Servo Motion

Application Note

Modbus TCP comms between AC500eco and E530PTI

AN504 Rev A (Chinese)

Brief introduction

This document takes the communication between ABB AC500 PLC and E530 PTI servo drive through Modbus TCP communication protocol as an example, and describes in detail the process of hardware configuration based on Automation Builder 2.7 and invoking function blocks to write communication programs.
If you need more technical details for consultation and support, please contact your local ABB technical service team.

compatibility

This example is based on

- AC500 PLC firmware version 3.7.0.0 test
- E530PTI Servo drive system firmware version 1.3.0.7 test
- Automation builder V2.7.0.24 test
- Servo composer V1.04.0 tested.

The hardware required to run this example is an ABB AC500 V3 series PLC. There are two main models, AC500 and AC500eco. Because all CPUs can support Modbus TCP, any CPU in this range will do. All AC500 V3 CPUs have a 4-digit type name after the PM part of the code, e.g. PM5072-T-2ETH, unlike the previous generation CPUs that only had 3 digits, e.g. PM564-ETH.

Warranty Liability

Contents

Preparation

Hardware Preparation

AC500 PM5052-T-ETH , one unit F**irmware Version**
Network switch, one
E530PTI servo drive, one unit **Firmware Version**
Debugging a computer, one
CAT5e network cables
220VAC power supply 24VDC power supply.

Software Preparation

Automation Builder 2.7 premium.
Servo Composer 1.4.0。

Knowledge

Qualified knowledge of electrical electricians
Understand the general principles of Modbus TCP.
By default, they have a basic understanding of ABB's PLC hardware and software, and a basic understanding of E530PTI servo drive system.

Modbus TCP principle

Modbus TCP communication protocol is an application-layer packet transmission protocol, the physical layer interface adopts the Ethernet interface, and the device interaction mode is master/slave Manner.
There are four types of Modbus operands

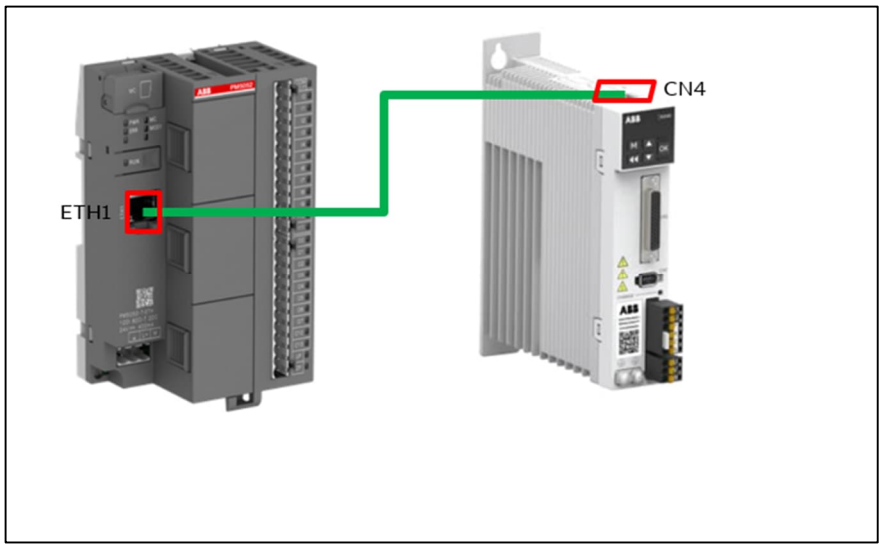
Operands	description
coil	The output bit of the PLC is a switching quantity, which supports read/write
Discrete quantity	The input bit of the PLC is a switching quantity and supports read-only
Enter the registers	Registers in the PLC that can only be operated at the analog input are read-only
Hold registers	A register in the PLC that outputs analog quantities and supports read/write

The following table shows the Modbus operation function codes supported by ABB E530PTI servo drives

Feature codes	name	The type of operational data	Description of the role	Register address
03H	Read hold registers	byte	Gets the current binary value in one or more hold registers	40001-49999
04H	Read the input register	byte	Gets the current binary value in one or more input registers	30001-39999
10H	Write multiple hold registers	byte	Write multiple hold registers and write multiple bytes to the registers	40001-49999
17H	Read/write multiple hold registers	Byte	Read and write the values of multiple hold registers	40001-49999

The following figure shows how the ABB AC500 eCo PLC and the E530PTI servo drive are connected

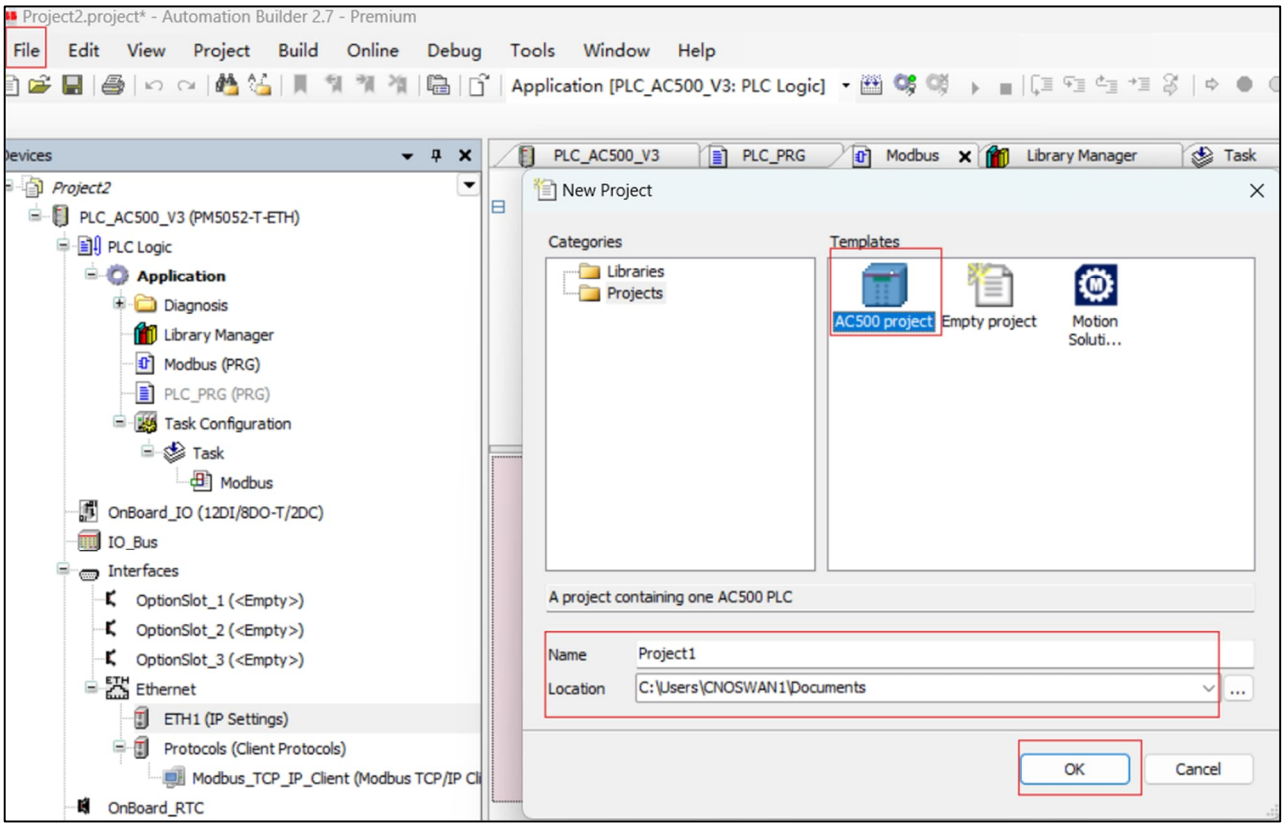
The communication cable should be a standard CAT5e shielded twisted pair cable with RJ45 connectors, in this case the PLC side is connected to the ETH1 interface, and the server driver side is connected to the CN4 interface above the driver.



