Expanding
Possibility

Function Manual for Hi6 Controller

Hi6 Controller Modbus Manual

HD HYUNDAI ROBOTICS

HD HYUNDAI ROBOTICS

Table of Contents

1. Overview

1.1 Preparatory information

The following information is needed in advance to understand the manual.

1. Knowledge about the operation of the Hi6 robot controller
2. Knowledge about the MODBUS protocol

1.2 Functions of MODBUS

The Hi6 robot controller supports the MODBUS master and slave functions both through serial communication and Ethernet communication.

1. Example of MODBUS master operation

   - Control of equipment :

     Capable of controlling the equipment (e.g., gripper) that supports MODBUS

MODBUS (serial or ethernet)

2. Example of MODBUS slave operation

- Function as an operation panel:

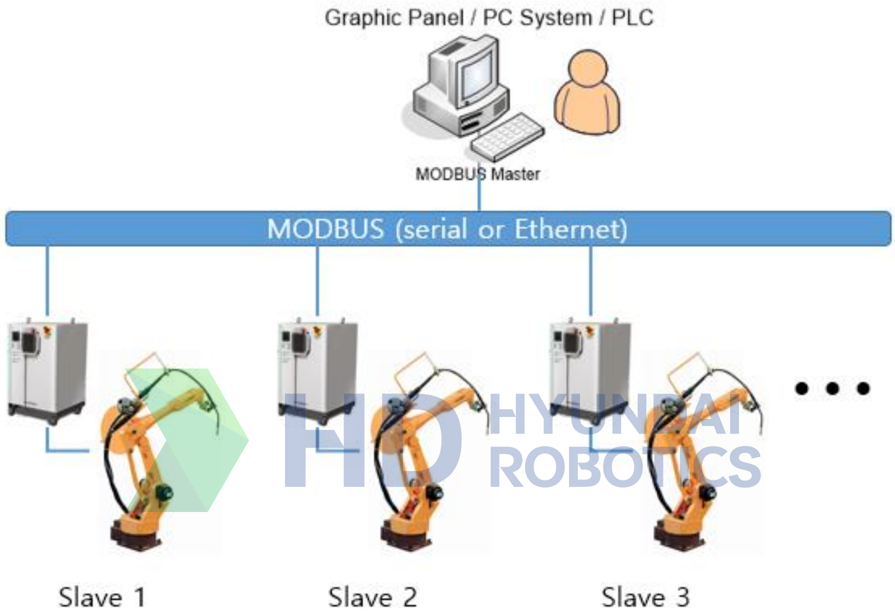
  With an inexpensive graphic panel (GP) that supports MODBUS, you can use one or multiple robots by connecting them through serial or Ethernet communication.

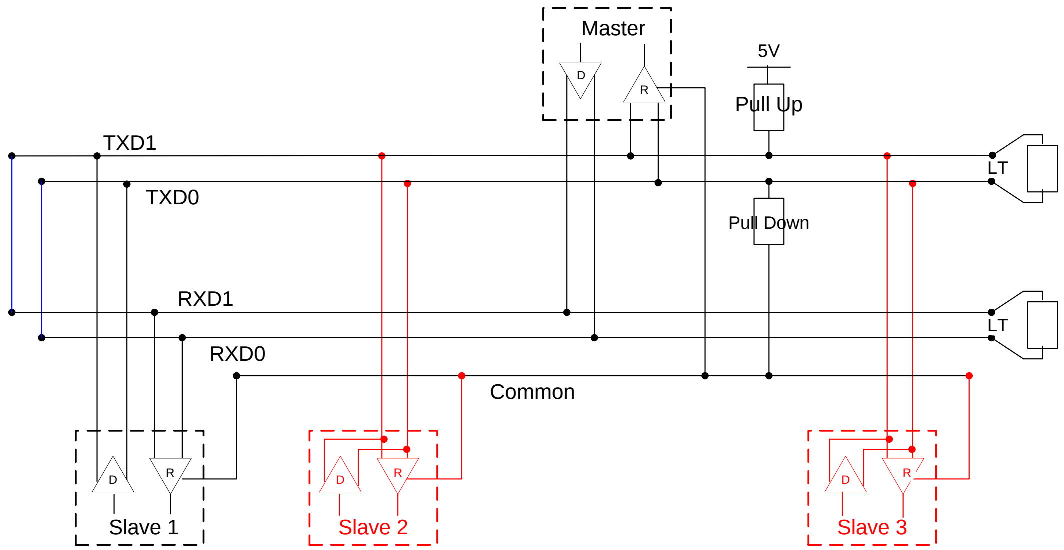
- Programmable Logic Controller (PLC) communication :

