EN7000 EIP Communication Configuration

Application Note 700239

1. Introduction

The EN7000 timer offers Modbus communication through Ethernet or RS-485 serial port. When the EN7000 timer needs to communicate with a PLC or any EtherNetIP (EIP) device, an HMS Anybus Communicator (ABC) will be used.

The ABC is a gateway between the EN7000 and the EIP network. It accesses the data in the EN7000 via MODBUS-RTU and presents it on the EIP side. The ABC is a MODBUS-RTU master to the EN7000 and Group 2/3 server on the EIP network.

Figure 1 shows the EN7000 using an IP address of 192.168.0.101 for Netflash programming via Ethernet TCP/IP and the ABC using an IP address of 192.168.0.113 for EtherNet/IP.

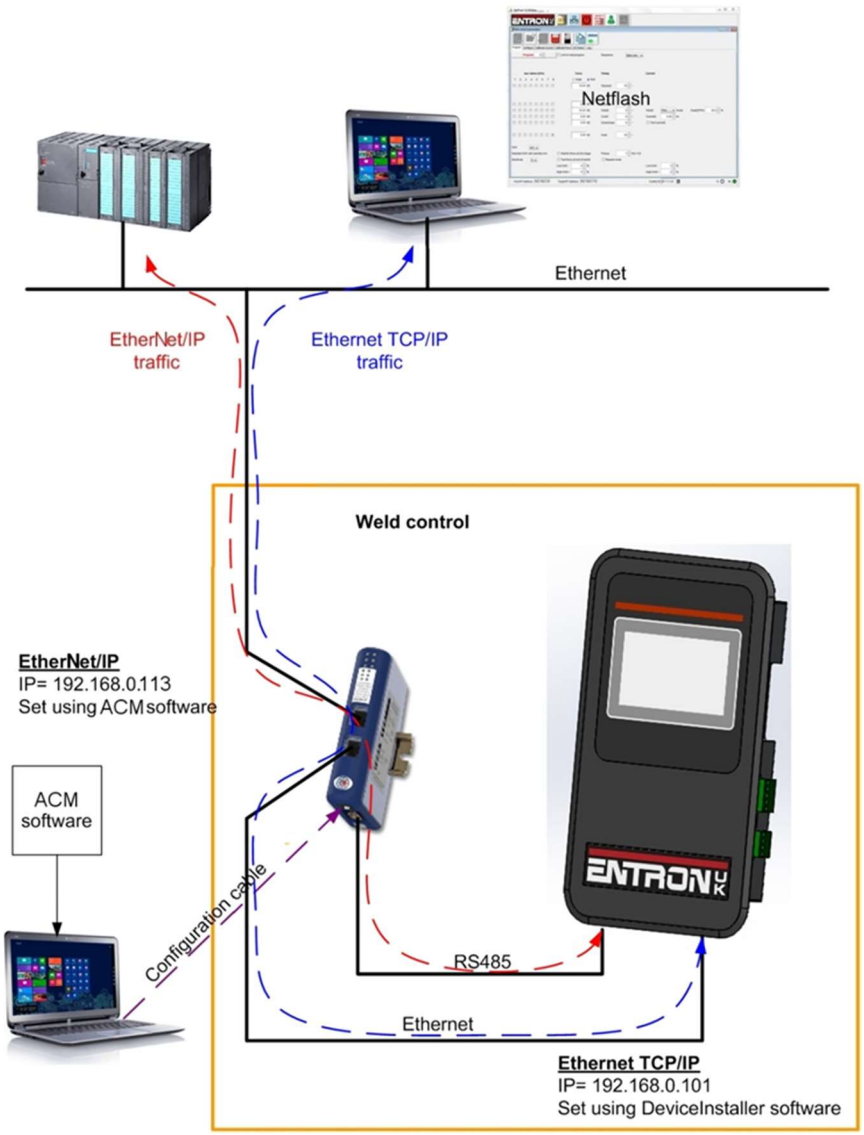


Figure 1 Generic system connection

2. Hardware, Software and Documentation

The following equipment and documentation are required:

- EN7000
- HMS ABC EIP/MODBUS-RTU module AB7072-B (ENTRON P/N: 318061)
- ABC Installation Sheet HMS- SP1708
- ABC User Manual HMSI-27-316
- +24 V dc power supply
- RS-485 MODBUS-RTU cable (EN7000-ABC) (ENTRON P/N: 326084)
- Ethernet cable (ABC-network)
- Configuration cable (ABC-PC)
- HMS Anybus Configuration Manager (ACM) software + EN7000 configuration file
- PLC
- EN7000 Technical Manual

The ABC is pre-configured at ENTRON to provide essential functionality. If the IP address of the ABC module or the other parameters need to be changed, the ACM software will be used.

The ACM software should be installed on a suitable PC as described in section 1.4 of the ABC User Manual. The minimum requirements are as follows:

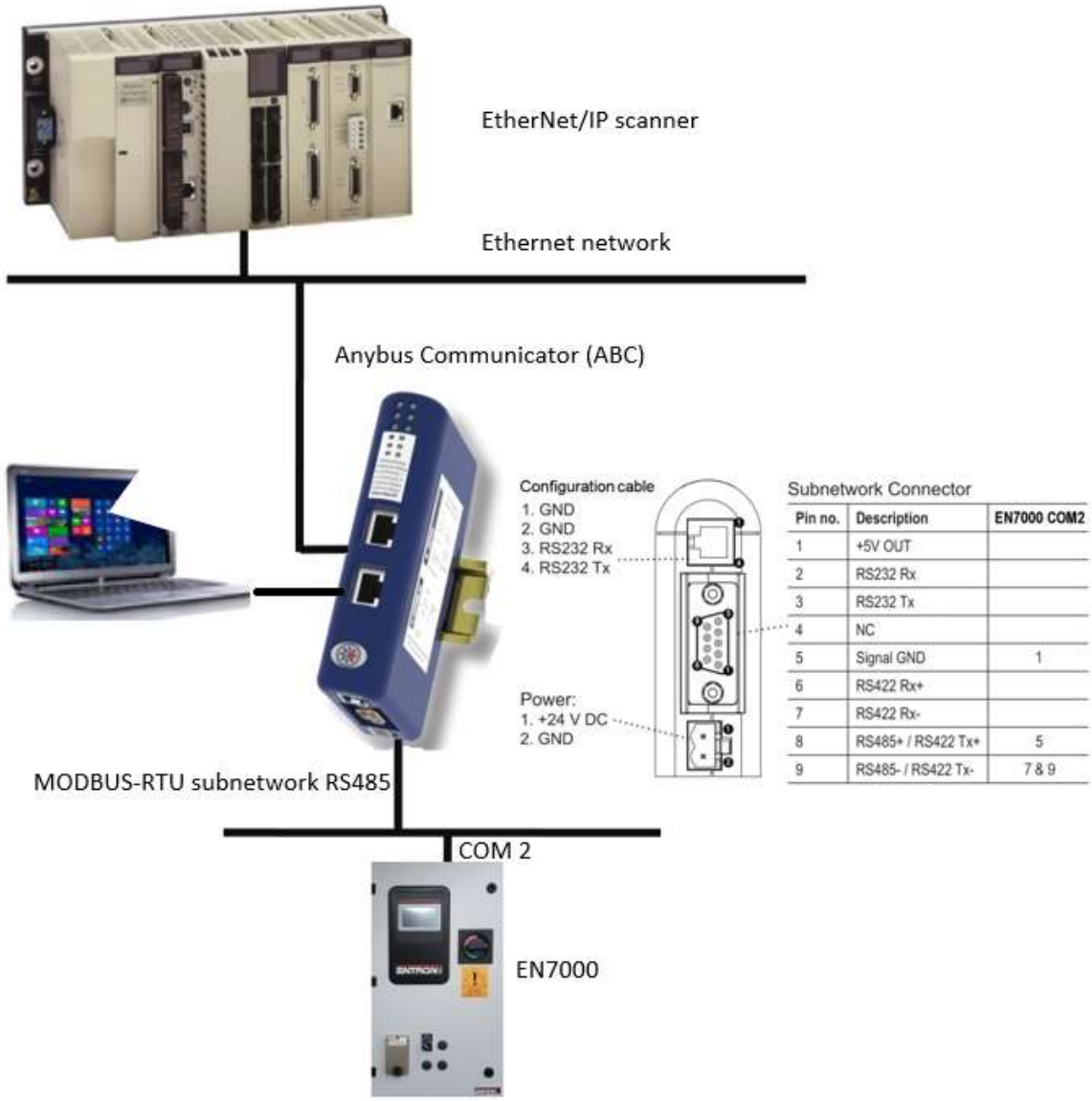
- Pentium 133 MHz or higher
- 650 MB of free space on the hard drive
- 32 MB RAM
- Screen resolution 800 x 600 (16 bit color) or higher
- Microsoft Windows® 2000 / XP / Vista / 7 (32- or 64-bit)
- Internet Explorer 4.01 SP1 or newer (or any equivalent browser)