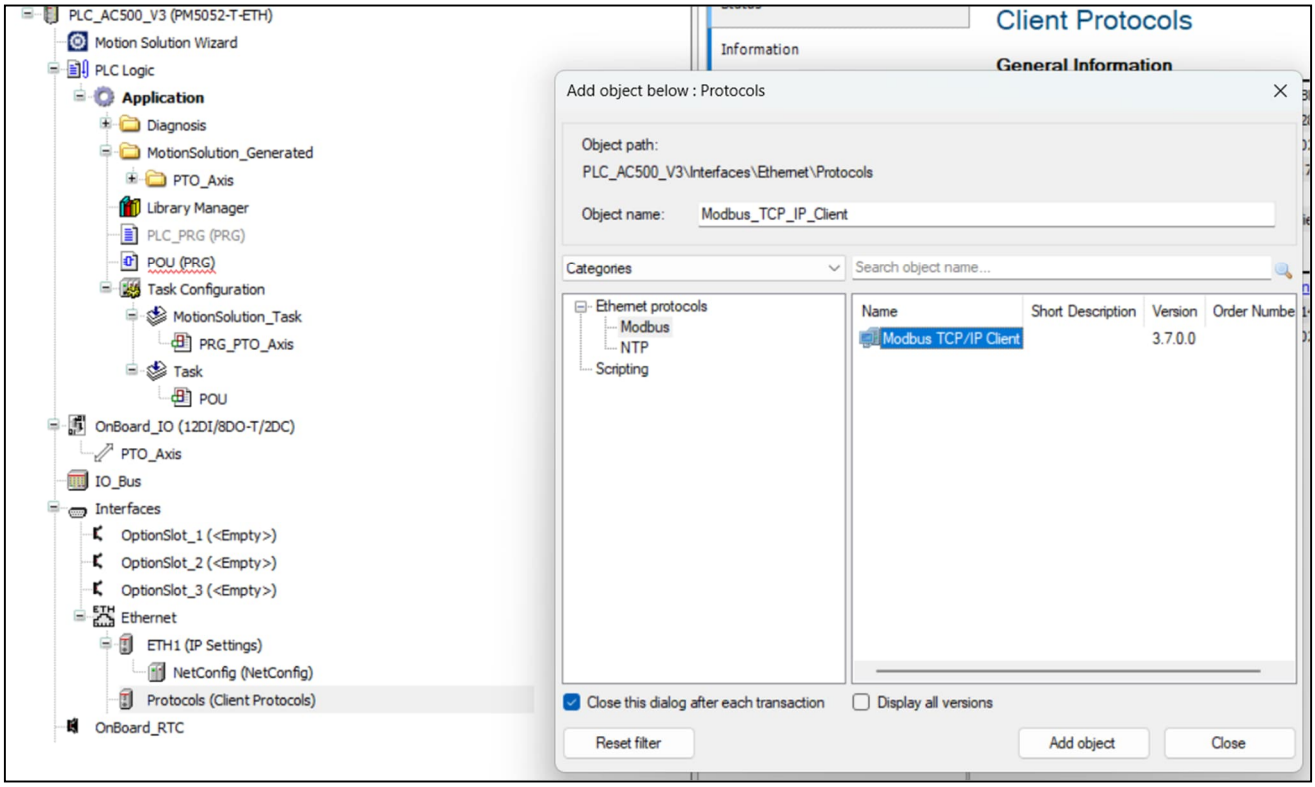
Programming and validation

Hardware configuration

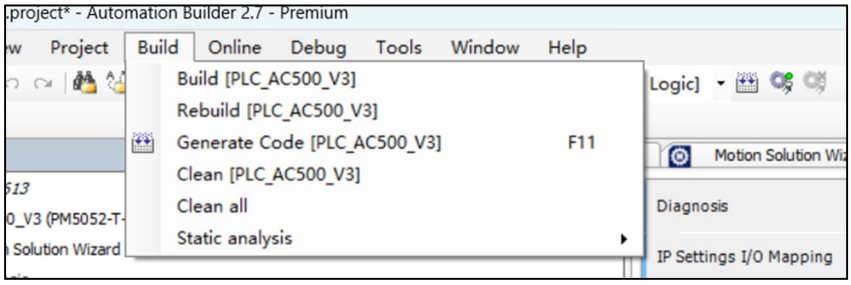
1. Open Automation Builder 2.7, create a new standard project, select the "AC500 project" template, and name it, select the stored project path, and the path name should not contain Chinese characters to prevent unnecessary problems in project operation.



