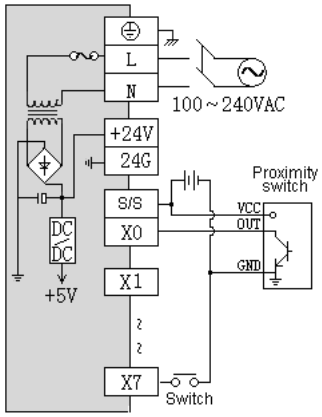
I/O indicator type	Indicate information	the state of the indicator
DI	Without signal Input	Not light
	With signalInput	Constant light
	Pulse signal input	Flicker(high frequency often bright)
DO	Without signal output	Not light
	With signal output	Constant light
	Pulse signal output	Flicker(high frequency often bright)
AI	Without signal Input	Not light
	With signal Input	Constant light
AQ	Without signal output (Channel abnomal)	Not light
	With signal output	Constant light

I/O wiring diagram

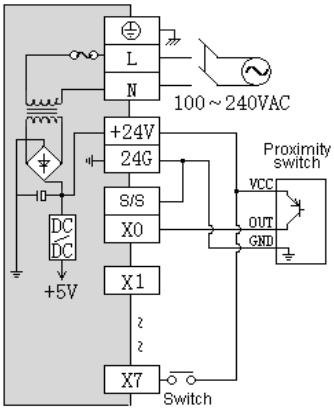
1. Digital Input (DI) wiring diagram



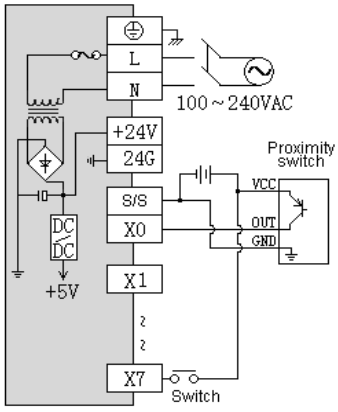
NPN Internal power



NPN External power

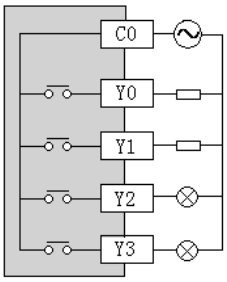


PNP Internal power

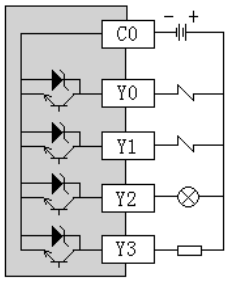


PNP External power

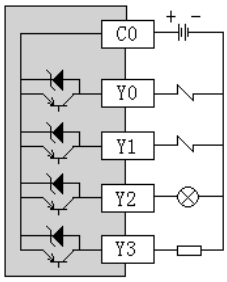
2. Digital output (DO) wiring diagram



AC/DC Relay output

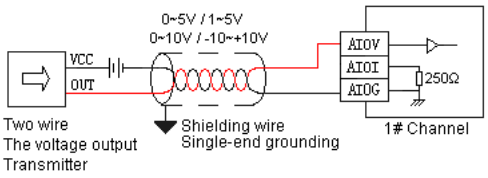


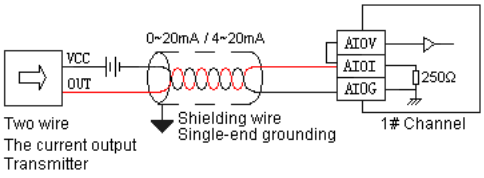
DC NPN Transistor output

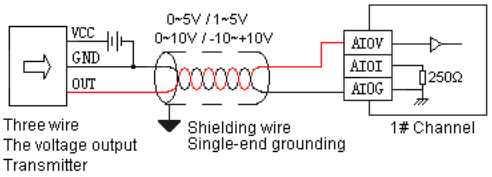


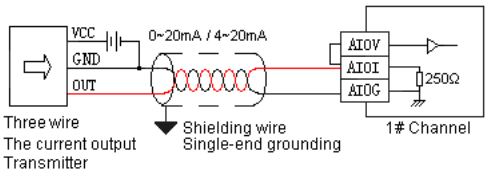
DC PNP Transistor output

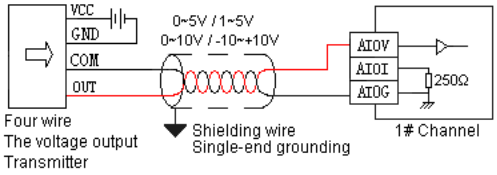
3. Analog input (AI) wiring diagram

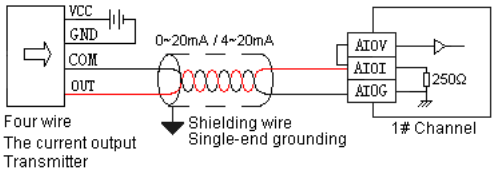




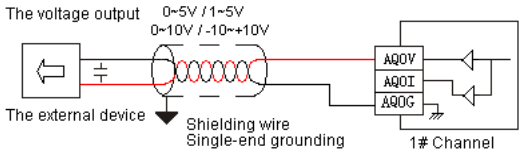


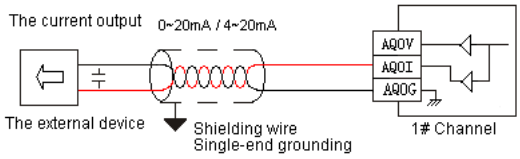




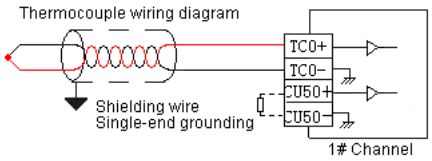


4. Analog output (AQ) wiring diagram

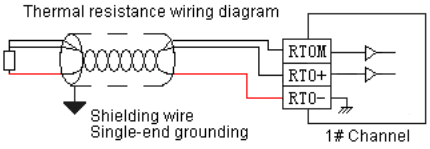




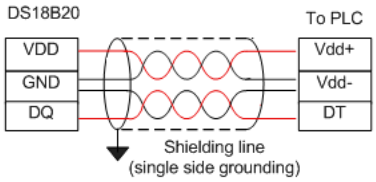
5. Thermocouple Input wiring diagram

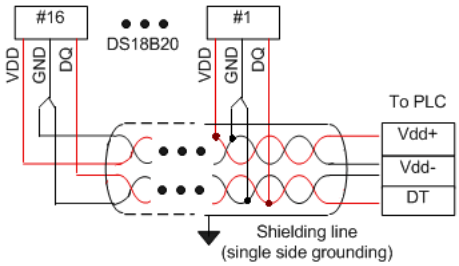


6. RTD Input wiring diagram

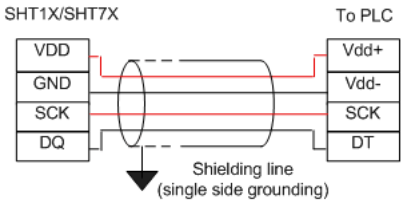


7. DS18B20, RW1820, DS1990 single or multiple sensor wiring diagram





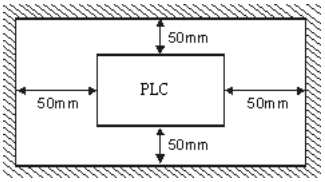
8. SHT1X or SHT7X sensor wiring diagram



PLC installation and precautions

[PLC installation]

PLC be installed,must be in closed switching box,installed any mode,PLC must can heat dissipation well,prevent temperature build-up,the plc be sure not to installed near the bottom side. up side and in vertical direction ,and keep space all around the PLC (as follows).



1. Rail mount :the rail specification :standard 35mm rail.

2. Screw mount :each MPU or extend module have two screw fix hole ,the diameter of the hole is 4.5mm,the position and distance please refer to the external dimension diagram.

3. Connect the extend module :extend module connect to the MPU by BUS,each extend module default comes with a extend cable when ex-factory for connect to the last module .Connection method : open tthe right expansion interface of the last module(MPU or extend module),insert the extend cable to the expansion interface,reset the renovate after fast the extend cable,the right expansion interface of the module for the next extend module.in this way connect all extend module.

[power supply and earth]

PLC power supply divide into AC and DC,pay attention to as follows when use:

1. AC supply Input voltage is 100~240VAC 50/60Hz,two line of the power supply (power line and zero line) please connect to the L. N terminal blocks,please connect the power line (L) to the L terminal ,connect the zero line to the N terminal.

2. DC supply Input voltage is 24VDC -15%~+20%,the positive and negative of the power supply connect +24V. 0V terminals.

3. if connect the AC110V or AC220V to +24V terminal or Input point terminals ,the PLC will be damaged ,please pay special attention to it.