The ABC Installation Sheet, User Manual and information are available from
https://www.anybus.com/support/file-doc-downloads/communicator-specific/?orderCode=ab7072

3. Hardware Connection and EN7000 Setting

Follow the steps below to connect the equipment as Figure 2 shows:

- Power off all equipment
- Connect the ABC to the Ethernet network.
- Connect the ABC to the COM2 port of the EN7000 via the MODBUS-RTU subnetwork.
- Connect the configuration cable between the ABC and the PC containing the ACM software. (The second Ethernet port on the ABC allows it to be used as a hub if required.)
- Power on all equipment
- Set EN7000 I/O source to COM2
- Set EN7000 COM2 parameters to MODBUS-RTU slave/address 1/57600 baud
- Restart the EN7000



Subnetwork Connector

Pin no.	Description	EN7000 COM2
1	+5V OUT	
2	RS232 Rx	
3	RS232 Tx	
4	NC	
5	Signal GND	1
6	RS422 Rx+	
7	RS422 Rx-	
8	RS485+ / RS422 Tx+	5
9	RS485- / RS422 Tx-	7 & 9

Figure 2 Hardware Connection

4. IP address setting

The ABC will be shipped from ENTRON with the following default settings:

IP address	192.168.0.113
Default gateway	192.168.0.251
Subnet mask	255.255.255.0

If the default values are incompatible with your Ethernet network, ACM can be used to change them.

- Open the EN7000 configuration file in ACM:

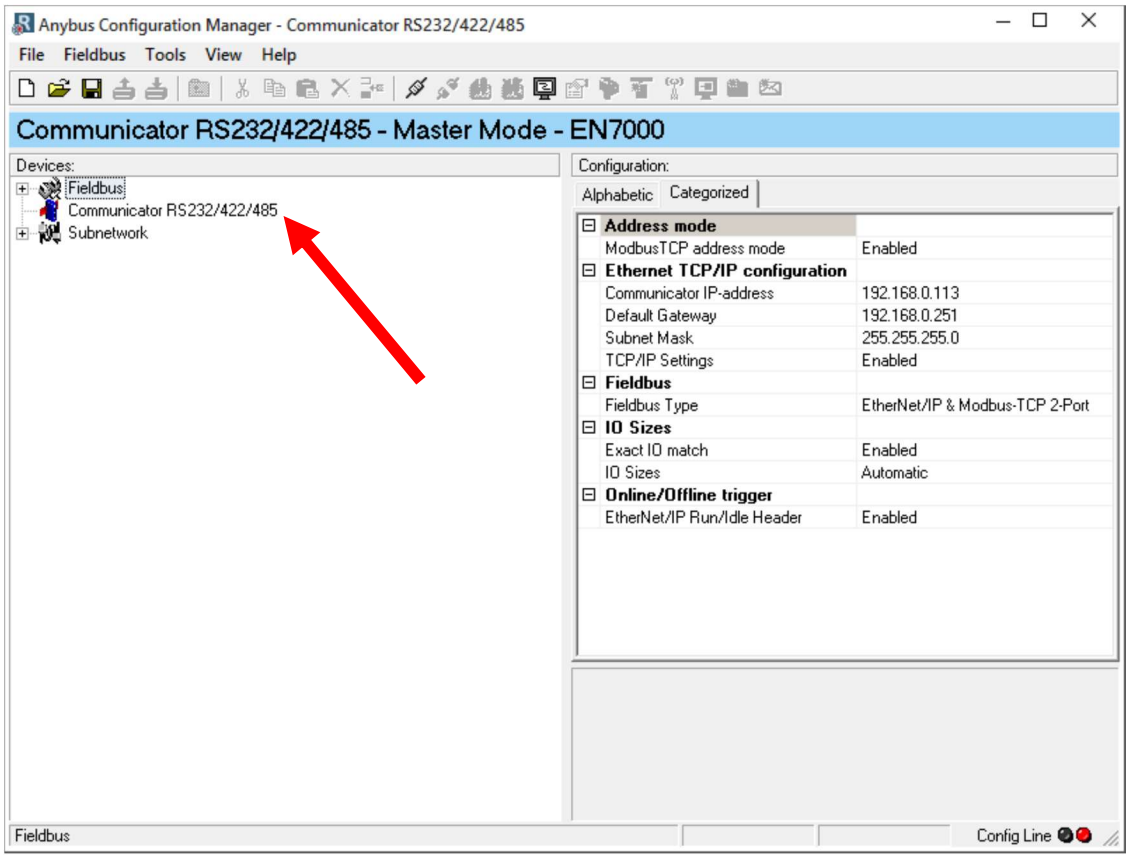


Figure 3 Configuration file in ACM

- Right-click on the Communicator node and select Connect
- Select the Fieldbus node

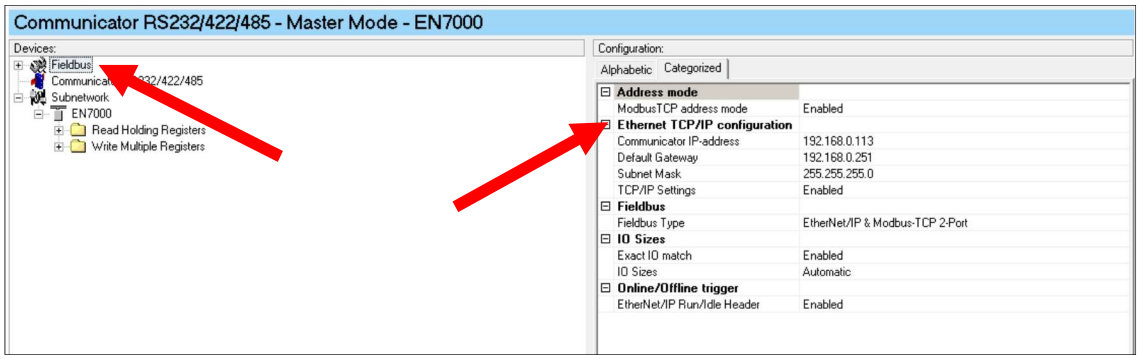
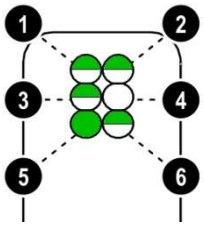


Figure 4 Select the Fieldbus node

- Enter new values for the IP address, gateway and subnet mask if required
- Select Tools->Download configuration. When the download is complete the LED indicators should be as follows:

LED1 – flashing green
LED2 – flashing green
LED3 – flashing green
LED4 – flashing green (if the ABC is being used as a hub)
LED5 – green
LED6 – flashing green



Consult the ABC User Manual if the indicators are different.

Alternatively, the ABC has a built-in web server that can be used to change the configuration. Use your preferred internet browser and enter http://192.168.0.113 in the address bar. The ABC will show the configuration page and the TCP/IP settings can be changed:

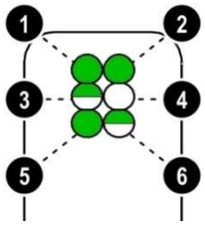
5. Principle of EIP Communication

The ABC is controlled from the EIP side via the Assembly Object (class 04h). The ABC requires a cyclic Class 1 connection to access the Assembly Object. The functions are shown below:

Function	Instance	Attribute	Connection	Data size
Write to EN7000	96h (150)	3	O->T	138
Read from EN7000	64h (100)	3	T->O	178

When the Class 1 connection is established, the LED indicators should be as follows:

LED1 – green
LED2 – green
LED3 – flashing green
LED4 – flashing green (if the ABC is being used as a hub)
LED5 – green
LED6 – flashing green



Consult the ABC User Manual if the indicators are different.