2. Click on the device tree and add "Modbus TCP/IP Client" under "interfaces-ethernet-protocols (Client protocols)".

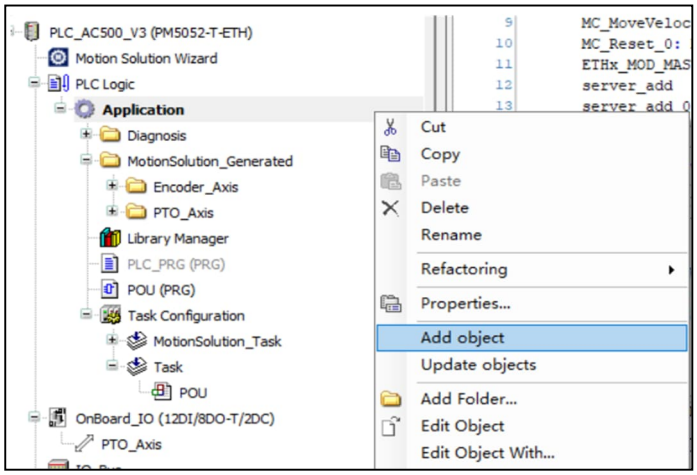


3. In the menu bar, select "build-Rebuild". Once the compilation is correct, it's ready to be programmed.

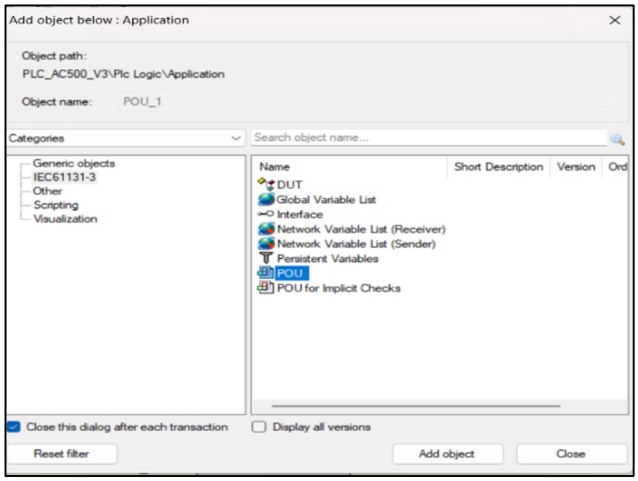


Programming

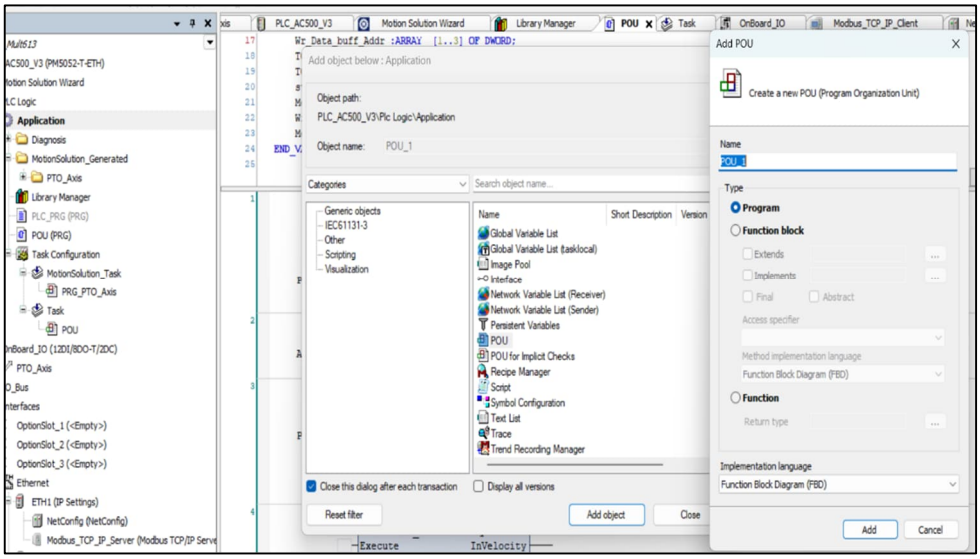
1, In the device tree window, click "PLC Logic->application", right-click and select "Add object".



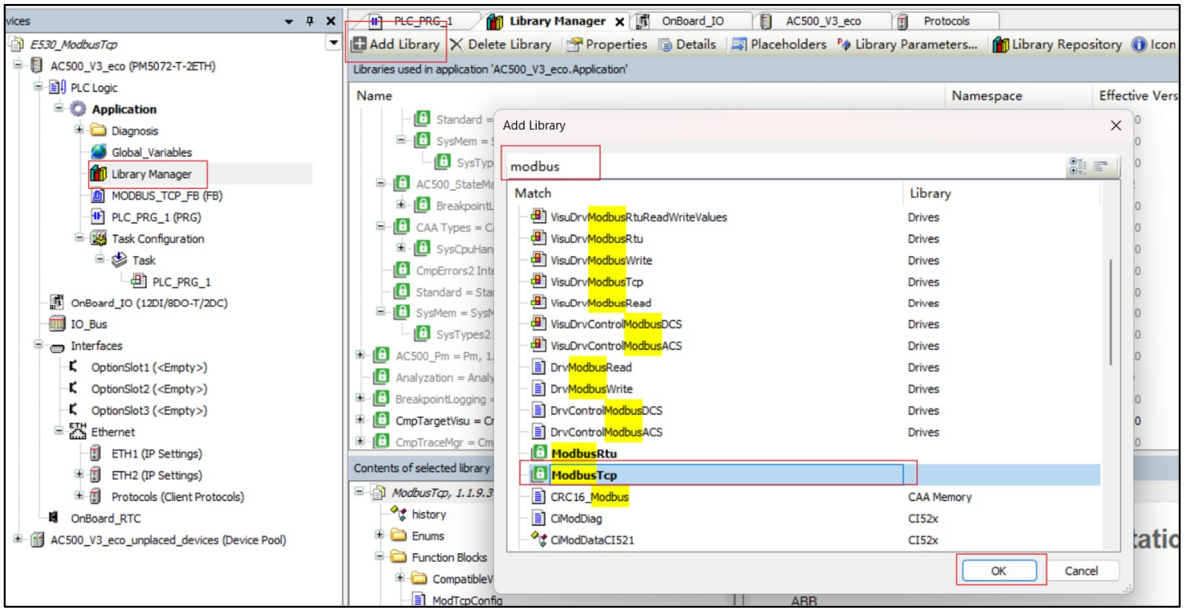
2. Select the POU program organizational unit.