4. Please connect the PLC to earth correctly ,the diameter of the line must be above 1.6mm.

[Matters need attention]

1. Please don't mount the PLC within dust . lamblack . electric conduction dust and corrosivity or combustible gas .Please don't mount the PLC within the environment high temperature . moisture condensation .

2. The machine is a OPEN TYPE enclosure ,so it must be used within the environment dustproof . moistureproof . anti-corrosionbe . safe against electric shock and impact of external forces .Must have safeguard procedures (as: opened by special tools or key) prevent not maintenance personnel operated or unexpected shock the machine, cause dangerous and damage.

3. When holemaking the screw hole and wiring ,don't drop-in the scrap iron or wire head to the machine, potential trigger fire hazard . fault or malfunction.

4. There is a seal cover the aspirail of the PLC when ex-factory, use for prevent dust or foreign bodies invaded, before wiring, don't wipe off, after wiring , before proceed power on the PLC, please evulsion the seal cover, in order to heat dissipation very well. Otherwise potential trigger fire hazard. fault or malfunction.

5. Please secure mount the connecting line and all kinds of the extend modules , bad contact potential malfunction.

6. Keep 50mm space all around the PLC , and far away the high voltage line and large electric power to the greatest extend.

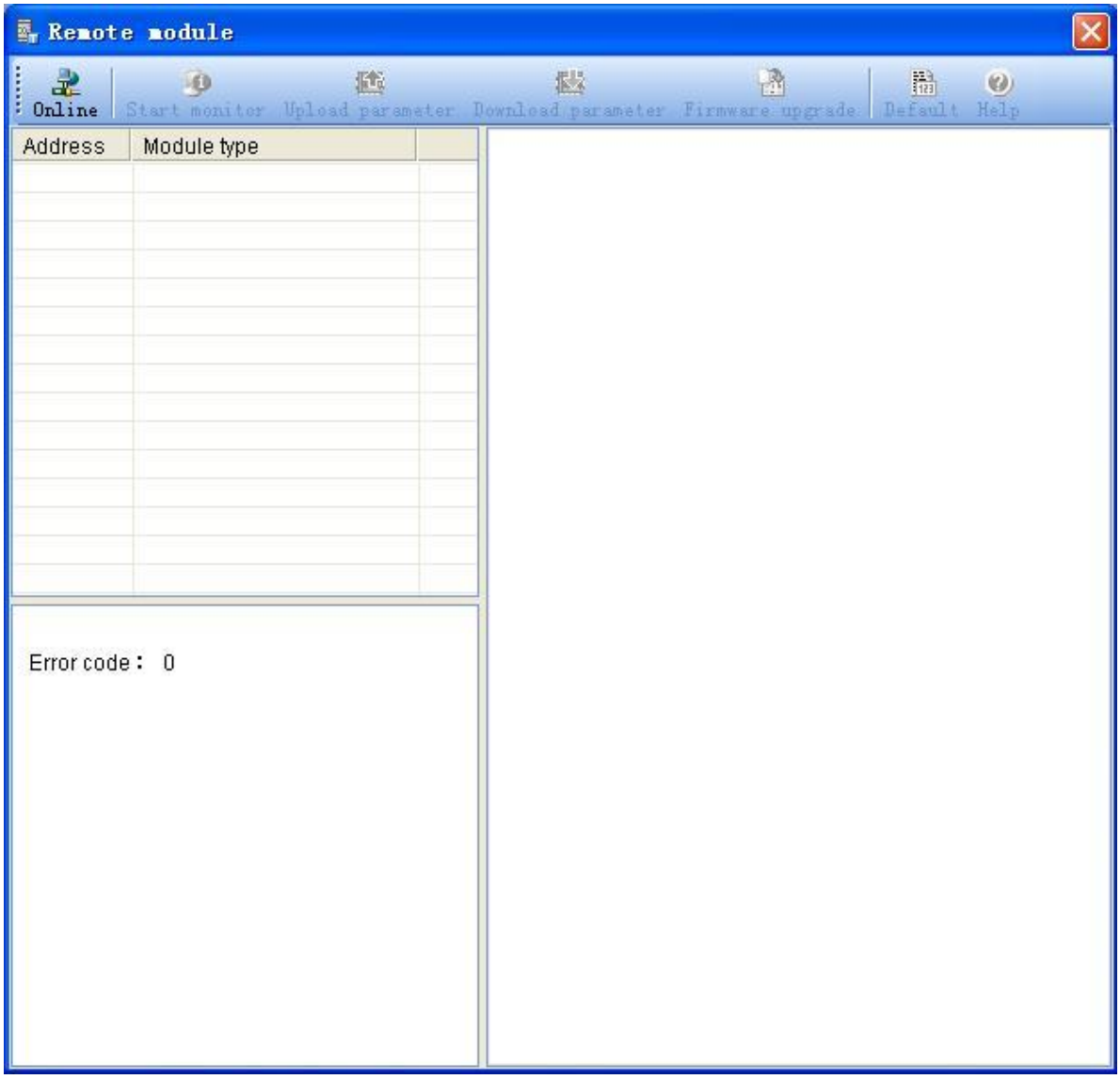
Remote module

The section introduce the usage of PLC extend module use for remote IO module

Overview

PLC extend module built-in a RS485 communication port , support parallel bus (use extend bus connect to parallel interface of PLC MPU) also support serial bus(use RS485 communication port and communication port of PLC MPU networking , master use communication instruction control remote module),when use serial bus to expand (as remote IO module), expand don't limited by the system points, may be distributed installation . This very important for a large number of disperse digital or analog signal(temperature. humidity. differential pressure. blowing rate. flow. fan rotate speed. valve open degree etc.) need be sampled and monitored in the system, easy realize distributed installation moreover no expand point limited , vastly improve control system configuration flexible and in the future expand capacity, reduce wire and work load of all kinds of signal , meanwhile reduce disturb because analog signal wire too long, save project invest cost.

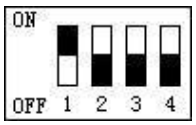
Click menu [Tools/Remote module] or click tool bar button , open " remote module " manage window. All control the remote module via the window complete.



Module station number setting up

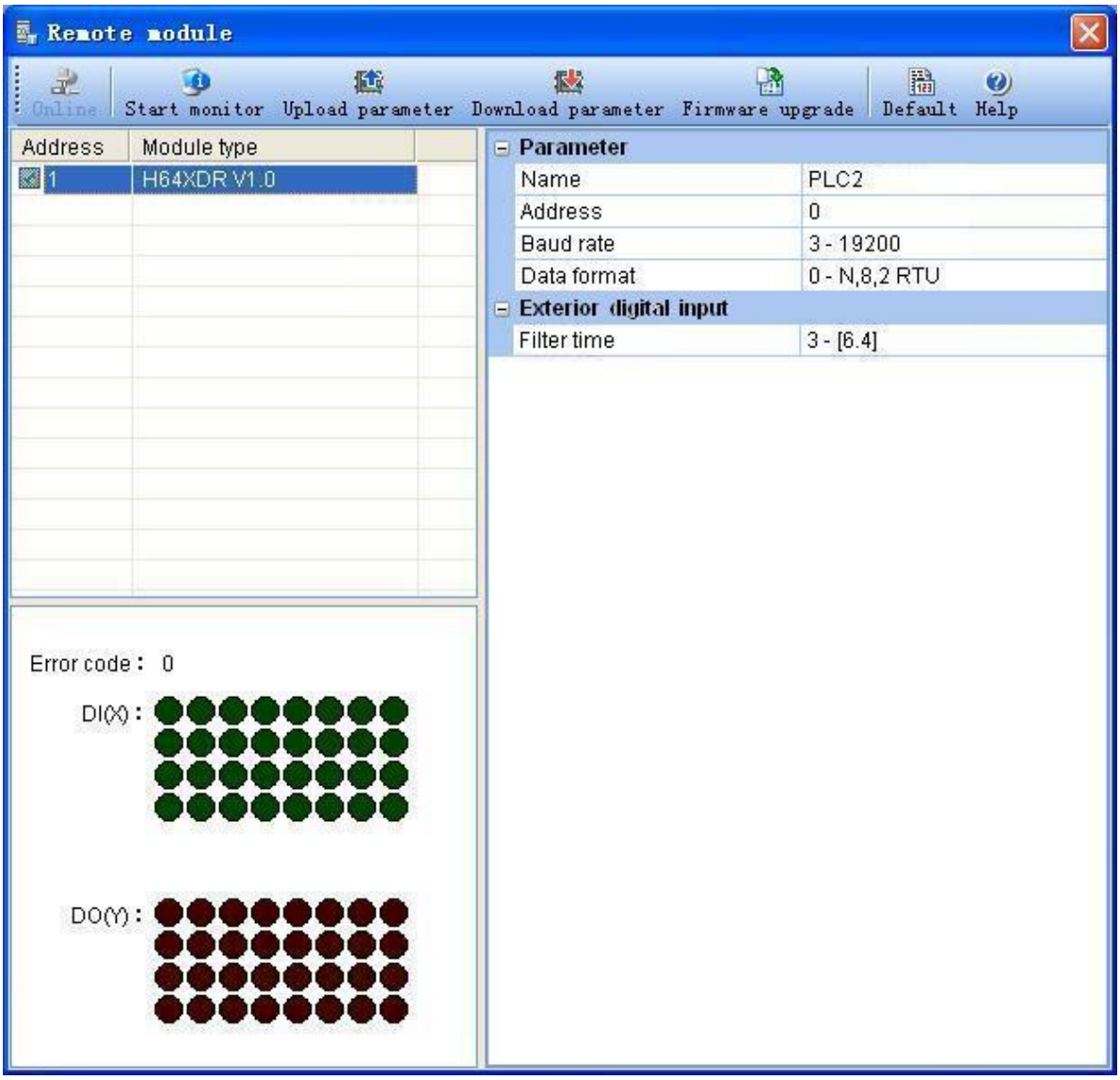
Via hardware 4 bit DIP switch setting up remote module station number(also name station address. communication address). legal