The ABC uses cyclic and triggered transactions as described below. It is configured to cyclically access the EN7000 I/O data. Parameter data transactions must be triggered as needed. The Assembly Object is used to transfer data and to control the transactions.

When the ABC is added to a PLC project, the controller tag ABC.O and ABC.I will be created to carry the Original to Target data and Target to Original data.

ABC.O includes 138 tags, it carries the data which will be sent to EN7000 control. The memory map is as follows:

ABC.O (Originator to Target data) (138 bytes)

Byte address	Function	Comment
0 – 3	Inputs to EN7000	see EN7000 Technical Manual
4	not used	
5	Read trigger	change value to trigger Read
6	not used	
7	Write trigger	change value to trigger Write
8	Start address (high byte)	address to read from/write to
9	Start address (low byte)	address to read from/write to
10 - 137	Parameters to EN7000 (little endian)	see EN7000 Technical Manual

ABC.I includes 178 tags, it carries the data which will be sent from EN7000 control. The memory map is as follows:

ABC.I (Target to Originator data) (178 bytes)

Byte address	**Function**	**Comment**
0 – 47	Outputs & Status from EN7000	see EN7000 Technical Manual
48 – 175	Parameters from EN7000 (little endian)	see EN7000 Technical Manual
176	Start address used in Write Query (high byte)	address written to
177	Start address used in Write Query (low byte)	address written to

I/O Data Access

When the PLC is in Run mode and the Class 1 connection is established, the I/O transaction starts automatically and is updated every 100 milliseconds.

The EN7000 inputs are stored in the tag ABC:O.Data[0]---[3]. Consult the MODBUS mapping (inputs to EN7000) Table in Section 5 of the EN7000 Technical Manual for details of the input data encoding. For example to activate the Weld On input, set Bit ABC:O.Data[0].1. If the Class 1 connection becomes inactive, the inputs to the EN7000 are cleared.

The state of the EN7000 outputs is returned in the tag ABC:I.Data[0]—[47]. Consult the MODBUS mapping (outputs from EN7000) Table in Section 5 of the EN7000 Technical Manual for details of the output data encoding.

Parameter Data Access

The EN7000 parameters include the weld programs, electrode programs, calibration information, and configuration settings. Consult the EN7000 Technical Manual for details of the registers that are used to store these parameters. Unlike the I/O data, the parameter data is accessed only when required using triggered transactions. A triggered transaction to write or read requires the following steps:

To trigger a transaction and **Write** parameters to EN7000, the following steps are required:

1) Set the Start address in the tag ABC:O.Data[8] and ABC:O.Data[9]

   The Start address is the Modbus start address of the parameters in the EN7000. See below the Modbus Mapping Table in Section 13 of the EN7000 Technical Manual for the Start address values for different type of parameters.

Variable	Address	Type	Description
Weld programs			256 x 64 WORDS
Weld program 0	16#0000 (= 0)	WORD ARRAY [0..63]	
Weld program 1	16#0040 (= 64)	WORD ARRAY [0..63]	
Weld program 2	16#0080 (= 128)	WORD ARRAY [0..63]	
Weld program 3	16#00C0 (= 192)	WORD ARRAY [0..63]	
...	...	...	
Weld program 254	16#3F80 (= 16256)	WORD ARRAY [0..63]	
Weld program 255	16#3FC0 (= 16320)	WORD ARRAY [0..63]	
Electrodes			8 x 64 WORDS
Electrode 0	16#4000 (= 16384)	WORD ARRAY [0..63]	
Electrode 1	16#4040 (= 16448)	WORD ARRAY [0..63]	
Electrode 2	16#4080 (= 16512)	WORD ARRAY [0..63]	
Electrode 3	16#40C0 (= 16576)	WORD ARRAY [0..63]	
Electrode 4	16#4100 (= 16640)	WORD ARRAY [0..63]	
Electrode 5	16#4140 (= 16704)	WORD ARRAY [0..63]	
Electrode 6	16#4180 (= 16768)	WORD ARRAY [0..63]	
Electrode 7	16#41C0 (= 16834)	WORD ARRAY [0..63]	
Calibration			8 x 64 WORDS
Calibration 0	16#5000 (= 20480)	WORD ARRAY [0..63]	
Calibration 1	16#5040 (= 20544)	WORD ARRAY [0..63]	
Calibration 2	16#5080 (= 20608)	WORD ARRAY [0..63]	
Calibration 3	16#50C0 (= 20672)	WORD ARRAY [0..63]	
Calibration 4	16#5100 (= 20736)	WORD ARRAY [0..63]	
Calibration 5	16#5140 (= 20800)	WORD ARRAY [0..63]	
Calibration 6	16#5180 (= 20864)	WORD ARRAY [0..63]	
Calibration 7	16#51C0 (= 20928)	WORD ARRAY [0..63]	
Configuration			1 x 64 WORDS
Configuration	16#6000 (= 24576)	WORD ARRAY [0..63]	

Table 1 MODBUS Start address (MODBUS mapping)

ABC:O.Data[8] should store the high byte of the Start address and ABC:O.Data[9] should store the low byte of the Start address.

For example:

- If Configuration's parameters will be written to EN7000, the Start address will be 16#6000,

  ABC:O.Data[8]=16#60

  ABC:O.Data[9]=16#00

- If Weld program 2's parameters will be written to EN7000, the Start address will be 16#0080,

  ABC:O.Data[8]=16#00

  ABC:O.Data[9]=16#80

- If Electrode 7's parameters will be written to EN7000, the Start address will be 16#41C0,

  ABC:O.Data[8]=16#41

  ABC:O.Data[9]=16#C0

2) Set the parameters in ABC:O.Data[10] through ABC:O.Data[137]

   All the parameters should be copied to ABC:O.Data[10]--[137] before a transaction is executed.

   For example, if Configuration's parameters will be written to EN7000, 26-word (52-byte) Configuration parameters will be copied to ABC:O.Data[10]--[61] in Little-endian format.

3) Change the value of the trigger byte in ABC:O.Data[7]

   ABC:O.Data[7] is the writing trigger byte for the ABC. The transaction begins when the trigger byte value changes; the actual value is not significant.

To trigger a transaction and **READ** parameters from EN7000, the following steps are required:

1) Set the Start address in ABC:O.Data[8] and ABC:O.Data[9]

   The same as step 1) of above writing transaction.

2) Change the value of the trigger byte in ABC:O.Data[5]

   ABC:O.Data[5] is the reading trigger byte for ABC. The transaction begins when the trigger byte value changes; the actual value is not significant.

3) Implement a delay on PLC and wait for the parameters

   A recommend delay in the PLC is 200 mS so that ABC has enough time to receive parameters from the EN7000 and send them to the PLC.

4) Read parameter from ABC:I.Data[10]---[137]

6. PLC Project Examples

6.1 Example 1: Exchange control's Configuration data

As an example, an Allen Bradley CompactLogix PLC will be used to exchange the control's Configuration data with the EN7000. When the PLC's Input00 is closed, the PLC will write the Configuration data to the EN7000 timer; when the PLC's Input01 is closed, the PLC will read the Configuration data from the EN7000 timer.

The example only addresses steps required to create an EIP message. It is not intended to cover all aspects of programming a PLC.

6.1.1 Adding the ABC to the PLC project

Prior to programming any message instructions, the ABC must first be added to the PLC project Ethernet network. Follow the below steps to add the ABC:

a. Set Studio 5000 in **Offline** mode.
b. Right-click on the EtherNetIP gateway in the I/O configuration and select **New Module**, as shown in Figure 5.