3. Select program, according to personal preference, select the programming language, in this case FBD.

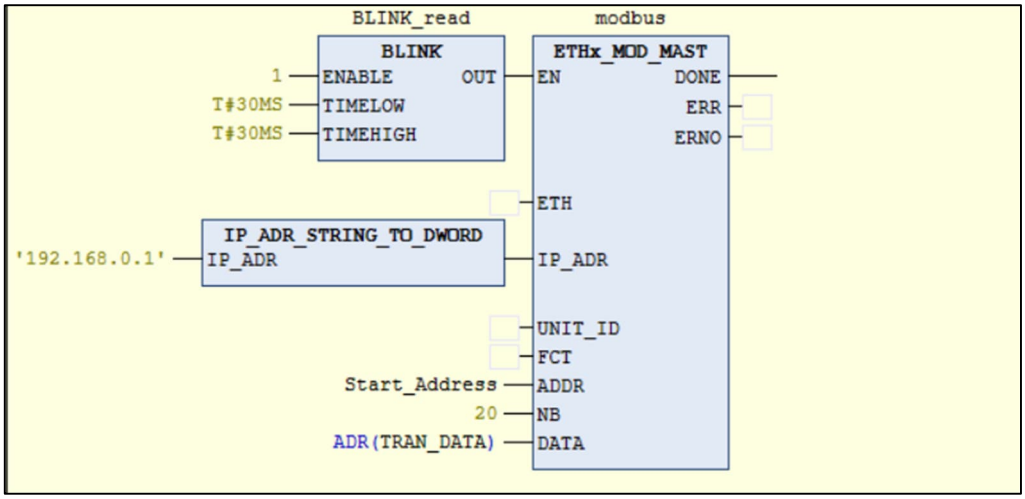


4. To create a new project, you need to call the "Modbus TCP 1.1.9.3 (ABB)" library in the "library manager" manager, and you can proceed to the next step after the loading is completed.



5. At the same you will also need to instigate the ‘Util’ Library for the ‘Blink’ block that you will use in the next program example.

6. Click to open the new POU block, insert it into the network, and call the "ETHx_MOD_MAST" function block, which can realize the communication task of Modbus-TCP.



Pin name	data type	description
EN	Bool	After the input port receives the rising edge signal, it is triggered, and the function block is executed, and after the communication is completed, DONE is set, otherwise the function block will report an error, and check the configuration according to the error code guidelines.
ETH	Byte	Specify the Ethernet interface for communication, the default value is 1, and the default value is generally selected
IP_ADR	Dword	Specify the IP address of the slave communicating (the IP address of the servo drive, mentioned below), the data type, the DWORD double-byte type
UNIT_ID	Byte	Specifies the node number of the slave, which can be ignored
Function Code	Byte	The function code of communication, the function code of commonly used Modbus TCP communication, is shown above.
ADDR	Word	Specifies the register byte start address for read/write operations
NB	Word	The number of bytes of the register address to be read/write from, starting from the start address
DATA	Dword	Pointer to Read / Write data Array

A guide to the useful Function codes is here:

Function Code	WHAT THE FUNCTION DOES	VALUE TYPE	ACCESS TYPE
03 (0x03)	Read Multiple Holding Registers	16 bit	**Read**
06 (0x06)	Write Single Register	16 bit	**Write**
16 (0x10)	Write Multiple Registers	16 bit	**Write**

At this point, on the PLC side, a simple Modbus TCP communication program is established, compiled and downloaded to the PLC, and ready to read and write data.

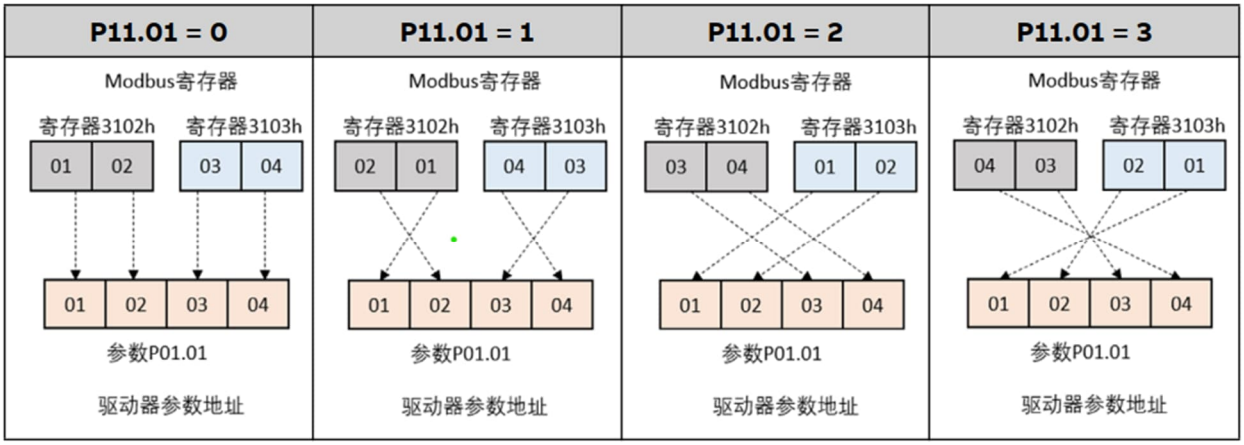
Servo drive-side configuration

Next, the configuration is done on the E530PTI servo drive side via the Servo Composer servo commissioning software tool.