  Provides an inexpensive solution for communication with PLCs that have the MODBUS master function.

- PC-based robot operation system :

  Allows a robot operation system to be built that monitors or controls the robot's input and output signals using a PC.





3. Support method

	Serial communication	Ethernet communication
Operation of master	- Robot language statement	
Operation of slave	- Setting in controller	- Ip: Setting in controller - Port: 502 (fixed)

4. Transmission mode

	Serial communication	Ethernet communication
Operation of master	- Binary mode	
Operation of slave	- ASCII mode - RTU (binary) mode	- Binary mode

5. Functions supported

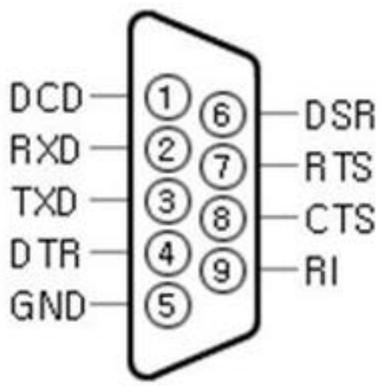
	Serial and Ethernet communication
Operation of master	- 03: Read holding registers (multiple) - 16: Write holding registers (multiple)
Operation of slave	- 01: Read coils (bits) - 02: Read discrete inputs (bits) - 03: Read holding registers (multiple) - 04: Read input registers (multiple) - 05: Write single coil (bit) - 06: Write single holding register - 15: Write coils (multiple bits) - 16: Write holding registers (multiple)

6. Slave address

    - Slave address: 1–247
    - If the slave address of a command is 0, the broadcasting function, which allows all slaves to operate regardless of the set address, will be supported.

7. Serial communication connection

    - Connector (DSUB – 9pin: female)



DB-9 RS232 Connector

    - Pin map

Suggested DB9 Connector Pinout

DB9 Pin	RS-232	RS-485/RS-422 Full Duplex	RS-485 Half Duplex
1	DCD	TX-	Data-
2	RXD	TX+	Data+
3	TXD	RX+	
4	DTR	RX-	
5	Ground		
6	DSR		
7	RTS		
8	CTS		
9	RI		

8. Address map

MODBUS data model	Relay name	Relay mapping: 1 bit Register	Relay mapping: 1 bit Logical addr.	Relay mapping: 16 bit Register	Relay mapping: 16 bit Logical addr.	Function
Input Discrete Add: 0x0000–0xffff Quantity: 1–2039 (bit)	MW					Read Function 02: Read discrete Inputs (bits) 04: Read input registers (multiple)
	DO					
	DI	di0–959 … fb9.di0–959 *1*	12000–13999	diw0–118 … fb9.diw0–118 *3*	12000–13999	
Input Registers Add: 0x0000–0xffff Quantity: 1–127	SO					
	SI	si0–959	15000–16999	siw0–118	15000–16999	
	SW					
Coils Add: 0x0000–0xffff Quantity: 1–2039 (bit)	MW			_mw0–9999	0–9999	Read Function 01: Read coils (bits) 03: Read holding registers (multiple) Write Function 05: Write single coil (bit) 15: Write coils (multiple bits) 06: Write single holding register 16: Write holding registers (multiple)
	DO	do0–959 … fb9.do0–959 *0*	10000–11999	dow0–118 … fb9.dow0–118 *4*	10000–11999	
	DI	di0–959 … fb9.di0–959	12000–13999	diw0–118 … fb9.diw0–118	12000–13999	
Holding Registers Add: 0x0000–0xffff Quantity: 1–127	SO	so0–959	14000–14999	sow0–118	14000–14999	
	SI	si0–959	15000–16999	siw0–118	15000–16999	
	SW			_sw0–9999	16000–65535	