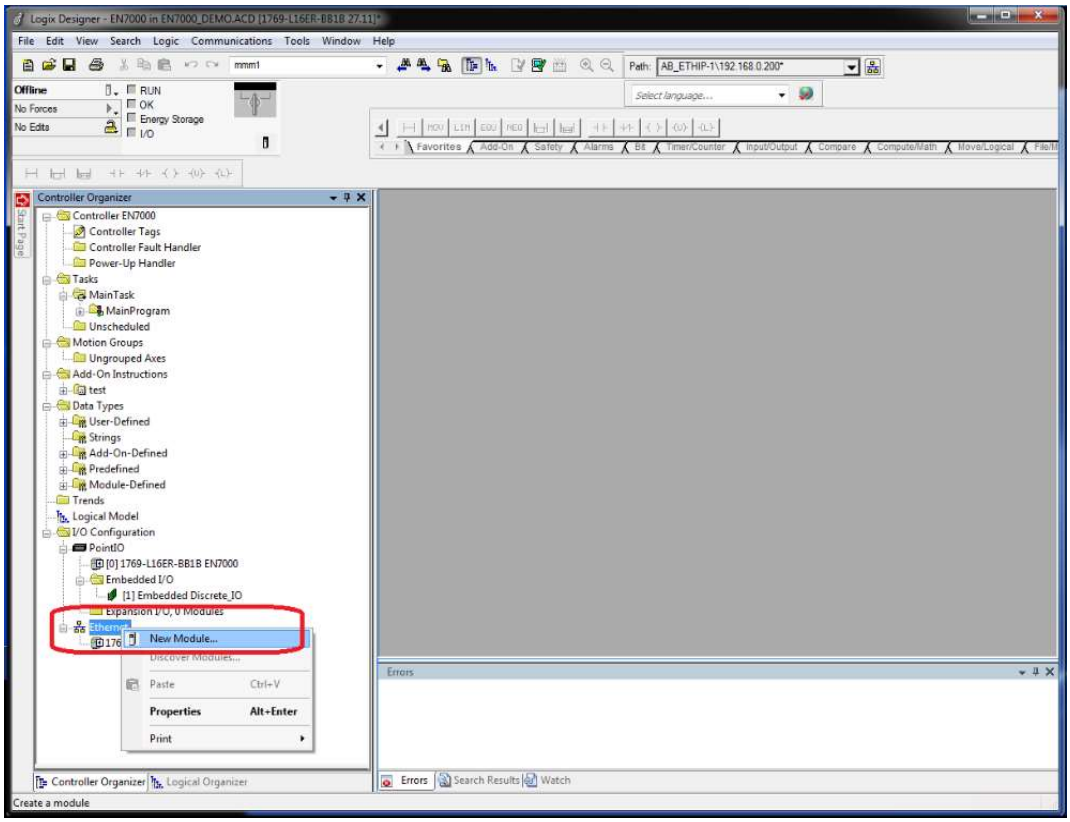


Figure 5 Create new module

c. Select **Generic Ethernet Module** and click on **Create**, as shown in Figure 6.

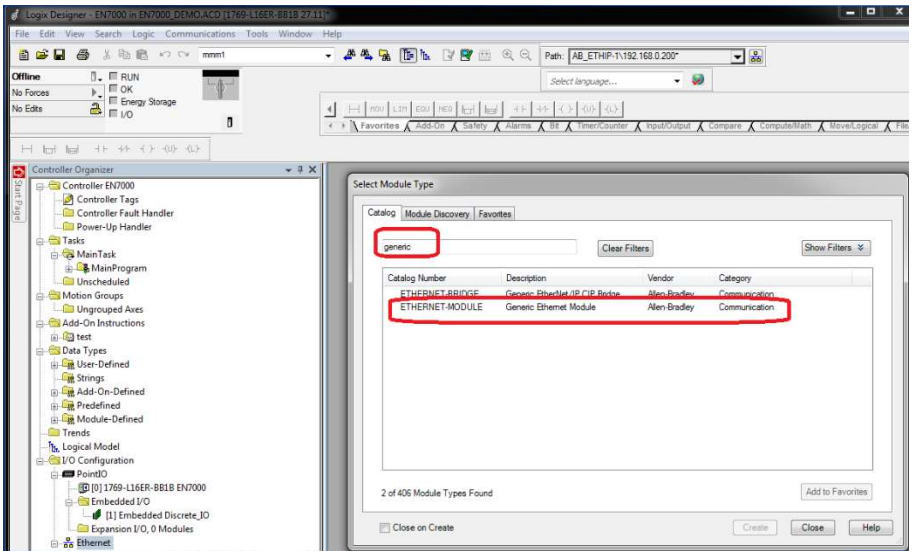


Figure 6 Create Generic Ethernet Module

d. In the **Module Properties** window, enter a name for the new module. In this example the module will be named **ABC**. This will create tags in Studio 5000 which can be used to access the memory location in the PLC where the data for the Communicator is stored. See Figure 7.

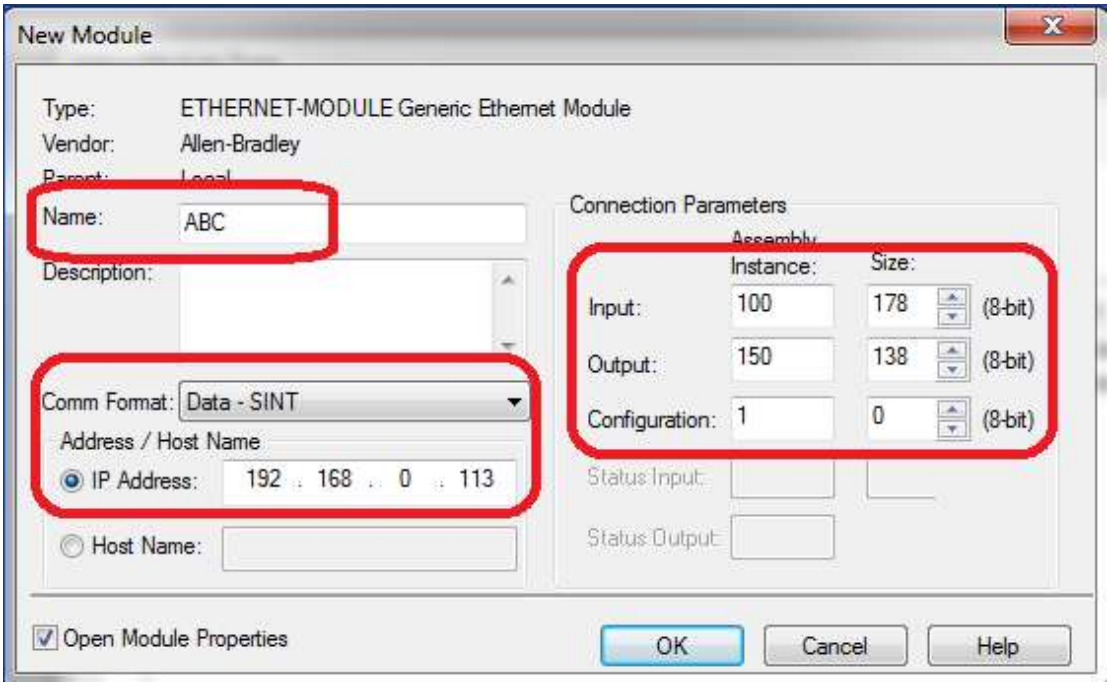


Figure 7 Setup Module properties

e. Select the **Comm Format** (data type). **Data-SINT** is selected, which will represent the data in the ABC as a field of 8-bit values.

f. Enter the **Assembly Instance** parameters. For the Anybus Communicator these values should be Input = **100**, Output = **150** and Configuration = **1**.
g. Enter the **Size** of the input and output data corresponding to the data sizes configured for the ABC, in this case 1**78** bytes in and **138** bytes out.
h. Enter the **IP address** for the module. The default IP address of the ABC is 192.168.0.113.
i. Click on **OK** to confirm the module properties and continue. The ABC has now been added to the I/O configuration in Studio 5000.
j. In the **Module Properties** window, click on **Connection** tab, Enter the **Requested Packet Interval**(RPI), as shown in Figure 8. For this example, **50** (ms) is entered. Click on **OK** to confirm.