station number is 1~254(0 is broadcast address).4 bit DIP may be express decimal number 1~15(binary number from 0001 ~ 1111), method of modify the station number via DIP switch as follow:



Up picture is the 4 bit DIP switch use to set the station number of the module, above is ON, below is OFF, in the picture black part express the switch position, the bit set side to ON express the bit is 1, set side to OFF express the bit is 0, first bit in up picture is ON, other bits are OFF. The first bit of the DIP switch express the 0 bit of binary(b0), the fourth bit of the DIP switch express third bit of binary(b3),thus ,such as the DIP switch in up picture express 0001,also decimal data is 1, express the module station is 1(default is 1 when leave factory); If set the 1. 2 bits to ON side, moreover others witch set to OFF side ,then be 0011, also decimal data is 3, express module station number is 3.

If 4xbit DIP switch express the hardware address not enough to use or there is no DIP switch of the module, may be use programming software set the station number. after online with the module may be modify the module station number in " Communication parameter " column "Address", then click " Parameter download", change the module station number.



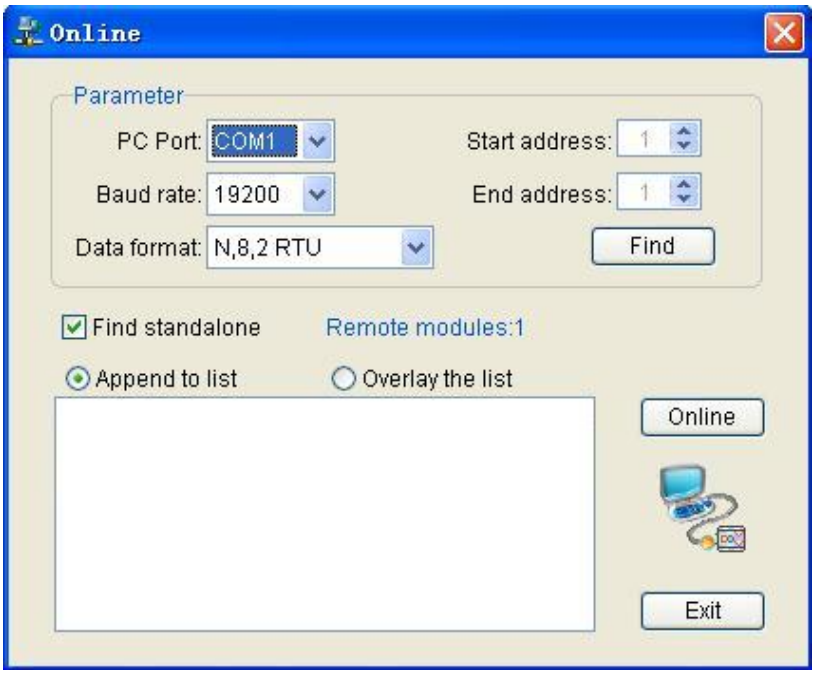
Note :If there is external DIP switch of the module, the address set by switch will be granted (4 bit DIP switch may be set the address range is 1~15);16 and above address or there is no DIP switch of the module then the address be set by programming software will be granted.

Online with module

Connect to one or more module on the network. Because the module is RS485 communication, so must use RS232/485 converter convert the computer RS232 communication port to RS485 then online the module. After only online with the module succeed ,then may be control the module on the network ,such as up down load . parameter. online monitor etc. .

Click " remote module " window 

button , open " Online " window .



Set the related parameters , general condition use the default parameter need not modify it.

- PC port number: select the computer serial port communicate with the module. Useable COM serial port number in the listing different according to the computer , system will automatic search all useable COM serial port of the computer to user for select by user .
- Baud rate: select communication baud rate, system default is 19200.
- Data format: select communication data format, system default is "N,8,2 RTU".
- Start address. end address: if communicate to single module, use "Stand-alone search" option ; if communicate to many modules, then "Start address" input the minimum station number of the module, "End address" input the maximum station number of the module.
- Append to listing. cover listing: select the already online modules use append or cover to the listing.
- Stand-alone search: if communicate to single module, use "Stand-alone search" option ; if communicate to many modules, then must cancel "Stand-alone search" option moreover set "Start address" and "End address".

Online operation

- Online: direct click "Online" button, searched module (online succeed) will be display in the listing.
- Search: if forget about the baud rate etc. communication parameters, may be click "Find" button to search, search function will try all baud rate . all data format to communicate with the module, so need spend longer time for waitting for search the module, searched module (online succeed) will be display in the listing.

Click "Exit", close "Online" window .

If online unsuccessful ,chances are below cause, look over to exclusion the problem, if can not exclusion please connect the technical support .

A. Selected PC serial port not correct

B. Selected communication parameter and the communication parameter of the module are different

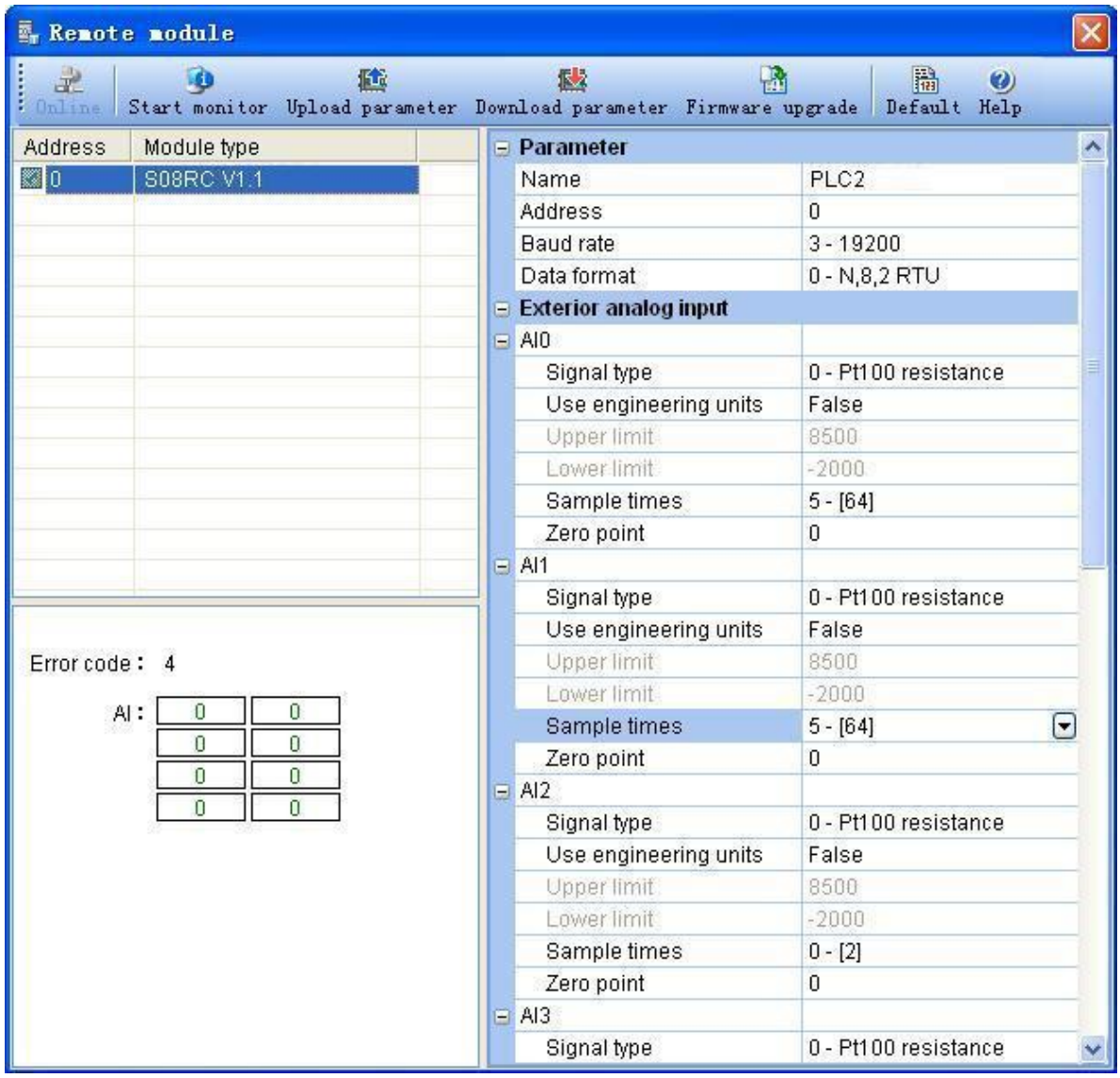
C. Module power off

D. Communication cable connected wrong or not connected well

E. Use "USB convert to RS232 serial port data line" not correct installed the driver

Parameter modify

After online succeed ,parameters of the module will be upload automatic, as follow:



A. On the left side of the upper is module listing , here display the already connected modules. May be double click any module in the listing to select be current module, all online operation of the module as: upload download parameter. firmware upgrade etc. only fro current module, don't influence others module.

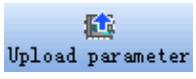
B. On the left side of the bottom is monitor zone , online monitoring use for display current module real time data.

Righ side parameter of module different according to different module type, major include signal type. use quantities. quantities upper lower limit. sampling number of times. zero revise . filtering time etc.. Refer to hardware manual "Extend module parameter".

User can modify the parameter arbitrarily, then via " parameter download " button download to module.

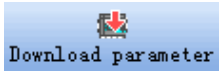
Note :signal type selected must the same as signal type of external sensor .

Parameter upload

Click 

button will upload and display the parameter of current module at " remote module " window .

Parameter download

Click 

button will download the modified parameter to the object module.

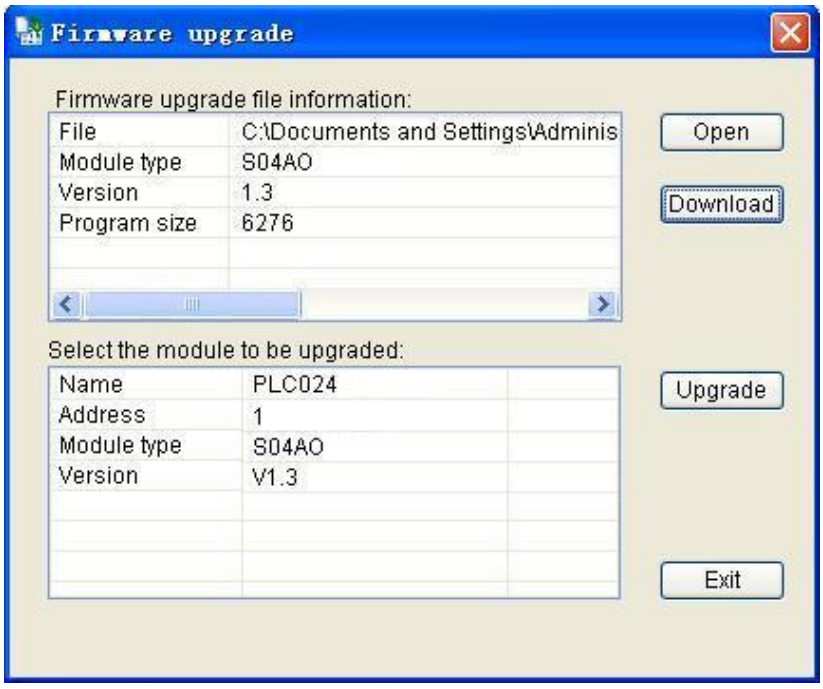
Note :any parameter be modified, only take effect after via "Parameter download" download to object module .There is not execute "Parameter download" operation, parameters of object module will be no change.

Firmware upgrade

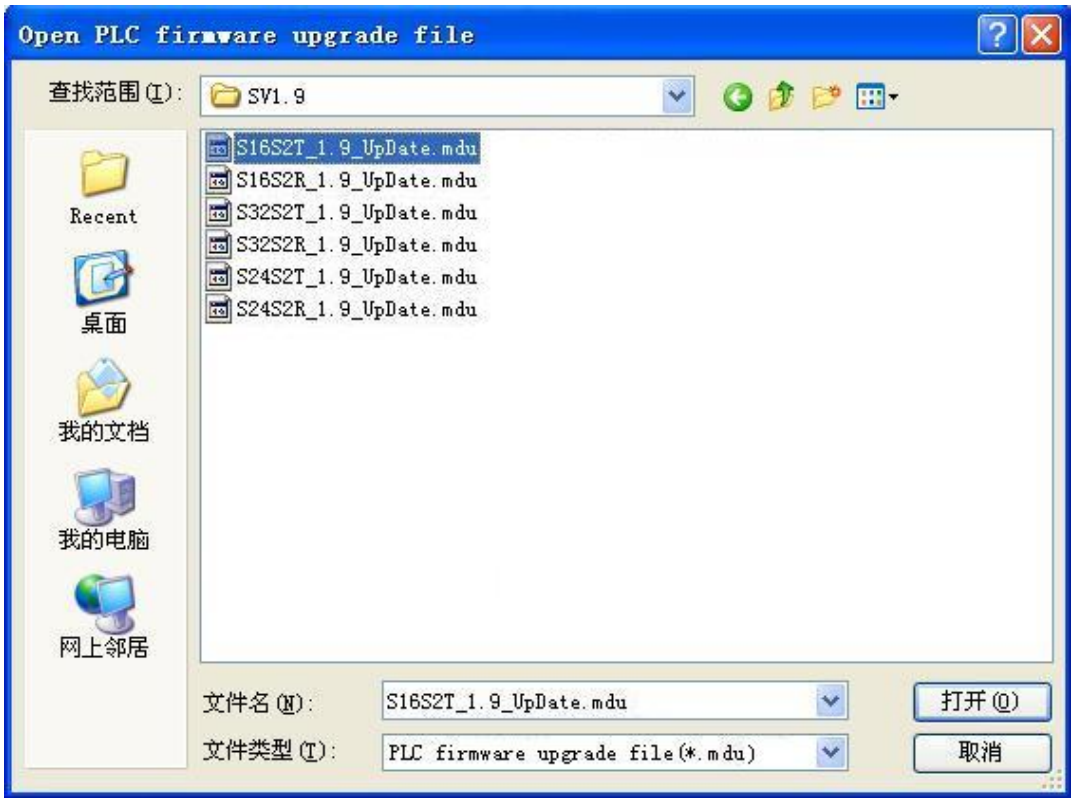
Upgrade the firmware of the module, make the module support new function.

Click 

button ,, open " firmware upgrade" window



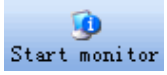
Click " open " button , select the upgrade file according to the module.