- The enlarged numbers in italics in the table above represent the relay groups used in the MODBUS function.
- MW (data memory for user), DO (digital output), DI (digital input), SO (system output), SI (system input), and SW (system memory)
- Data format: For the float format, IEEE single-precision 32-bit float point is used, and, in the case of 8 bit / 16 bit / 32 bit, signed integers are used all.
- For the relay's endian, little endian is used.
- Example: In the case of dof0=6.515625(0x40D08000), which is in the float format
- dol0=0x40D08000 → dow0=0x8000, dow2=0x40D0 → dob0=0x00, dob1=0x80, dob2=0xD0,

dob3=0x40

- Explanation: In MODBUS transmission, big endian of 16-bit align is used. In other words, the transmission described above will be performed in the order of 0x80, 0x00, 0x40, and 0xD0.

9. SW memory map

Relay	MODBUS address (0-based, decimal)	ProConOs shared memory: Data type	ProConOs shared memory: Address	Description	Remarks
PLC-related					
SW0	16000	INT	%MW3.32000	PLC execution mode (0 = On, 1 = Holding, 2 = Starting, 3 = Running, 4 = Halt requested, 5 = Halt, 6 = Stopping, 7 = Stop, 8 = Resetting, Others = Unknown)	
Software version					
SW5	16005	INT	%MW3.32010	1st of the Main SW Version	60.01-02 -> 60
SW6	16006	INT	%MW3.32012	2nd of the Main SW Version	60.01-02 -> 01
SW7	16007	INT	%MW3.32014	3rd of the Main SW Version	60.01-02 -> 02
Program counter					
SW101	16101	INT	%MW3.32202	Current program number of the controller	
SW102	16102	INT	%MW3.32204	Current step number of the controller	
SW103	16103	INT	%MW3.32206	Current function number of the controller	
SW104	16104	INT	%MW3.32208	Main program number of the controller	
Total time of operation					
SW199	16199	INT	%MW3.32398	Selection mode 0=Invalid 1= Communication(After initialization) 2= Communication(After power on) 3= Last cycle 4= Current cycle	

SL200	16200	DINT	%MD3.32400	Days of motor on	
SL202	16202	DINT	%MD3.32404	Time of motor on (in ms)	
SL204	16204	DINT	%MD3.32408	Days of operation	
SL206	16206	DINT	%MD3.32412	Time of operation (in ms)	
SL208	16208	DINT	%MD3.32416	Days of movement	
SL210	16210	DINT	%MD3.32420	Time of movement (in ms)	
SL212	16212	DINT	%MD3.32424	Count of cycles	
SL214	16214	DINT	%MD3.32428	Days of wait and D1 wait	
SL216	16216	DINT	%MD3.32432	Days of wait and D1 wait (in ms)	
SL218	16218	DINT	%MD3.32436	Days of wait by timer	
SL220	16220	DINT	%MD3.32440	Time of wait by timer (in ms)	
Robot position					
SF300	16240	REAL	%MD3.32600	Base coordinate value X (Unit: mm)	
SF302	16242	REAL	%MD3.32604	Base coordinate value Y (Unit: mm)	
SF304	16244	REAL	%MD3.32608	Base coordinate value Z (Unit: mm)	
SF306	16246	REAL	%MD3.32612	Base coordinate value RX (Unit: deg)	
SF308	16248	REAL	%MD3.32616	Base coordinate value RY (Unit: deg)	
SF310	16250	REAL	%MD3.32620	Base coordinate value RZ (Unit: deg)	
SF312	16252	REAL	%MD3.32624	Position of axis 1 (Unit: mm or deg)	
SF314	16254	REAL	%MD3.32628	Position of axis 2 (Unit: mm or deg)	
SF316	16256	REAL	%MD3.32632	Position of axis 3 (Unit: mm or deg)	
SF318	16258	REAL	%MD3.32636	Position of axis 4 (Unit: mm or deg)	
SF320	16260	REAL	%MD3.32640	Position of axis 5 (Unit: mm or deg)	
SF322	16262	REAL	%MD3.32644	Position of axis 6 (Unit: mm or deg)	
SF324	16264	REAL	%MD3.32648	Position of axis 7 (Unit: mm or deg)	
SF326	16266	REAL	%MD3.32652	Position of axis 8 (Unit: mm or deg)	
SF328	16268	REAL	%MD3.32656	Position of axis 9 (Unit: mm or deg)	
SF330	16270	REAL	%MD3.32660	Position of axis 10 (Unit: mm or deg)	
SF332	16272	REAL	%MD3.32664	Position of axis 11 (Unit: mm or deg)	
SF334	16274	REAL	%MD3.32668	Position of axis 12 (Unit: mm or deg)	
SF336	16276	REAL	%MD3.32672	Position of axis 13 (Unit: mm or deg)	
SF338	16278	REAL	%MD3.32676	Position of axis 14 (Unit: mm or deg)	