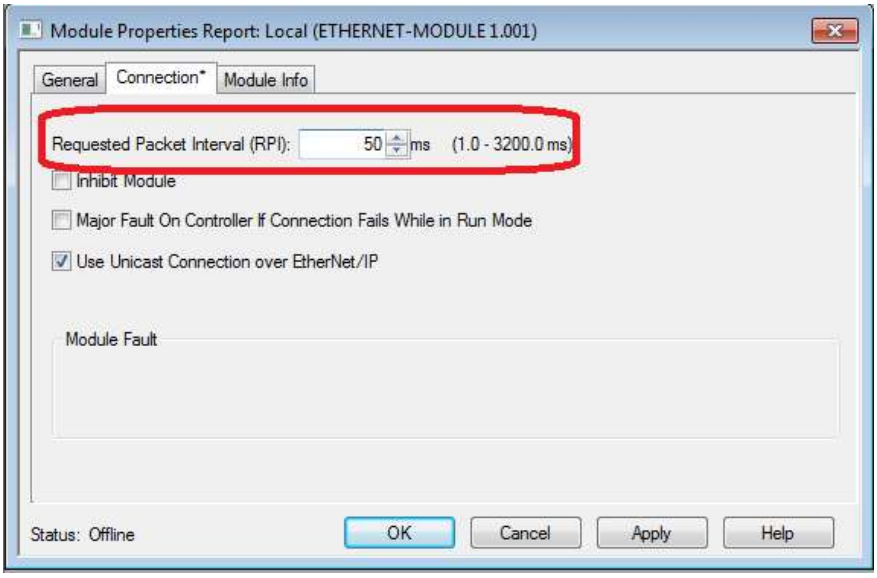


Figure 8 Enter Requested Packet Interval

6.1.2 Adding Data tags

Add necessary data tags to save the Configuration data for the control and variables for ladder diagram.

a. Double click the "Controller Tags" to show the list of All Tags.
b. Create a new 27-integer data tag**, Config_Data**[27], as shown in Figure 9. This tag will save the Configuration data from/to the EN7000 timer. The definition of Configuration data can be found on Configuration Parameters Table in Section 13 of the **EN7000 Technical Manual**.
c. Create a 2-short-integer data tag, **Config_Data_Address**[2], as shown in Figure 10. This tag will save the address of Configuration data in the EN7000 timer. The data structure value can be found in the Modbus Access Type Table in Section 13 of the **EN7000 Technical Manual**.
d. Create a TIMER tag, Timer1. This timer will be use in the process of reading Configuration data from the EN7000 Timer; it provides a 200mS delay.

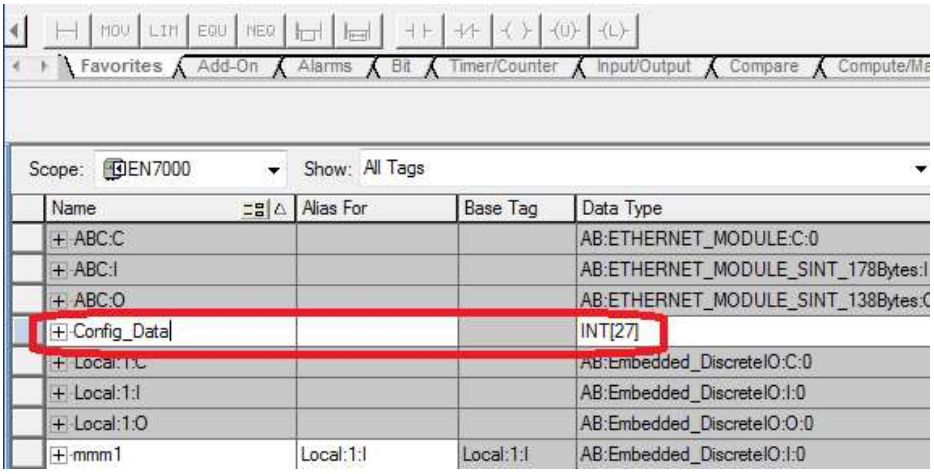


Figure 9 Create data tag Config_Data

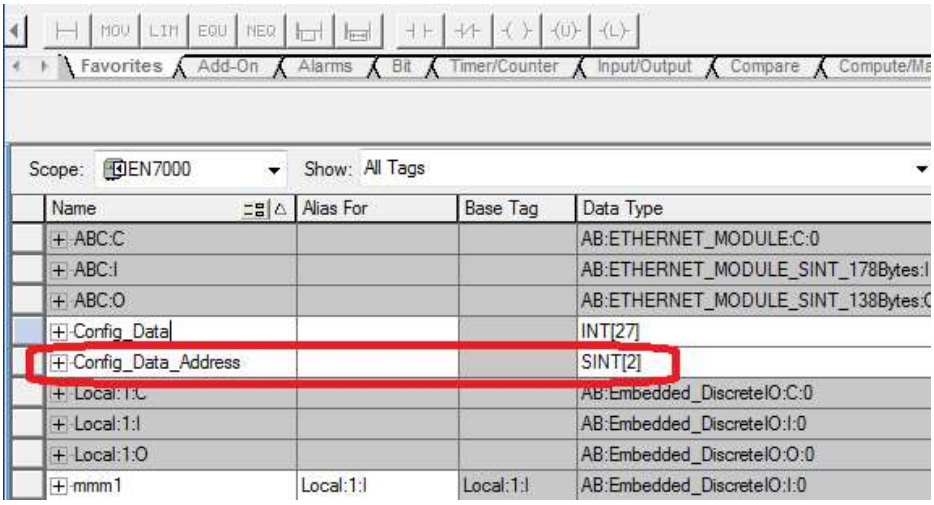


Figure 10 Create data tag Config_Data_Address

e. Click on "+" in front of **Config_Data** to expand the list of tags. Reference the Configuration Parameters Table in Section 13 of the **EN7000 Technical Manual**, fill in the configuration data, which will be written to EN7000 timer. See Figure 11 as the reference.

f. Click on "+" in front of **Config_Data_Address** to expand the list of tag, as shown in Figure 11. Reference the Modbus Mapping Table in Section 13 of the **EN7000 Technical Manual**, type in the address of configuration data (16#6000). **Config_Data_Address**[0] will carry the high byte of address (16#60) and **Config_Data_Address**[1] will carry the low byte of address (16#00). See Figure 12 as the reference.