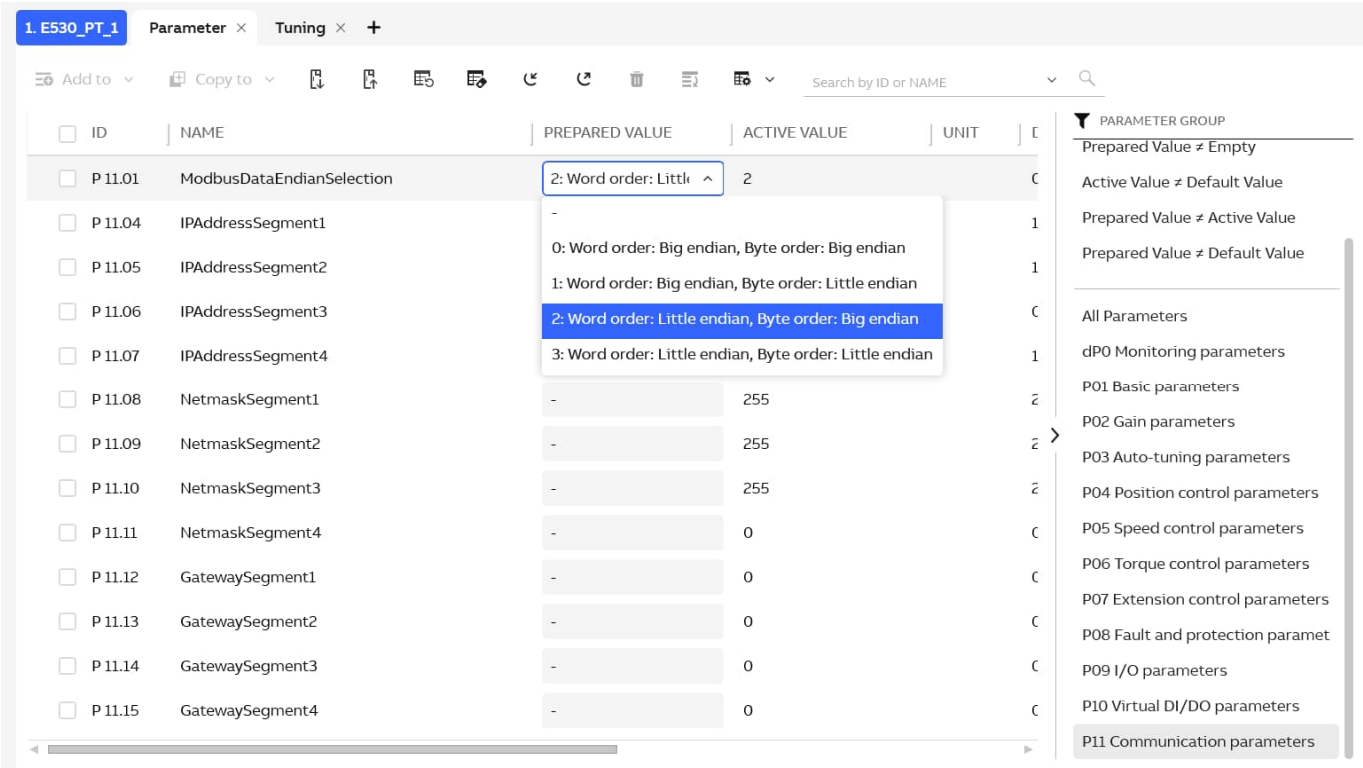
1. Open Servo Composer, search online for the drive you are communicating with, check it, and click "Create".

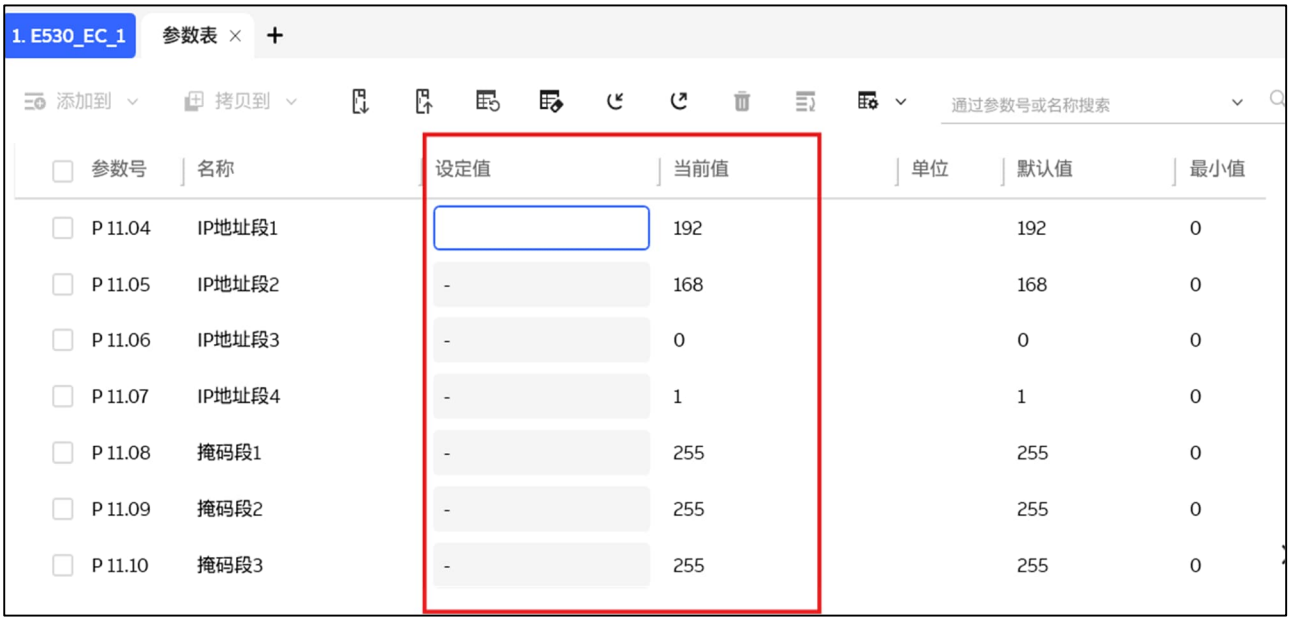
2. Click on the drive inside the red box to open the parameter table.

3. Select the P11 communication parameters, the default AC500 PLC is the byte big-end order, so set the P11.01--Modbus data big-end selection parameters, set to 2 word order small-Indian, byte order big-end. The following diagram illustrates the correspondence between Modbus registers and driver parameters in different size-side configuration modes