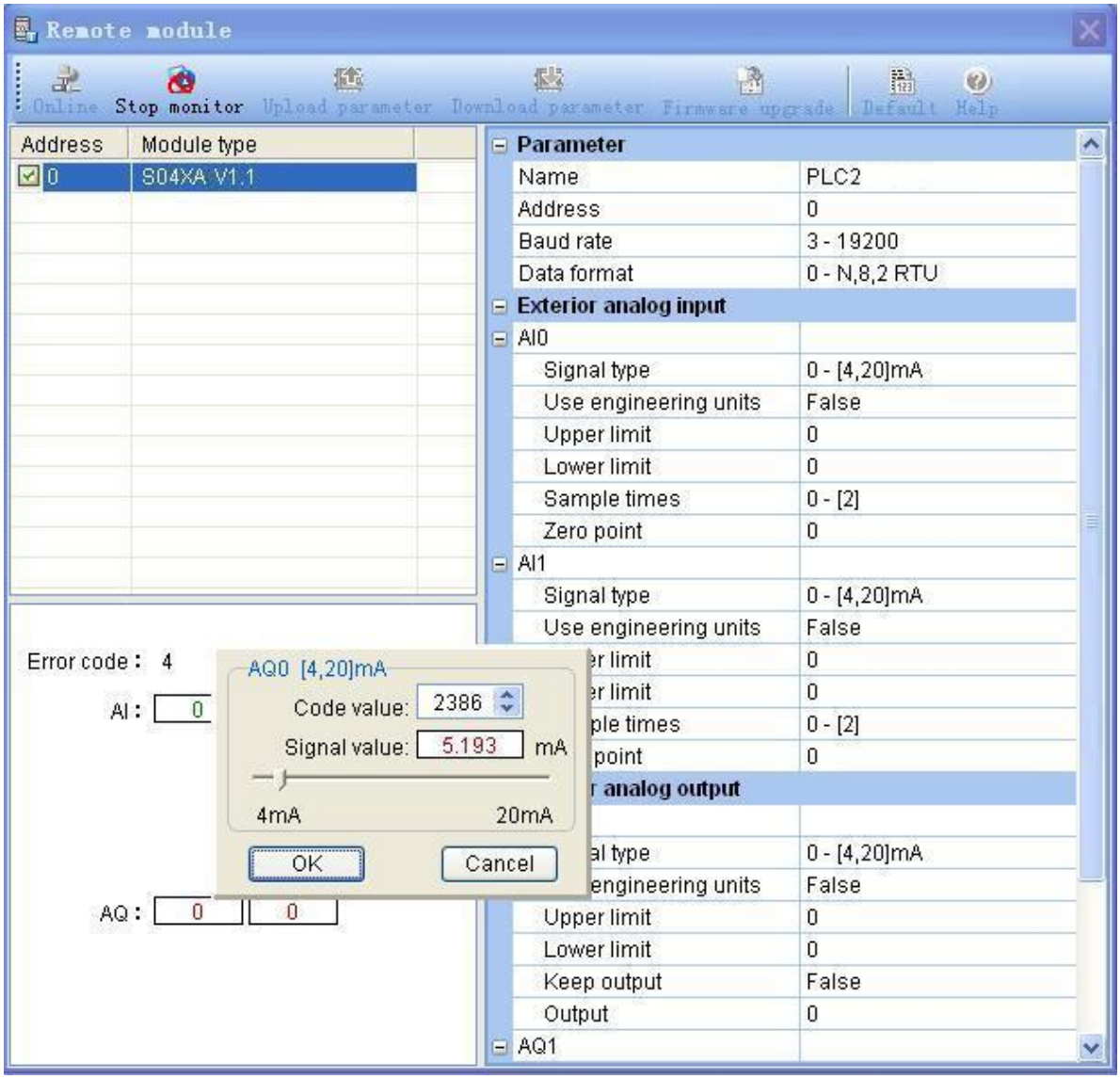
Click "Upgrade" button ,upgrade the firmware of the module.

Note :If interrupt during upgrading because power of or other reasons ,must rerun firmware upgrade until upgrade successful.

Online monitor

Click 

button enter into monitor status. Real time monitor input channel data of current module , also may force change the output data of the opu module .



Click  button exit monitor status.

Appendix

List the resource of the system in PLC .

SM system status bit

SM system status bit is a group of special internal relay of the system,can be used unlimited in the program, each SM has a special function.Do not use the SM which unlisted.

SM	Function declare	R/W	Power-off preserve	Default
SM0	On during running, Off during stopping	R	No	0
SM1	Off during running, On during stopping	R	No	0
SM2	On during the first scan when PLC starts RUN and then be Off	R	No	0
SM3	10ms clock pulse	R	No	0
SM4	100ms clock pulse	R	No	0
SM5	1s clock pulse	R	No	0
SM8	Scan time-out	R	No	0
SM9	PLC switch status	R	No	0
SM10	Run status	R	No	0
SM11	System failure	R	No	0
SM12	Hardware configure table mismatch the module	R	No	0
SM13	Battery in low voltage, malfunction or no battery	R	No	0

SM14	Divide by zero flag	R	No	0
SM15	Data overflow flag	R	No	0
SM16	COM1 communicate error	R	No	0
SM17	COM2 communicate error	R	No	0
SM18	COM3 communicate error	R	No	0
SM19	COM4 communicate error	R	No	0
SM20	COM5 communicate error	R	No	0
SM25	HSC0 study enable control,0 is normal state,1 is study state	R/W	No	0
SM26	HSC0 study confirm control	R/W	No	0
SM27	HSC0 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM30	HSC0 direction indication,0 is increase,1 is decrease	R	No	0
SM31	HSC0 error indication	R	No	0
SM33	HSC1 study enable control,0 is normal state,1 is study state	R/W	No	0
SM34	HSC1 study confirm control	R/W	No	0
SM35	HSC1 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM38	HSC1 direction indication,0 is increase,1 is decrease	R	No	0

SM39	HSC1 error indication	R	No	0
SM41	HSC2 study enable control,0 is normal state,1 is study state	R/W	No	0
SM42	HSC2 study confirm control	R/W	No	0
SM43	HSC2 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM46	HSC2 direction indication,0 is increase,1 is decrease	R	No	0
SM47	HSC2 error indication	R	No	0
SM49	HSC3 study enable control,0 is normal state,1 is study state	R/W	No	0
SM50	HSC3 study confirm control	R/W	No	0
SM51	HSC3 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM54	HSC3 direction indication,0 is increase,1 is decrease	R	No	0
SM55	HSC3 error indication	R	No	0
SM57	HSC4 study enable control,0 is normal state,1 is study state	R/W	No	0
SM58	HSC4 study confirm control	R/W	No	0
SM59	HSC4 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM62	HSC4 direction indication,0 is increase,1 is decrease	R	No	0
SM63	HSC4 error indication	R	No	0

SM65	HSC5 study enable control,0 is normal state,1 is study state	R/W	No	0
SM66	HSC5 study confirm control	R/W	No	0
SM67	HSC5 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM70	HSC5 direction indication,0 is increase,1 is decrease	R	No	0
SM71	HSC5 error indication	R	No	0
SM73	HSC6 study enable control,0 is normal state,1 is study state	R/W	No	0
SM74	HSC6 study confirm control	R/W	No	0
SM75	HSC6 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM78	HSC6 direction indication,0 is increase,1 is decrease	R	No	0
SM79	HSC6 error indication	R	No	0
SM81	HSC7 study enable control,0 is normal state,1 is study state	R/W	No	0
SM82	HSC7 study confirm control	R/W	No	0
SM83	HSC7 reset control 0 is automatic reset 1 is not reset	R/W	No	0
SM86	HSC7 direction indication,0 is increase,1 is decrease	R	No	0
SM87	HSC7 error indication	R	No	0

SM93	PLS0 prohibit the forward pulse	R/W	yes	0
SM94	PLS0 prohibit the reverse pulse	R/W	yes	0
SM95	PLS0 prohibit the brake function	R/W	yes	0
SM96	PLS0 pulse output indication	R	yes	0
SM97	PLS0 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM98	PLS0 error flag	R	yes	0
SM99	PLS0 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM100	PLS0 pulse output complete	R	yes	0
SM109	PLS1 prohibit the forward pulse	R/W	yes	0
SM110	PLS1 prohibit the reverse pulse	R/W	yes	0
SM111	PLS1 prohibit the brake function	R/W	yes	0
SM112	PLS1 pulse output indication	R	yes	0
SM113	PLS1 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM114	PLS1 error flag	R	yes	0
SM115	PLS1 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM116	PLS1 pulse output complete	R	yes	0
SM125	PLS2 prohibit the forward pulse	R/W	yes	0

SM126	PLS2 prohibit the reverse pulse	R/W	yes	0
SM127	PLS2 prohibit the brake function	R/W	yes	0
SM128	PLS2 pulse output indication	R	yes	0
SM129	PLS2 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM130	PLS2 error flag	R	yes	0
SM131	PLS2 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM132	PLS2 pulse output complete	R	yes	0
SM141	PLS3 prohibit the forward pulse	R/W	yes	0
SM142	PLS3 prohibit the reverse pulse	R/W	yes	0
SM143	PLS3 prohibit the brake function	R/W	yes	0
SM144	PLS3 pulse output indication	R	yes	0
SM145	PLS3 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM146	PLS3 error flag	R	yes	0
SM147	PLS3 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM148	PLS3 pulse output complete	R	yes	0
SM157	PLS4 prohibit the forward pulse	R/W	yes	0
SM158	PLS4 prohibit the reverse pulse	R/W	yes	0

SM159	PLS4 prohibit the brake function	R/W	yes	0
SM160	PLS4 pulse output indication	R	yes	0
SM161	PLS4 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM162	PLS4 error flag	R	yes	0
SM163	PLS4 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM164	PLS4 pulse output complete	R	yes	0
SM173	PLS5 prohibit the forward pulse	R/W	yes	0
SM174	PLS5 prohibit the reverse pulse	R/W	yes	0
SM175	PLS5 prohibit the brake function	R/W	yes	0
SM176	PLS5 pulse output indication	R	yes	0
SM177	PLS5 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM178	PLS5 error flag	R	yes	0
SM179	PLS5 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM180	PLS5 pulse output complete	R	yes	0
SM189	PLS6 prohibit the forward pulse	R/W	yes	0
SM190	PLS6 prohibit the reverse pulse	R/W	yes	0
SM191	PLS6 prohibit the brake function	R/W	yes	0

SM192	PLS6 pulse output indication	R	yes	0
SM193	PLS6 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM194	PLS6 error flag	R	yes	0
SM195	PLS6 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM196	PLS6 pulse output complete	R	yes	0
SM205	PLS7 prohibit the forward pulse	R/W	yes	0
SM206	PLS7 prohibit the reverse pulse	R/W	yes	0
SM207	PLS7 prohibit the brake function	R/W	yes	0
SM208	PLS7 pulse output indication	R	yes	0
SM209	PLS7 pulse output direction indication ,0 is forward ,1 is reverse	R	yes	0
SM210	PLS7 error flag	R	yes	0
SM211	PLS7 position model ,0 is relative address ,1 is absolute address	R/W	yes	0
SM212	PLS7 pulse output complete	R	yes	0
SM216	Whether to match the station number during Modbus TCP communication, 0 means mismatch, 1 means match.	R/W	yes	0

SV system register

SV system register is a group of special internal register of the system,can be used unlimited in the program, each SV has a special function.Do not use the SM which unlisted.