SF340	16280	REAL	%MD3.32680	Position of axis 15 (Unit: mm or deg)	
SF342	16282	REAL	%MD3.32684	Position of axis 16 (Unit: mm or deg)	

Robot speed

SW349	16349	INT	%MW3.32698	Selection mode 0= Invalid 1= Speed of axis (Unit:mm/s or deg/s) 2= Speed of Motor (Unit:rpm)	
SF350	16350	REAL	%MD3.32700	Speed of axis 1 (Unit: mm/s or deg/s)	
SF352	16352	REAL	%MD3.32704	Speed of axis 2 (Unit: mm/s or deg/s)	
SF354	16354	REAL	%MD3.32708	Speed of axis 3 (Unit: mm/s or deg/s)	
SF356	16356	REAL	%MD3.32712	Speed of axis 4 (Unit: mm/s or deg/s)	
SF358	16358	REAL	%MD3.32716	Speed of axis 5 (Unit: mm/s or deg/s)	
SF360	16360	REAL	%MD3.32720	Speed of axis 6 (Unit: mm/s or deg/s)	
SF362	16362	REAL	%MD3.32724	Speed of axis 7 (Unit: mm/s or deg/s)	
SF364	16364	REAL	%MD3.32728	Speed of axis 8 (Unit: mm/s or deg/s)	
SF366	16366	REAL	%MD3.32732	Speed of axis 9 (Unit: mm/s or deg/s)	
SF368	16368	REAL	%MD3.32736	Speed of axis 10 (Unit: mm/s or deg/s)	
SF370	16370	REAL	%MD3.32740	Speed of axis 11 (Unit: mm/s or deg/s)	
SF372	16372	REAL	%MD3.32744	Speed of axis 12 (Unit: mm/s or deg/s)	
SF374	16374	REAL	%MD3.32748	Speed of axis 13 (Unit: mm/s or deg/s)	
SF376	16376	REAL	%MD3.32752	Speed of axis 14 (Unit: mm/s or deg/s)	
SF378	16378	REAL	%MD3.32756	Speed of axis 15 (Unit: mm/s or deg/s)	
SF380	16380	REAL	%MD3.32760	Speed of axis 16 (Unit: mm/s or deg/s)	

Robot load factor

SW399	16399	INT	%MW3.32798	Load factor selection (0 = Invalid, 1 = I/Ir, 2 = I/Ip, 3 = Continuous)	
SF400	16400	REAL	%MD3.32800	Load factor of axis 1	
SF402	16402	REAL	%MD3.32804	Load factor of axis 2	
SF404	16404	REAL	%MD3.32808	Load factor of axis 3	
SF406	16406	REAL	%MD3.32812	Load factor of axis 4	
SF408	16408	REAL	%MD3.32816	Load factor of axis 5	