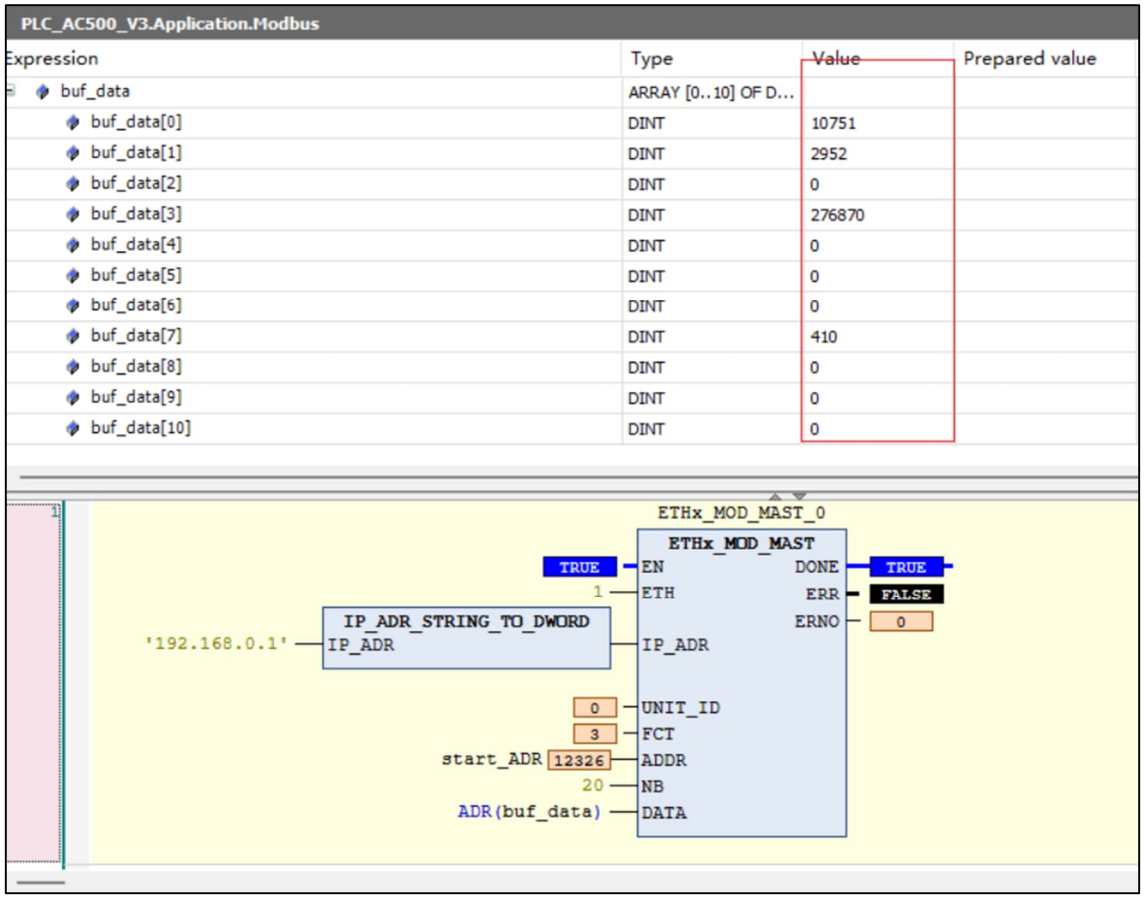
Select P11.04-P11.07 and set the IP address to ensure that the address of the servo drive and the address of the PLC are in the same network segment and do not conflict. Save, power off and restart, and the settings take effect.

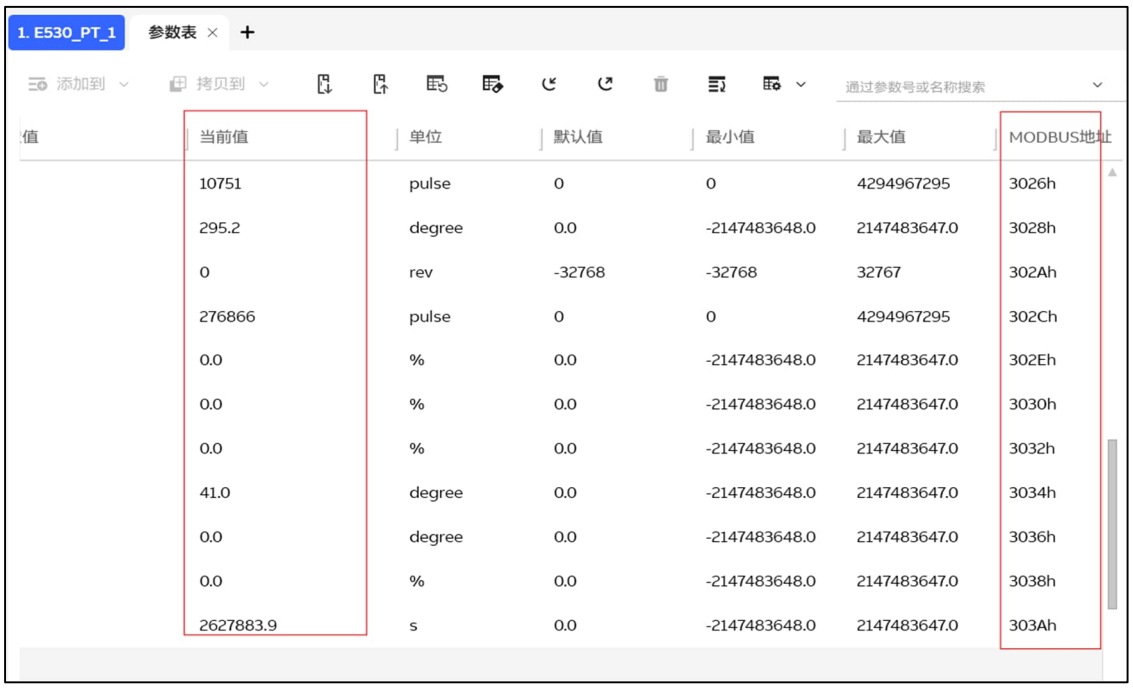




Read and writing and Scaling

1. Take 3026h as the starting address of the read, its decimal numeric address is 12326, and write it to the input pin of the function block "ADDR", because the parameters on the driver side of the E530 PTI server occupy the register address in the form of double bytes, so in the function block "ETH-Modbus TCP Master" on the PLC side, the number of bytes NB of the read or written address should be twice that of the read register, so ten parameters are read, The number of bytes that need to be read by a function block is NB 20 And trigger the function block in the PLC program, by comparing the data of PLC online monitoring, the data is consistent, so far, it means that the communication is successful.