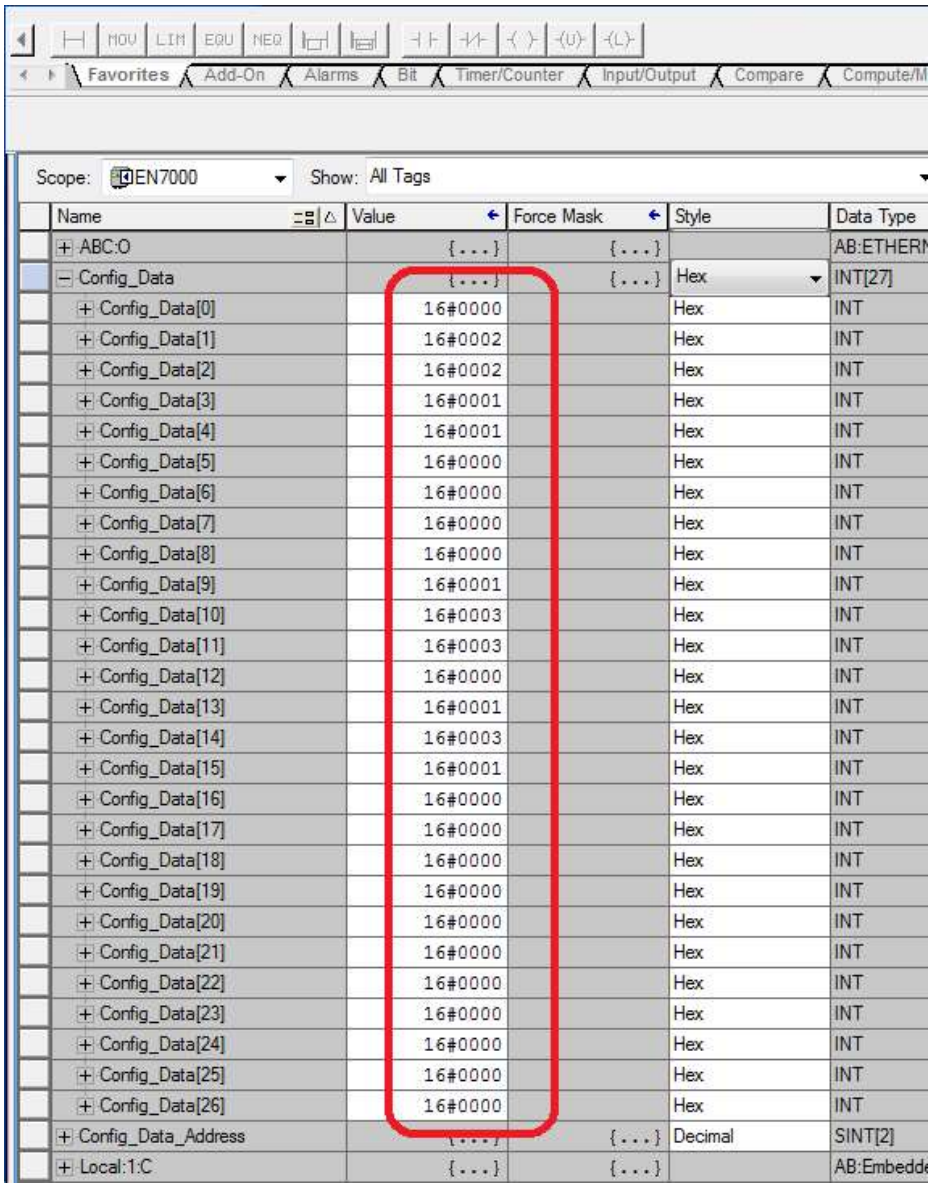


Figure 11 Fill in the configuration data

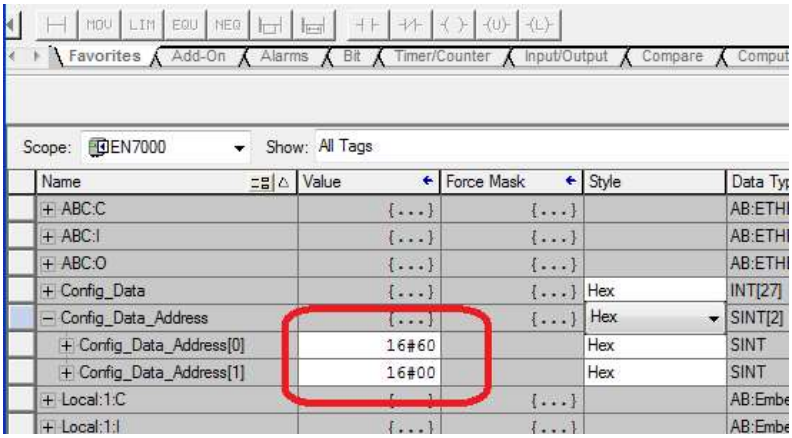


Figure 12 Fill in the address of configuration data in EN7000

6.1.3 Program Ladder Diagram

Figure 13 shows the ladder diagram. Rung 0 and 1 are used to write the Configuration data to the EN7000 timer; rung 2, 3 and 4 are used to read the Configuration data from the timer. Detail functionalities of each instruction blocks are described in the following:

a. Writing Configuration data to EN7000 timer.

- A One Shot Rising instruction **OSR** (1) is used to control that rung 1 is implemented only one time when input IN00 is turned on.

- Copy File instruction **COP** (2) copies the address of Configuration data from the tag **Config_Data_Address** to byte [8] and [9] of the **ABC** output tag (**ABC:O.Data[8]—[9]**).

- Copy File instruction **COP** (3) copies 54-byte of Configuration data from the tag **Config_Data** to byte [10]- [63] of the **ABC** output tag (**ABC:O.Data[10]---[63]**).

- Add instruction ADD (4) reads byte [7] of the **ABC** output tag **(ABC:O.Data[7])**, increment it by one and saves it back to the same byte. . Once the value of **ABC:O.Data[7]** changes, **ABC** will send the data in its output tag to EN7000 timer

b. Reading Configuration data from EN7000 timer.

- A One Shot Rising instruction **OSR** (5) is used to control that rung 3 is implemented only one time when input IN00 is turned on.

- Time On Delay instruction **TON** (6) introduces a 200 milliseconds delay into the process. Once the PLC and the **ABC** send the Read-data instruction to EN7000 timer, the timer needs some time to process the instruction and prepare response data. A 200 milliseconds' delay helps PLC to read back the correct data.

- Copy File instruction **COP** (7) copies the address of Configuration data from the tag **Config_Data_Address** to byte [8] and [9] of the **ABC** output tag (**ABC:O.Data[8]--[9]**).

- Add instruction ADD (8) reads byte [5] of the **ABC** output tag, increment it by one and saves it back to the same byte. Once the value of **ABC:O.Data[5]** changes, the **ABC** will send the instruction data in its output tag to EN7000 timer, and EN7000 timer will send back the response data.

- Copy File instruction **COP** (9) copies 54-byte of Configuration data from byte [48]-[101] of the tag **ABC:O** to the tag **Config_Data**.

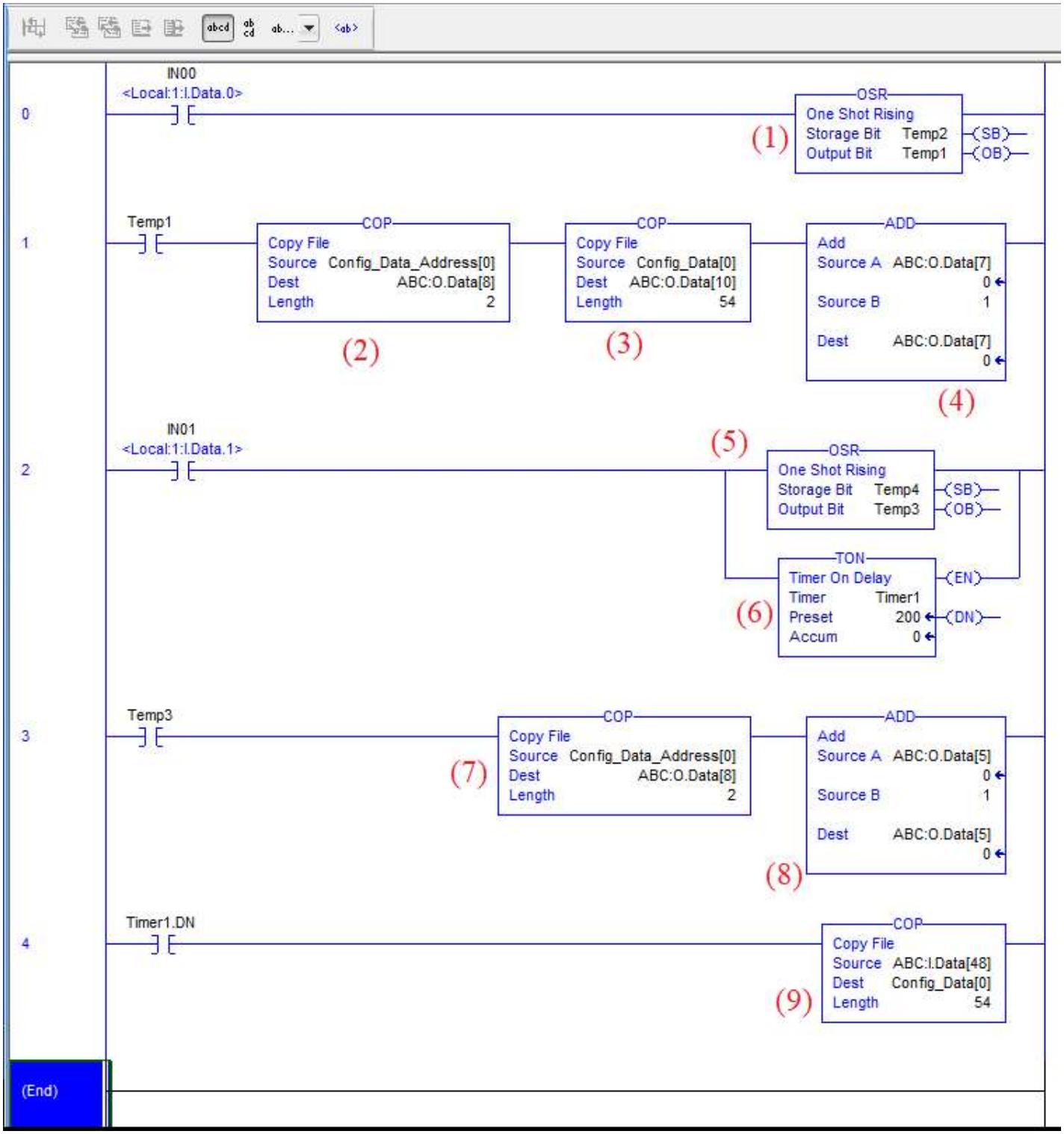


Figure 13 Ladder diagram for Exchanging Configuration data

6.2 Example 2: Exchange Weld Program 2 Data

In this example, when the PLC's Input02 is closed, the PLC will write the Weld Program 2 data to the EN7000 timer; when the PLC's Input03 is closed, the PLC will read the Weld Program 2 data from the EN7000 timer.