SF410	16410	REAL	%MD3.32820	Load factor of axis 6	
SF412	16412	REAL	%MD3.32824	Load factor of axis 7	
SF414	16414	REAL	%MD3.32828	Load factor of axis 8	
SF416	16416	REAL	%MD3.32832	Load factor of axis 9	
SF418	16418	REAL	%MD3.32836	Load factor of axis 10	
SF420	16420	REAL	%MD3.32840	Load factor of axis 11	
SF422	16422	REAL	%MD3.32844	Load factor of axis 12	
SF424	16424	REAL	%MD3.32848	Load factor of axis 13	
SF426	16426	REAL	%MD3.32852	Load factor of axis 14	
SF428	16428	REAL	%MD3.32856	Load factor of axis 15	
SF430	16430	REAL	%MD3.32860	Load factor of axis 16	

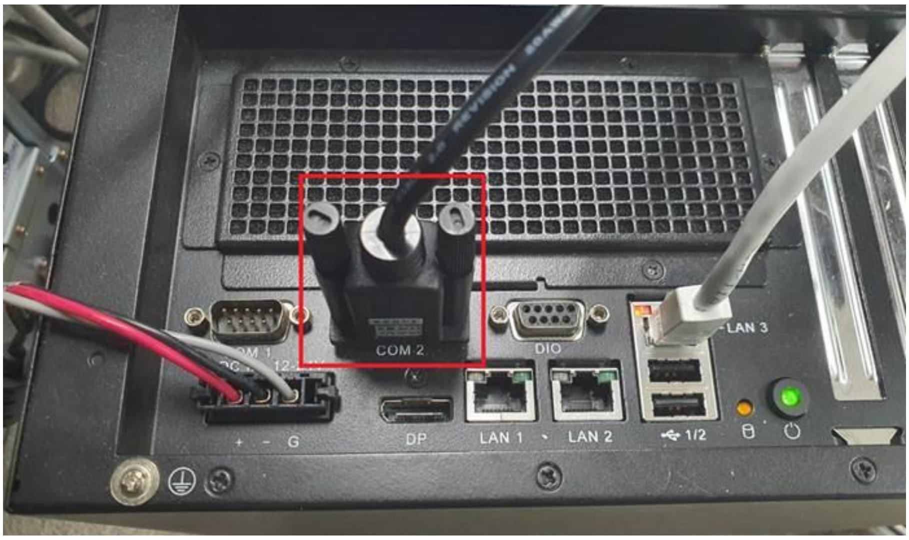
Conveyor synchronization

SW2200	18200	INT	%MW3.36400	Pulse data (Channel 1)	
SW2201	18201	INT	%MW3.36402	Workpiece position (Channel 1)	
SW2202	18202	INT	%MW3.36404	Moving speed (Channel 1)	
SW2203	18203	INT	%MW3.36406	Count of workpieces for entry (Channel 1)	
SW2204	18204	INT	%MW3.36408	Limit switch input (Channel 1)	
SW2205	18205	INT	%MW3.36410	Raw pulse data (Channel 1)	
SW2210	18210	INT	%MW3.36420	Pulse data (Channel 2)	
SW2211	18211	INT	%MW3.36422	Workpiece position (Channel 2)	
SW2212	18212	INT	%MW3.36424	Moving speed (Channel 2)	
SW2213	18213	INT	%MW3.36426	Count of workpieces for entry (Channel 2)	
SW2214	18214	INT	%MW3.36428	Limit switch input (Channel 2)	
SW2215	18215	INT	%MW3.36430	Raw pulse data (Channel 2)	

2. Serial Communication Setting

2.1 Serial cable connection

The serial cable is erectly connected to the COM2 port as shown in the figure below.