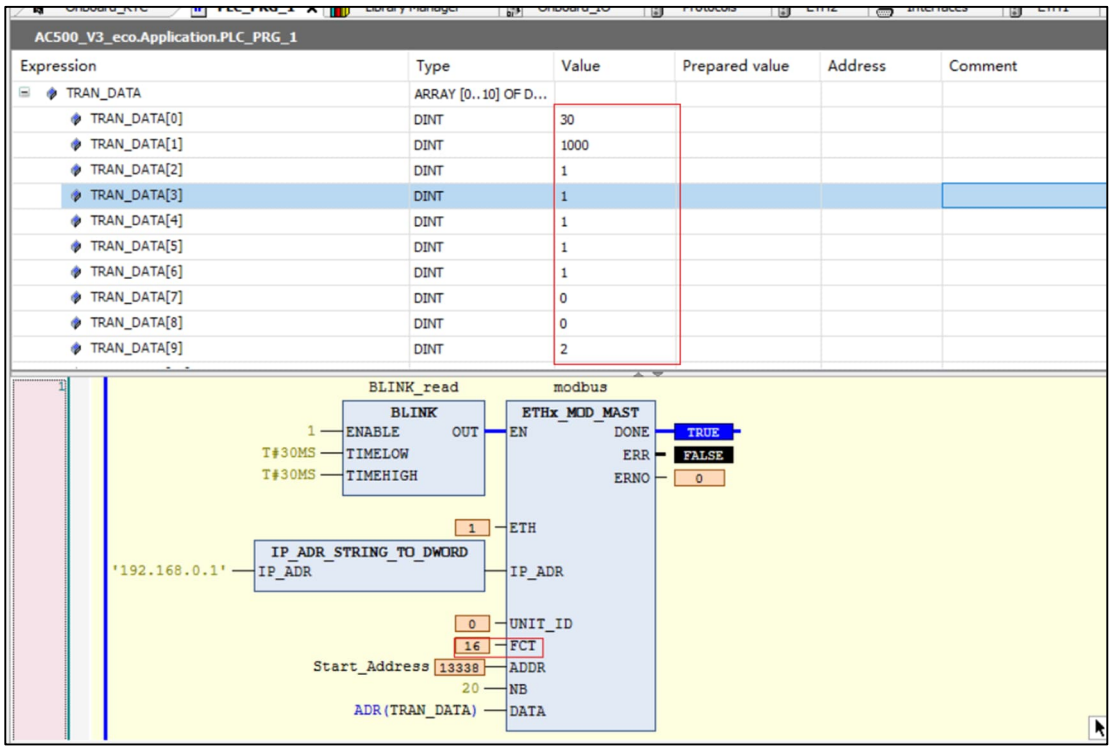
值	当前值	单位	默认值	最小值	最大值	MODBUS地址
	10751	pulse	0	0	4294967295	3026h
	295.2	degree	0.0	-2147483648.0	2147483647.0	3028h
	0	rev	-32768	-32768	32767	302Ah
	276866	pulse	0	0	4294967295	302Ch
	0.0	%	0.0	-2147483648.0	2147483647.0	302Eh
	0.0	%	0.0	-2147483648.0	2147483647.0	3030h
	0.0	%	0.0	-2147483648.0	2147483647.0	3032h
	41.0	degree	0.0	-2147483648.0	2147483647.0	3034h
	0.0	degree	0.0	-2147483648.0	2147483647.0	3036h
	0.0	%	0.0	-2147483648.0	2147483647.0	3038h
	2627883.9	s	0.0	-2147483648.0	2147483647.0	303Ah

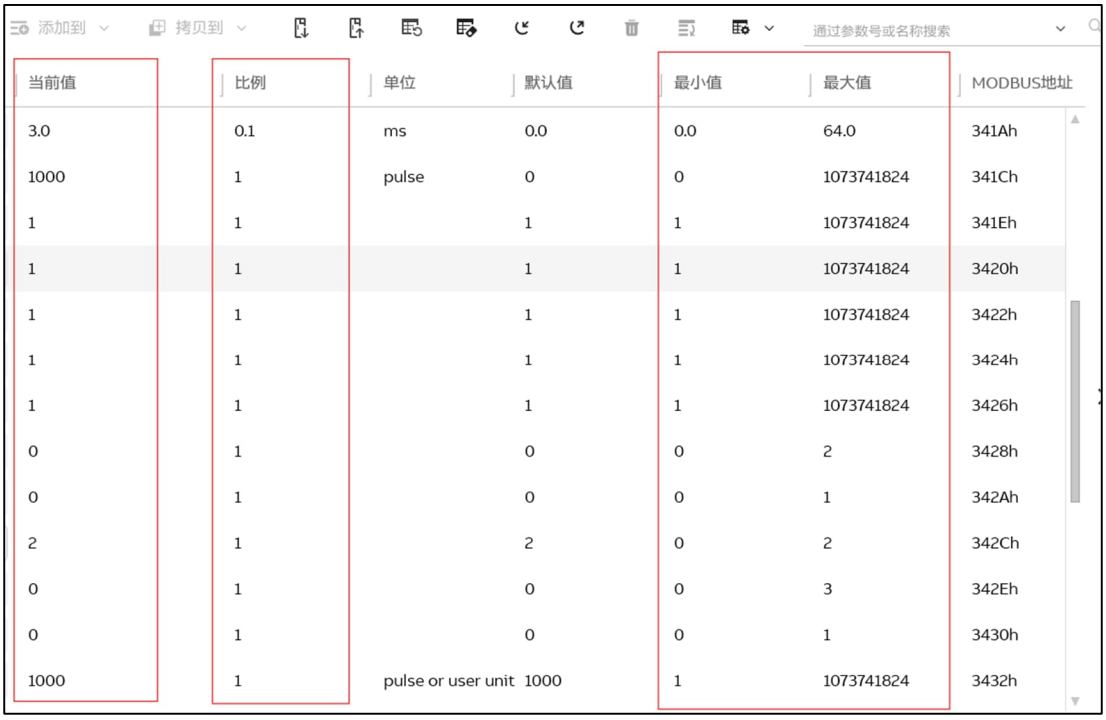
2. It should be noted that the data monitored by Servo Composer online, for example, 3028h, has a current value of 295.2, while the data read by the PLC side is 2952, because the current value read by the Servo Composer side is scaled according to the scale of 0.1, which is convenient for Modbus TCP communication to read and store at all times.
The value is hidden by default and can be checked in the menu bar of the Servo Composer software, which will display the zoom of the corresponding parameter. Therefore, on the PLC side, the value of the corresponding register read is converted according to the zoom dimension. For example, the actual value of 3028h should be 2952*0.1=295.2, and the actual value of 3026h should be 10751*1=10751.