The procedures to create and setup the PLC project file are similar to the ones for Example 1: the structure of ladder diagram will be identical; the major difference is the data to be processed. For this example, the tags to store Weld program data will be created and manipulated.

6.2.1 Adding the ABC to the PLC project

Use above section 6.1.1 as the reference to add the ABC to project file.

6.2.2 Adding Data tags

a. Create a new 64-integer data tag, **Weld_Program2_Data**[64], as shown in Figure 14. This tag will save the Weld Program 2 data from/to the EN7000 timer.

b. Create a 2-short-integer data tag **Weld_Program2_Address** [2], as shown in Figure 14. This tag will save the address of Weld Program 2 data in the EN7000 timer.

c. Create a TIMER tag, **Timer2**. This timer will be used in the process of reading Weld_Program2 data from the EN7000 Timer; it provides a 200mS delay.

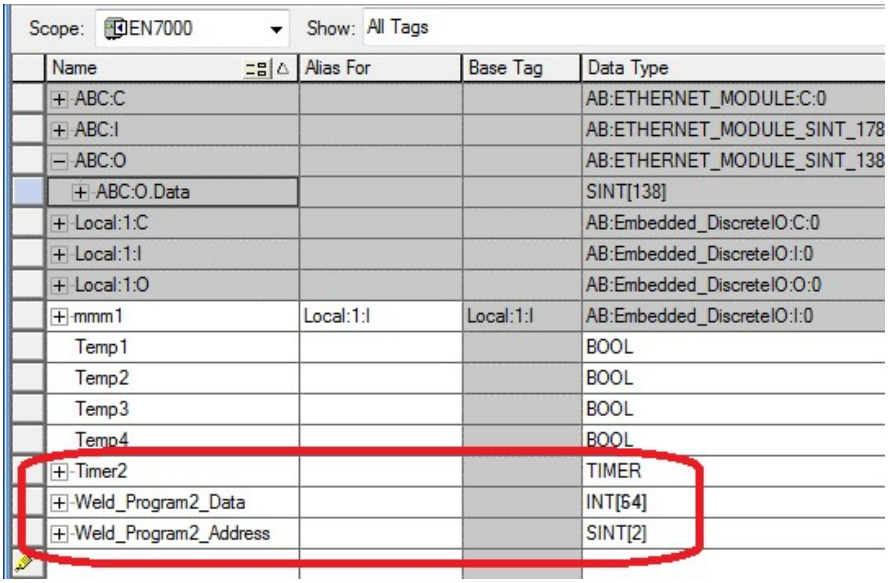


Figure 14 Create tags

d. Click on "+" in front of **Weld_Program2_Data** to expand the list of the tag, as shown in Figure 15. Fill in the weld program data, which will be written to EN7000 timer. The definition of Weld Program data can be found on the Weld Program Parameters Table in Section 13 of the **EN7000 Technical Manual**.

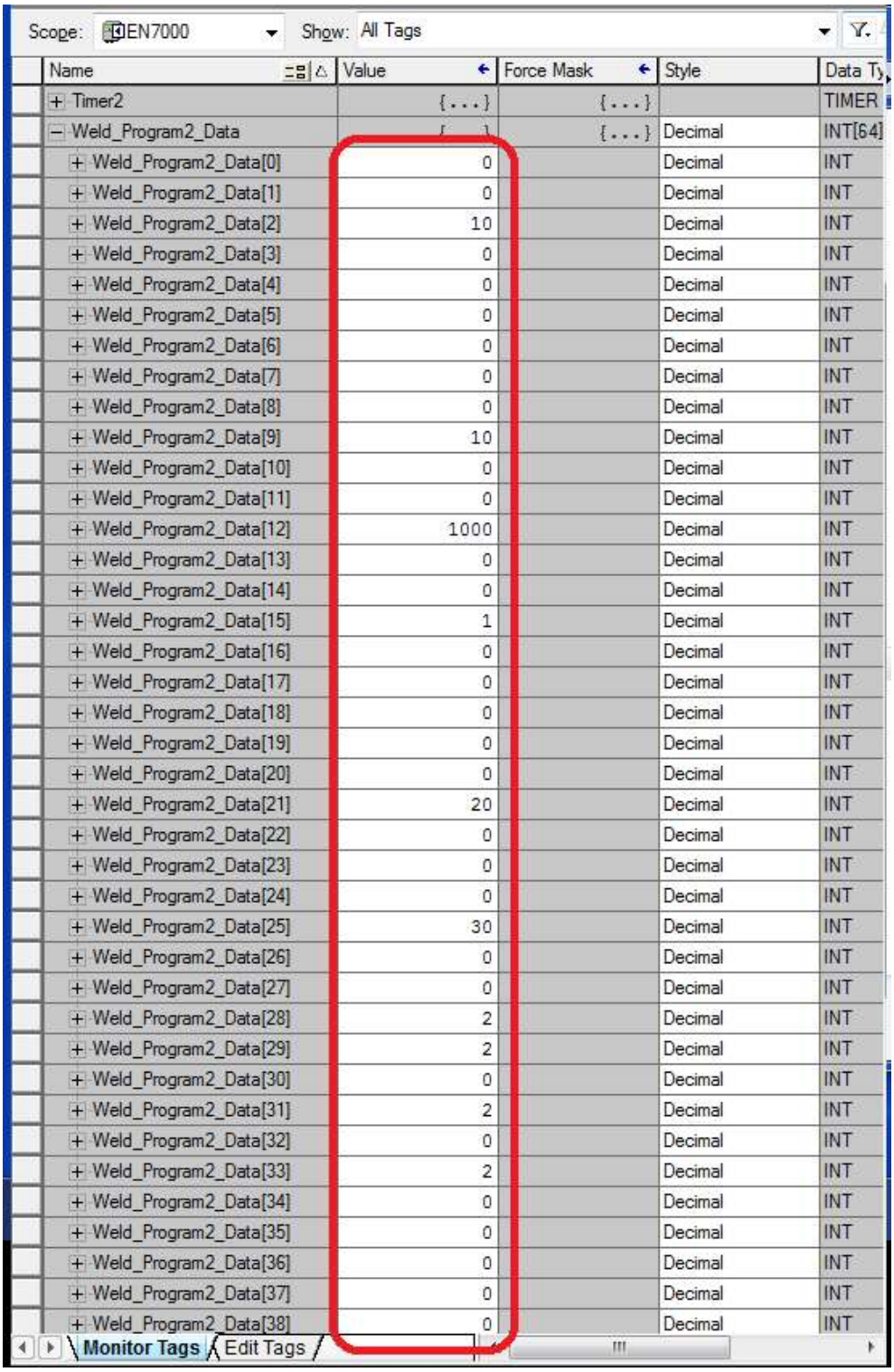


Figure 15 Fill in Weld Program 2 data

e. Click on "+" in front of **Weld_Program2_Address** to expand the list of tag, as shown in Figure 18. Reference the Modbus Mapping Table in the **EN7000 Technical Manual**, type in the address of Weld Program2 data (16#0080). **Weld_Program2_Address** [0] will carry the high byte of address (16#00) and **Weld_Program2_Address** [1] will carry the low byte of address (16#80). See Figure 16 as the reference.