SV	Function declare	R/W	Power-off preserve	Default
SV0	The present scan time(unit 0.1ms)	R	No	0
SV1	The minimum scan time(unit 0.1ms)	R	No	0
SV2	The maximum scan time(unit 0.1ms)	R	No	0
SV3	System fault code,detail see the system fault code table	R	No	0
SV4	COM1 communicate error code	R	No	0
SV5	COM2 communicate error code	R	No	0
SV6	COM3 communicate error code	R	No	0
SV7	COM4 communicate error code	R	No	0
SV8	COM5 communicate error code	R	No	0
SV9	Modbus TCP client port settings, server fixed 502	R/W	yes	0
SV11	AI input on the CPU module break off alarm every bit express one channel 0-Normal 1-break off	R	No	0
SV12	Year	R	No	0
SV13	Month(1-12)	R	No	0
SV14	Day(1-31)	R	No	0
SV15	Hour(0-23)	R	No	0
SV16	Minute(0-59)	R	No	0

SV17	Second(0-59)	R	No	0
SV18	Week(1-7,Monday~Sunday)	R	No	0
SV19	PLC station's name	R/W	yes	0
SV20	PLC station's name	R/W	yes	0
SV21	PLC station's name	R/W	yes	0
SV22	PLC station's name	R/W	yes	0
SV23	PLC station's name	R/W	yes	0
SV24	PLC station's name	R/W	yes	0
SV25	Timer of program scan time-out(unit ms)	R/W	yes	200 ms
SV26	PLC address 1~254	R	yes	1
SV27	Low byte is extend modules 0~31 High byte is type	R	yes	0
SV28	Low byte is CPU’s type High byte is CPU’s version	R	yes	0
SV29	Low byte is first extend module's code High byte is first extend module's version	R	yes	0
SV30	Low byte is second extend module's code High byte is second extend module's version	R	yes	0
SV31	Low byte is third extend module's code High byte is third extend module's version	R	yes	0
SV32	Low byte is fourth extend module's code High byte is fourth extend module's	R	yes	0

	version			
SV33	Low byte is fifth extend module's code High byte is fifth extend module's version	R	yes	0
SV34	Low byte is sixth extend module's code High byte is sixth extend module's version	R	yes	0
SV35	Low byte is seventh extend module's code High byte is seventh extend module's version	R	yes	0
SV36	Low byte is eighth extend module's code High byte is eighth extend module's version	R	yes	0
SV37	Low byte is ninth extend module's code High byte is ninth extend module's version	R	yes	0
SV38	Low byte is tenth extend module's code High byte is tenth extend module's version	R	yes	0
SV39	Low byte is eleventh extend module's code High byte is eleventh extend module's version	R	yes	0
SV40	Low byte is twelfth extend module's code High byte is twelfth extend module's version	R	yes	0
SV41	Low byte is thirteenth extend module's code High byte is thirteenth extend module's version	R	yes	0
SV42	Low byte is ffourteenth extend module's code High byte is fourteenth extend module's version	R	yes	0
SV43	Low byte is fifteenth extend module's code High byte is fifteenth extend module's version	R	yes	0

SV44	COM1 communicate protocol: Low 4 bit of low byte:0 - N,8, 2 For RTU 1 - E,8, 1 For RTU 2 - O 8, 1 For RTU 3 - N,7, 2 For ASCII 4 - E,7, 1 For ASCII 5 - O,7, 1 For ASCII 6 - N,8, 1 For RTU(H/N serial support) High 4 bit of low byte:0 - 2400 1 - 4800 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200(H/N serial support)	R/W	yes	0x30,19200, N,8, 2 RTU
SV45	COM1 and Ethernet communicate overtime ,unit ms	R/W	yes	200ms
SV46	COM2 communicate protocol,the same as COM1	R/W	yes	0x30,19200, N,8, 2 RTU
SV47	COM2 communicate overtime ,unit ms	R/W	yes	200ms
SV48	PLC program size	R	yes	0
SV49	Low byte of system clock ,unit 16us Max 1073741824	R	yes	
SV50	High byte of system clock ,unit 16us Max 1073741824	R	yes	
SV54	COM3 communicate protocol,the same as COM1	R/W	yes	0x30,19200, N,8, 2 RTU
SV55	COM3 communicate overtime ,unit ms	R/W	yes	200ms
SV56	COM4 communicate protocol,the same as COM1	R/W	yes	0x30,19200, N,8, 2 RTU
SV57	COM4 communicate overtime ,unit ms	R/W	yes	200ms

SV58	COM5 communicate protocol,the same as COM1	R/W	yes	0x30,19200, N,8, 2 RTU
SV59	COM5 communicate overtime ,unit ms	R/W	yes	200ms
SV60	HSC0 current segment number	R	yes	0
SV61	HSC0 low word of current value	R	yes	0
SV62	HSC0 high word of current value	R	yes	0
SV63	HSC0 error code	R	yes	0
SV64	HSC1 current segment number	R	yes	0
SV65	HSC1 low word of current value	R	yes	0
SV66	HSC1 high word of current value	R	yes	0
SV67	HSC1 error code	R	yes	0
SV68	HSC2 current segment number	R	yes	0
SV69	HSC2 low word of current value	R	yes	0
SV70	HSC2 high word of current value	R	yes	0
SV71	HSC2 error code	R	yes	0
SV72	HSC3 current segment number	R	yes	0
SV73	HSC3 low word of current value	R	yes	0
SV74	HSC3 high word of current value	R	yes	0
SV75	HSC3 error code	R	yes	0

SV76	HSC4 current segment number	R	yes	0
SV77	HSC4 low word of current value	R	yes	0
SV78	HSC4 high word of current value	R	yes	0
SV79	HSC4 error code	R	yes	0
SV80	HSC5 current segment number	R	yes	0
SV81	HSC5 low word of current value	R	yes	0
SV82	HSC5 high word of current value	R	yes	0
SV83	HSC5 error code	R	yes	0
SV84	HSC6 current segment number	R	yes	0
SV85	HSC6 low word of current value	R	yes	0
SV86	HSC6 high word of current value	R	yes	0
SV87	HSC6 error code	R	yes	0
SV88	HSC7 current segment number	R	yes	0
SV89	HSC7 low word of current value	R	yes	0
SV90	HSC7 high word of current value	R	yes	0
SV91	HSC7 error code	R	yes	0
SV92	PLS0 current segment number	R	yes	0
SV93	PLS0 low word of pulse output number	R	yes	0
SV94	PLS0 high word of pulse output number	R	yes	0

SV95	PLS0 low word of current position	R/W	yes	0
SV96	PLS0 high word of current position	R/W	yes	0
SV97	PLS0 error code	R	yes	0
SV98	PLS1 current segment number	R	yes	0
SV99	PLS1 low word of pulse output number	R	yes	0
SV100	PLS1 high word of pulse output number	R	yes	0
SV101	PLS1 low word of current position	R/W	yes	0
SV102	PLS1 high word of current position	R/W	yes	0
SV103	PLS1 error code	R	yes	0
SV104	PLS2 current segment number	R	yes	0
SV105	PLS2 low word of pulse output number	R	yes	0
SV106	PLS2 high word of pulse output number	R	yes	0
SV107	PLS2 low word of current position	R/W	yes	0
SV108	PLS2 high word of current position	R/W	yes	0
SV109	PLS2 error code	R	yes	0
SV110	PLS3 current segment number	R	yes	0
SV111	PLS3 low word of pulse output number	R	yes	0
SV112	PLS3 high word of pulse output number	R	yes	0