当前值	比例	单位	默认值	最小值	最大值
0.55	0.01	volt		-2147483648.00	2147483647.00
0.00	0.01	amper		-2147483648.00	2147483647.00
10751	1	pulse		0	4294967295
295.2	0.1	degree		-2147483648.0	2147483647.0
0	1	rev		-32768	32767
276872	1	pulse		0	4294967295
0.0	0.1	%	0.0	-2147483648.0	2147483647.0
0.0	0.1	%	0.0	-2147483648.0	2147483647.0
0.0	0.1	%	0.0	-2147483648.0	2147483647.0
41.2	0.1	degree	0.0	-2147483648.0	2147483647.0

3. The real-time value of the "buf_data" array in the above two pictures is consistent with the real-time value of the address starting from the Modbus address "3025h" corresponding to the servo drive monitored by the Servo Composer online tool. Indicates that the ModbusTCP communication is successful.

4. At the same time, for the write operation, as long as the function code "FCT" of the function block is modified to "10h", the corresponding decimal value is 16, take the starting write address 341Ah as an example, the corresponding decimal value is 13338, and write the corresponding "FCT" pin.
Write the prepared value of the written array to the TRAN_DATA, and after triggering the function block again, you can write the prepared value to the corresponding register address of the servo drive.





5. At the same time, it should be noted that the written data, pay attention to the value range and scale of the corresponding value, otherwise, in the writing process, the function block will report an error.

Debugging

During debugging you may find there are some errors given by the ETHx_MOD_TCP function block. If any are seen for quick reference below is a list of possible error codes that could be generated (this can also be found by the integrated help in the AC500_ModbusTCP library entry).

Enumeration	Hex Value	Dec Value	Comment

NO_ERROR	16#0	0	Execution successfully completed
ERR_NOT_CONFIGURED	16#1	1	Modbus TCP not configured
ERR_INVALID_INTERFACE	16#2	2	Invalid Ethernet interface or no Server configured on Ethernet interface
ERR_ENDIANESS_NOT _SUPPORTED	16#3	3	Endianness / byte order not supported
ERR_FCT_NOT_SUPPORTED	16#4	4	Function code not supported
ERR_ILLEGAL_REGISTER _ADDRESS	16#5	5	Illegal register address
ERR_DATA_SIZE	16#6	6	Number of data to read/write exceeds capabilities
ERR_INVALID_ACCESS	16#7	7	Invalid address of local storage of data to read/write. When using FCT22/FCT23, please use struct ETH_MOD_FC22_TYPE or ETH_MOD_FC23_TYPE.
ERR_INVALID_TIMEOUT_VALUE	16#8	8	Invalid value for response timeout
ERR_ALREADY_IN_STATE	16#10	16	Server already in requested state
ERR_FAILED_SET_STATE	16#11	17	Failed to set Server to requested state
ERR_SIMULTANEOUS_ACCESS	16#12	18	Server already accessed by other FB instance, try again later on
ERR_FAILED_CONNECT	16#100	256	Failed to connect to Server
ERR_EXCEPT_01_ILLEGAL _FUNCTION	16#101	257	Illegal function exception. Exception response by Server containing error code 01 / 16#01
ERR_EXCEPT_02_ILLEGAL _DATA_ADDRESS	16#102	258	Illegal data address. Exception response by Server containing error code 02 / 16#02
ERR_EXCEPT_03_ILLEGAL _DATA_VALUE	16#103	259	Illegal data value. Exception response by Server containing error code 03 / 16#03
ERR_EXCEPT_04_SERVER _DEVICE_FAILURE	16#104	260	Server device failure. Exception response by Server containing error code 04 / 16#04
ERR_EXCEPT_05_SERVER _ACKNOWLEDGE	16#105	261	Server acknowledge. Exception response by Server containing error code 05 / 16#05
ERR_EXCEPT_06_SERVER _DEVICE_BUSY	16#106	262	Server device busy. Exception response by Server containing error code 06 / 16#06
ERR_EXCEPT_07_SERVER _NO_ACKNOWLEDGE	16#107	263	Server no acknowledge. Exception response by Server containing error code 07 / 16#07
ERR_EXCEPT_08_MEMORY _PARITY_ERROR	16#108	264	Memory parity error. Exception response by Server containing error code 08 / 16#08
ERR_EXCEPT_09_SERVER _PASSIVE	16#109	265	Probably passive Server. Exception response by Server containing error code 09 / 16#09
ERR_EXCEPT_0A_GATEWAY _PATH_UNAVAILABLE	16#10A	266	Gateway path unavailable. Exception response by Server containing error code 10 / 16#0A
ERR_EXCEPT_0B_GATEWAY _TARGET_DEVICE	16#10B	267	Gateway target device failed to respond. Exception response by Server containing error code 11 / 16#0B
ERR_TIMEOUT	16#120	288	Timeout expired. Server did not respond within specified time
ERR_CONNECTION_CLOSED	16#121	289	Connection closed or unexpected result reported by stack
BUSY	16#FFF	4095	Busy, call again
FATAL_ERROR	16#5FFF	4095	Fatal error state machine
ERR_NO_MEMORY	16#8001	32769	Not enough memory
ERR_INTERNAL_IO	16#8002	32770	Internal error in I/O layer
ERR_INTERNAL_INVALID_STATE	16#8003	32771	Invalid state, fatal internal error
ERR_INTERNAL_ILLEGAL_STATE	16#8004	32772	Protocol stack in illegal state. Fatal internal error
ERR_INTERNAL_UNEXPECTED_STATE	16#8005	32773	Unexpected state. Fatal internal error.

ERR_INTERNAL_IO_RETRY	16#8006	32774	Retry I/O operation. Operation not yet completed, function to be called again
ERR_INTERNAL_ILLEGAL_ARGUMENT	16#8007	32775	Stack reports illegal argument (internal error)
ERR_INTERNAL_INVALID_HANDLE	16#8008	32776	Illegal argument, unknown instance ‘dwHandle’. Internal error.
ERR_INTERNAL_PORTING	16#8009	32777	Porting layer error. Operating system signaled error
ERR_UNKNOWN	16#800A	32778	Undefined error

Contact us

For more information, please contact your

A local ABB representative or one of the following:

new.abb.com/drives/low-voltage-ac/servo-products

new.abb.com/drives

new.abb.com/drivespartners

new.abb.com/PLC