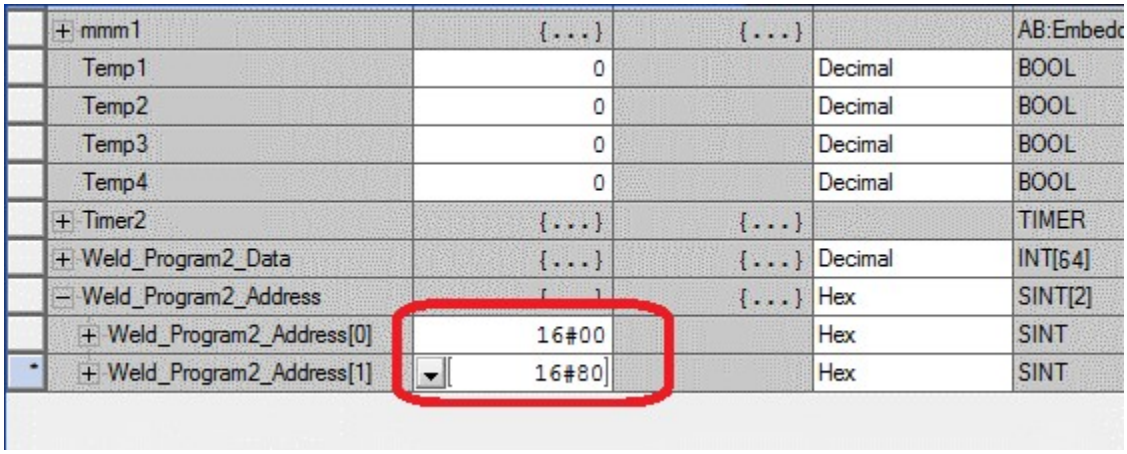


Figure 16 Fill in the address of Weld Program 2 data in EN7000

6.2.3 Program Ladder Diagram

Figure 19 shows the ladder diagram. Rung 0 and 1 are used to write the Weld Program 2 data to EN7000 timer; rung 2, 3 and 4 are used to read the Weld Program 2 data from the timer. Detail functionalities of each instruction blocks are described in the following:

a. Writing the Weld Program 2 data to EN7000 timer.

- One Shot Rising instruction **OSR** (1) is used to control that rung 1 is implemented only one time when input IN02 is turned on.

- Copy File instruction **COP** (2) copies the address of Weld Program 2 data from the tag **Weld_Program2_Address** to byte [8] and [9] of the **ABC** output tag (**ABC:O.Data[8]—[9]**)

- Copy File instruction **COP** (3) copies 128-byte of Weld Program 2 data from the tag **Weld_Program2_Data** to byte [10] - [137] of the **ABC** output tag (**ABC:O.Data[10]---[137]**).

- Add instruction **ADD** (4) reads byte [7] of the **ABC** output tag **(ABC:O.Data[7])**, increments it by one and saves it back to the same byte. Once the value of **ABC:O.Data[7]** changes, **ABC** will send the data in its output tag to EN7000 timer.

b. Reading the Weld Program 2 data from EN7000 timer.

- One Shot Rising instruction **OSR** (5) is used to control that rung 3 is implemented only one time when input IN03 is turned on.

- Time On Delay instruction **TON** (6) introduces a 200 millisecond delay into the process. Once the PLC and ABC send the Read-data instruction to EN7000 timer, the timer needs some time to process the instruction and prepare response data. A 200 millisecond delay helps PLC to read back the correct data.

- Copy File instruction **COP** (7) copies the address of Weld Program 2 data from the tag **Weld_Program2_Address** to byte [8] and [9] of the **ABC** output tag (**ABC:O.Data[8]—[9]**).

- Add instruction ADD (8) reads byte [5] of the **ABC** output tag, increments it by one and saves it back to the same byte. Once the value of **ABC:O.Data[5]** changes, the **ABC** will send the instruction data in its output tag to EN7000 timer, and EN7000 timer will send back the response data.

- Copy File instruction **COP** (9) copies 128-byte of Configuration data from byte [48] - [175] of the tag **ABC:O** to the tag **Weld_Program2_Data**.

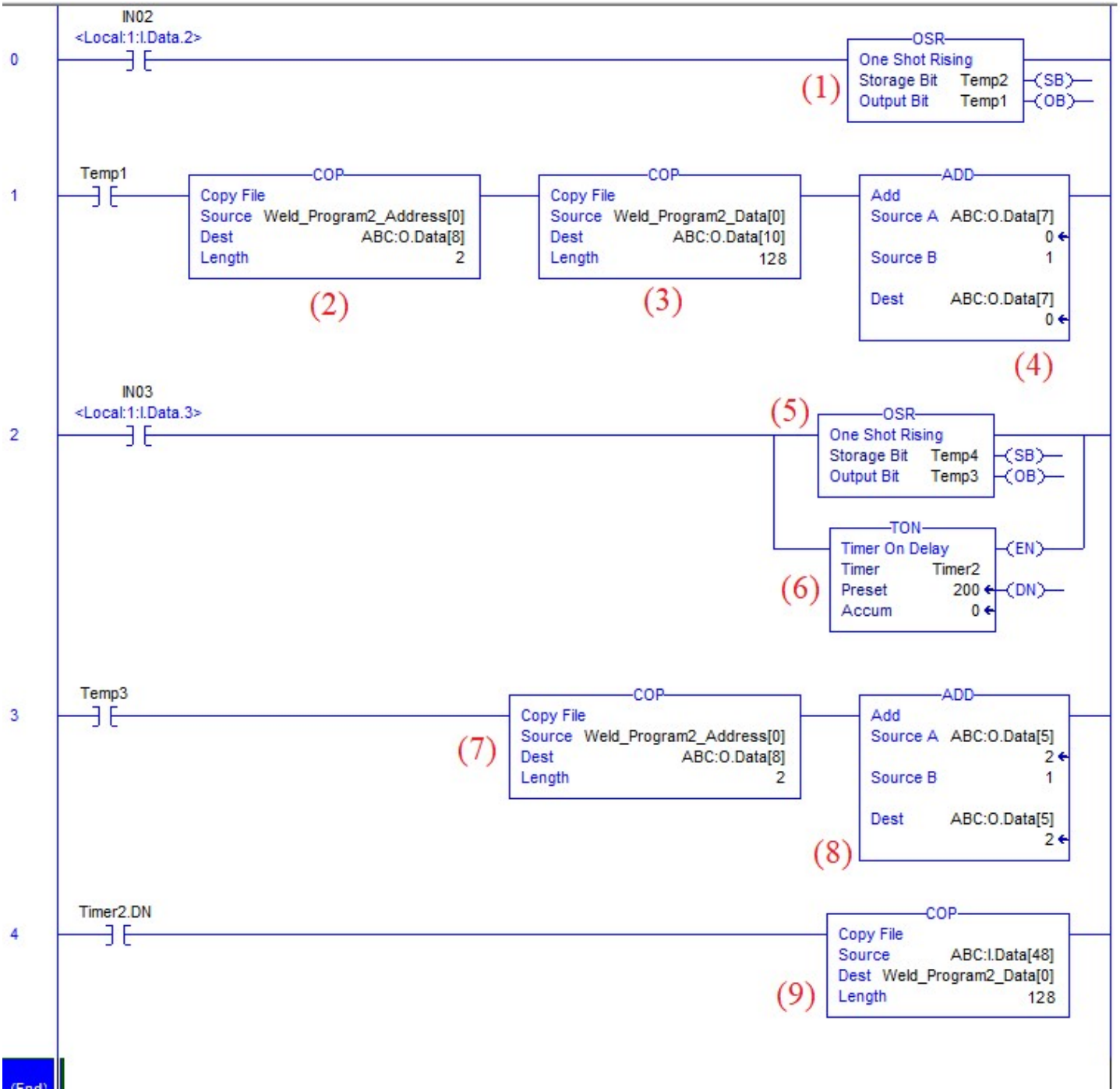


Figure 17 Ladder diagram for exchanging Weld Program 2 data