SV113	PLS3 low word of current position	R/W	yes	0
SV114	PLS3 high word of current position	R/W	yes	0
SV115	PLS3 error code	R	yes	0
SV116	PLS4 current segment number	R	yes	0
SV117	PLS4 low word of pulse output number	R	yes	0
SV118	PLS4 high word of pulse output number	R	yes	0
SV119	PLS4 low word of current position	R/W	yes	0
SV120	PLS4 high word of current position	R/W	yes	0
SV121	PLS4 error code	R	yes	0
SV122	PLS5 current segment number	R	yes	0
SV123	PLS5 low word of pulse output number	R	yes	0
SV124	PLS5 high word of pulse output number	R	yes	0
SV125	PLS5 low word of current position	R/W	yes	0
SV126	PLS5 high word of current position	R/W	yes	0
SV127	PLS5 error code	R	yes	0
SV128	PLS6 current segment number	R	yes	0
SV129	PLS6 low word of pulse output number	R	yes	0
SV130	PLS6 high word of pulse output number	R	yes	0
SV131	PLS6 low word of current position	R/W	yes	0

SV132	PLS6 high word of current position	R/W	yes	0
SV133	PLS6 error code	R	yes	0
SV134	PLS7 current segment number	R	yes	0
SV135	PLS7 low word of pulse output number	R	yes	0
SV136	PLS7 high word of pulse output number	R	yes	0
SV137	PLS7 low word of current position	R/W	yes	0
SV138	PLS7 high word of current position	R/W	yes	0
SV139	PLS7 error code	R	yes	0
SV140	When value is -23206 prohibit all output of Y	R/W	yes	0
SV141	COM1 Communicate instruction execute interval unit ms	R/W	yes	0
SV142	The soft address of PLC(1~254)	R	yes	0
SV143	The setted address of the external DIP switch	R	yes	0
SV144	Low word of serial number	R	yes	0
SV145	High word of serial number	R	yes	0
SV146	Time of the direction output before the pulse output(5~100us)	R/W	yes	5
SV151	Number of locked data	R	yes	0
SV152	IP address,default : 192.168.1.111	RW	yes	0x016F

SV153	IP address,default : 192.168.1.111	RW	yes	0xC0A8
SV154	Subnet mask ,default: 255.255.255.0	R/W	yes	0xFF00
SV155	Subnet mask ,default: 255.255.255.0	R/W	yes	0xFFFF
SV156	PLS0 low word of mechanical original point	R/W	yes	0
SV157	PLS0 high word of mechanical original point	R/W	yes	0
SV158	PLS0 number of pulses to compensate the reverse interval	R/W	yes	0
SV159	PLS0 follow performance parameters, range: 1~100	R/W	yes	50
SV160	PLS1 low word of mechanical original point	R/W	yes	0
SV161	PLS1 high word of mechanical original point	R/W	yes	0
SV162	PLS1 number of pulses to compensate the reverse interval	R/W	yes	0
SV163	PLS1 follow performance parameters, range: 1~100	R/W	yes	50
SV164	PLS2 low word of mechanical original point	R/W	yes	0
SV165	PLS2 high word of mechanical original point	R/W	yes	0
SV166	PLS2 number of pulses to compensate the reverse interval	R/W	yes	0
SV167	PLS2 follow performance parameters,	R/W	yes	50

	range: 1~100			
SV168	PLS3 low word of mechanical original point	R/W	yes	0
SV169	PLS3 high word of mechanical original point	R/W	yes	0
SV170	PLS3 number of pulses to compensate the reverse interval	R/W	yes	0
SV171	PLS3 follow performance parameters, range: 1~100	R/W	yes	50
SV172	PLS4 low word of mechanical original point	R/W	yes	0
SV173	PLS4 high word of mechanical original point	R/W	yes	0
SV174	PLS4 number of pulses to compensate the reverse interval	R/W	yes	0
SV175	PLS4 follow performance parameters, range: 1~100	R/W	yes	50
SV176	PLS5 low word of mechanical original point	R/W	yes	0
SV177	PLS5 high word of mechanical original point	R/W	yes	0
SV178	PLS5 number of pulses to compensate the reverse interval	R/W	yes	0
SV179	PLS5 follow performance parameters, range: 1~100	R/W	yes	50
SV180	PLS6 low word of mechanical original point	R/W	yes	0

SV181	PLS6 high word of mechanical original point	R/W	yes	0
SV182	PLS6 number of pulses to compensate the reverse interval	R/W	yes	0
SV183	PLS6 follow performance parameters, range: 1~100	R/W	yes	50
SV184	PLS7 low word of mechanical original point	R/W	yes	0
SV185	PLS7 high word of mechanical original point	R/W	yes	0
SV186	PLS7 number of pulses to compensate the reverse interval	R/W	yes	0
SV187	PLS7 follow performance parameters, range: 1~100	R/W	yes	50
SV801	HSC0 low word of frequency	R	yes	0
SV802	HSC0 high word of frequency	R	yes	0
SV803	HSC1 low word of frequency	R	yes	0
SV804	HSC1 high word of frequency	R	yes	0
SV805	HSC2 low word of frequency	R	yes	0
SV806	HSC2 high word of frequency	R	yes	0
SV807	HSC3 low word of frequency	R	yes	0
SV808	HSC3 high word of frequency	R	yes	0
SV809	HSC4 low word of frequency	R	yes	0

SV810	HSC4 high word of frequency	R	yes	0
SV811	HSC5 low word of frequency	R	yes	0
SV812	HSC5 high word of frequency	R	yes	0
SV813	HSC6 low word of frequency	R	yes	0
SV814	HSC6 high word of frequency	R	yes	0
SV815	HSC7 low word of frequency	R	yes	0
SV816	HSC7 high word of frequency	R	yes	0
SV817	Historical fault code	R	yes	0
SV818	Historical fault code	R	yes	0
SV819	Historical fault code	R	yes	0
SV820	Historical fault code	R	yes	0
SV821	Historical fault code	R	yes	0
SV822	Historical fault code	R	yes	0
SV823	Historical fault code	R	yes	0
SV824	Historical fault code	R	yes	0
SV825	Historical fault code	R	yes	0
SV826	Historical fault code	R	yes	0
SV827	Historical fault code	R	yes	0
SV828	Historical fault code	R	yes	0

SV829	Historical fault code	R	yes	0
SV830	Historical fault code	R	yes	0
SV831	Historical fault code	R	yes	0
SV832	Historical fault code	R	yes	0
SV833	COM2 Communicate instruction execute interval unit ms	R/W	yes	0
SV834	COM3 Communicate instruction execute interval unit ms	R/W	yes	0
SV835	COM4 Communicate instruction execute interval unit ms	R/W	yes	0
SV836	COM5 Communicate instruction execute interval unit ms	R/W	yes	0
SV840	System status error code	R	yes	0
SV841	System status error code	R	yes	0
SV842	CPU firmware version date (low byte for year, high byte for month)	R	yes	0
SV843	CPU firmware version date (low byte for day, high byte for hour)	R	yes	0
SV844	FPGA firmware version date (low byte for year, high byte for month)	R	yes	0
SV845	FPGA firmware version date (low byte for day, high byte for hour)	R	yes	0
SV846	Gateway address (default :192.168.1.1)	R/W	yes	0x0101

SV847	Gateway address (default :192.168.1.1)	R/W	yes	0xC0A8
SV848	MAC address	R	yes	0
SV849	MAC address	R	yes	0
SV850	MAC address	R	yes	0
SV851	COM1 Communication port timeout exception in receiving characters(in milliseconds)	R/W	yes	0
SV852	COM2 Communication port timeout exception in receiving characters(in milliseconds)	R/W	yes	0
SV853	COM3 Communication port timeout exception in receiving characters(in milliseconds)	R/W	yes	0
SV854	COM4 Communication port timeout exception in receiving characters(in milliseconds)	R/W	yes	0
SV855	COM5 Communication port timeout exception in receiving characters(in milliseconds)	R/W	yes	0
SV864	Lora network packet ID	R/W	yes	1

System interruption table

PLCsupport 52 system interruptions,include pulse output. edge catch. high speed counter and timed interruption.

Interruption No.	Interruption type	Declare	Priority level
1	Pulse output interruption	PLS0 pulse output start	Hight to low
2		PLS0 pulse output complete	

Interruption No.	Interruption type	Declare	Priority level (the small interruption no, priority the big interruption no.)
3		PLS1 pulse output start	
4		PLS1 pulse output complete	
5		PLS2 pulse output start	
6		PLS2 pulse output complete	
7		PLS3 pulse output start	
8		PLS3 pulse output complete	
9		PLS4 pulse output start	
10		PLS4 pulse output complete	
11		PLS5 pulse output start	
12		PLS5 pulse output complete	
13		PLS6 pulse output start	
14		PLS6 pulse output complete	
15		PLS7 pulse output start	
16		PLS7 pulse output complete	
17	Edge catch interruption	X0 rise edge catch	
18		X1 rise edge catch	
19		X2 rise edge catch	
20		X3 rise edge catch	
21		X4 rise edge catch	
22		X5 rise edge catch	
23		X6 rise edge catch	

Interruption No.	Interruption type	Declare	Priority level
24		X7 rise edge catch	
25		X0 drop edge catch	
26		X1 drop edge catch	
27		X2 drop edge catch	
28		X3 drop edge catch	
29		X4 drop edge catch	
30		X5 drop edge catch	
31		X6 drop edge catch	
32		X7 drop edge catch	
33	High speed counter interruption	HSC0 current value=preset value(each segment preset be generated)	
34		HSC0 input direction changed	
35		HSC1 current value=preset value(each segment preset be generated)	
36		HSC1 input direction changed	
37		HSC2 current value=preset value(each segment preset be generated)	
38		HSC2 input direction changed	
39		HSC3 current value=preset value(each segment preset be generated)	
40		HSC3 input direction changed	
41		HSC4 current value=preset value(each segment preset be generated)	
42		HSC4 input direction changed	
43		HSC5 current value=preset value(each segment preset be generated)	
44		HSC5 input direction changed	

Interruption No.	Interruption type	Declare	Priority level
45		HSC6 current value=preset value(each segment preset be generated)	
46		HSC6 input direction changed	
47		HSC7 current value=preset value(each segment preset be generated)	
48		HSC7 input direction changed	
49	Timed interruption	T252 timer reaches target	
50		T253 timer reaches target	
51		T254 timer reaches target	
52		T255 timer reaches target	

Communication address code table

1. PLC bit component table(equivalently Modbus address type 0. 1,support Modbus function code 1. 2. 5. 15)

Component	Name	Component range	Read/Write	Modbus communication address code		Declare
				Hexadecimal	Decimal	
X	External input	X0~X1023	R	0x0000~0x03FF	0~1023	
Y	External output	Y0~Y1023	R/W	0x0600~0x09FF	1536~2559	
M	Auxiliary relay	M0~M12287	R/W	0x0C00~0x3BFF	3072~15359	
T	Timer(output coil)	T0~T1023	R/W	0x3C00~0x3FFF	15360~16383	
C	Counter(output coil)	C0~C255	R/W	0x4000~0x40FF	16384~16639	
SM	System status bit	SM0~SM215	R/W	0x4200~0x42D7	16896~17111	
S	Step relay	S0~S2047	R/W	0x7000~0x77FF	28672~30719	

2. PLC component table(equivalently Modbus address type 3. 4,sepport Modbus function code 3. 4. 6. 16)

Component	Name	Component range	Read/Write	Modbus commucation address code	Declare

Component	Name	Component range	Read/Write	Modbus communication address code		Declare
				Hexadecimal	Decimal	
CR	Extend module parameter	CR0~CR255	R/W	**0x00~0xFF**	**0~255**	Use Modbus protocol to a extend module
AI	Analog input register	AI0~AI255	R	0x0000~0x00FF	0~255	
AQ	Analog output register	AQ0~AQ255	R/W	0x0100~0x01FF	256~511	
V	Internal data register	V0~V14847	R/W	0x0200~0x3BFF	512~15359	
TV	Timer(current value)	TV0~TV1023	R/W	0x3C00~0x3FFF	15360~16383	
CV	Counter(current value)	CV0~CV255	R/W	0x4000~0x40FF	16384~16639	16 bit register,among CV4 32 bit register
SV	System special register	SV0~SV900	R/W	0x4400~0x4784	17408~18308	

3. Declare:

1. PLC use the standard Modbus protocol(support RTU and ASCII mode),can communicate to HMI and configuration soft which support Modbus protocol
2. PLC's Modbus addressing number from 0,Some HMI or onfiguration soft from 1,if HMI or onfiguration soft modbus addressing from 0 then couumnicate direct,e.g. M0 is 0x3072,V0 is 4x0512;if HMI or onfiguration soft modbus addressing from 1 then the address must add 1,e.g.M0 is 0x3073[3072+1],V0 is 4x0513[512+1].The first place address is the Modbus protocol component type(0/1 is bit relay ,3/4 is word register , 0/4 can read and write,1/3 read only)other places are the component address.

Error code table

1. System error code table:

Error category	Description
A	Hardware failure, user program not runnable, needs to return to factory repair, red indicator light keeps on
B	Firmware exception or user program exception, user program not runnable, red indicator light will be on 0.5 seconds and be off 0.5 seconds
C	Communication exception between the modules, automatically remove the module with exception, yellow indicator light will be on 0.8 seconds and be off 0.2 seconds

D	Incorrect software setup, allow the user program to continu, yellow indicator light will be on 0.2 seconds and be off 0.8 seconds

Error Code	Message indicated	Error category	Indicator color	Indicator effect
0	System normal			
1	CPU firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
2	CPU memory 1 access exception	A	Red	Keep on
3	CPU memory 2 access exception	A	Red	Keep on
4	RTC access exception	A	Red	Keep on
5	CPU I/O access exception	A	Red	Keep on
6	CPU memory 3 access exception	A	Red	Keep on
7	I/O board access exception	A	Red	Keep on
8	Enhanced bus working abnormally	A	Red	Keep on
59	Slave CPU firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
60	1# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
61	2# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
62	3# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
63	4# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
64	5# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds

Error Code	Message indicated	Error category	Indicator color	Indicator effect
65	6# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
66	7# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
67	8# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
68	9# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
69	10# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
70	11# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
71	12# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
72	13# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
73	14# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
74	15# expand module firmware incomplete	B	Red	On 0.5 seconds and Off 0.5 seconds
75	Expand module hardware failure	B	Red	On 0.5 seconds and Off 0.5 seconds
87	Illegal table content	B	Red	On 0.5 seconds and Off 0.5 seconds
88	Out of program stack space	B	Red	On 0.5 seconds and Off 0.5 seconds
89	Programming software version is too low	B	Red	On 0.5 seconds and Off 0.5 seconds
90	User program corrupted	B	Red	On 0.5 seconds and Off 0.5 seconds
91	Step component exceed range	B	Red	On 0.5 seconds and Off 0.5 seconds

Error Code	Message indicated	Error category	Indicator color	Indicator effect
92	Step combine exceed range	B	Red	On 0.5 seconds and Off 0.5 seconds
93	The table record number is beyond range	B	Red	On 0.5 seconds and Off 0.5 seconds
94	Catch edge times exceed range	B	Red	On 0.5 seconds and Off 0.5 seconds
95	Configuration data is illegal when power supply drop	B	Red	On 0.5 seconds and Off 0.5 seconds
96	Function code illegal	B	Red	On 0.5 seconds and Off 0.5 seconds
97	Operand illegal	B	Red	On 0.5 seconds and Off 0.5 seconds
98	Number of instructions for the same sort out of scope	B	Red	On 0.5 seconds and Off 0.5 seconds
99	No end instruction	B	Red	On 0.5 seconds and Off 0.5 seconds
100	Access 1# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
101	Access 2# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
102	Access 3# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
103	Access 4# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
104	Access 5# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
105	Access 6# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
106	Access 7# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
107	Access 8# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds

Error Code	Message indicated	Error category	Indicator color	Indicator effect
108	Access 9# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
109	Access 10# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
110	Access 11# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
111	Access 12# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
112	Access 13# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
113	Access 14# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
114	Access 15# expand module I/O fails	C	Yellow	On 0.8 seconds and Off 0.2 seconds
131	RTC battery failure	C	Yellow	On 0.8 seconds and Off 0.2 seconds
132	Extension module power supply failure	C	Yellow	On 0.8 seconds and Off 0.2 seconds
133	Storage program and running program inconsistent	C	Yellow	On 0.8 seconds and Off 0.2 seconds
140	Hardware configuration incompatible	D	Yellow	On 0.2 seconds and Off 0.8 seconds
141	Scan timeout watchdog operate	B	Red	On 0.5 seconds and Off 0.5 seconds
142	Have locked datas	D	Yellow	On 0.2 seconds and Off 0.8 seconds
143	Current running step tasks is above upper limit	D	Yellow	On 0.2 seconds and Off 0.8 seconds

2. Communication error code table:

Error code	Declare

Error code	Declare
0	Normal
1	Function Code Error
2	Data Address Error
3	Data Value Error
4	Communication message too short or too long
5	Include not ASCII characters
6	Slave PLC receive message overtime
7	No end character
8	Write data information is too long or too short
9	Check Code Error
10	Application of resources are occupied
11	The firmware does not match with the hardware
12	Program capacity overrun, writing is prohibited

Programming cable wiring diagram

PC(RS232) PLC(COM1)

DB9 female 4 line S male