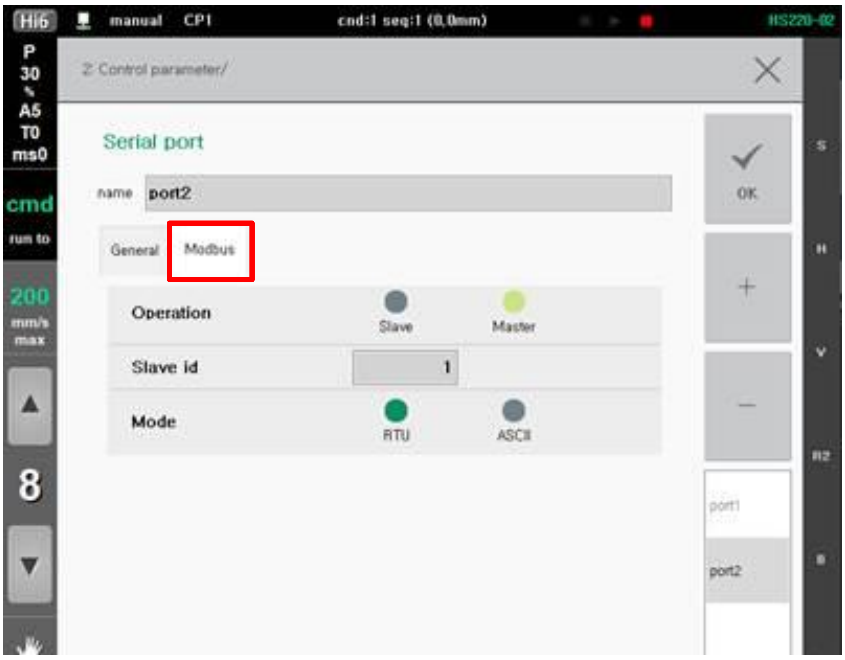


2.2 Serial port usage setting

The usage of the serial port can be set as MODBUS as shown below in the 『General』 tab on the 『Set Up → 2: Control parameter → 3: Serial port』 screen.

2.3 MODBUS environment setting

Details regarding MODBUS can be set in the 『Modbus』 tab, as shown below.



- Operation : You can select whether to operate the port as master or slave. When it is operated as master, the operation will be executed according to the robot language commands; thus, the Slave id and Mode will not be used.
- Slave id : You can set the id for the communication as slave for MODBUS serial communication.
- Mode : You can set the mode for the communication as slave for MODBUS serial communication.

3. Operation as Master

3.1 Robot Language

You can configure a Modbus master query using robot language statements and transmit it to the slave. Data will be transmitted and received when the statement is executed. In case it is necessary to transmit and receive data periodically, you can configure a job program for that purpose and then perform operation in multitask mode. However, in this case, it is recommended to communicate through the built-in PLC.

3.1.1. Command (Modbus)

Description	This statement is for Modbus master communication.		
Syntax	modbus enet2,sid=65,fc=16,addr=0,len=3,wait=3,var=arr		
Parameter	enet2	Object of communication (Ethernet or Serial)	
	sid	Slave id	1 ~ 255
	fc	Function code 03 = read holding register(multiple) 16 = write multiple register	03, 16
	addr	Start address	0 ~ 65534
	len	Data length	1 ~ 127
	wait	Communication waiting time	
	var	Array variable for transmitting/receiving data (an internal Modbus map is used if not specified)	
Details	• This statement is for performing Modbus master communication in robot language. • Please learn and study this separately for understanding Modbus communication.		

3.1.2. Sample program

3.1.2.1. Ethernet communication

The following shows a sample program for controlling the onRobot gripper. The Hi6 controller and onRobot gripper communicate through MODBUS tcp. In the example, the Hi6 controller is operated as master and the gripper as slave.

0060.job

```
Hyundai Robot Job File; { version: 1.6, mech_type: "780(YL012-0D)", total_axis: 6, aux_axis: 0 }
call 61,1 # onRobot module open
call 61,2,0 # onRobot gripper hold
delay(3)
call 61,2,300 # onRobot gripper release
call 61,0 # onRobot module close
delay(3)
end
```

0061.job

```
Hyundai Robot Job File; { version: 1.6, mech_type: "780(YL012-0D)", total_axis: 6, aux_axis: 0 }
param mode,grip
if (mode == 1) # enet module open
  import enet
  global enet2,arr
  if (arr==0)
    arr=Array(5)
  endif
  # onRobot gripper enet connect
  enet2=enet.ENet("tcp") #udp,tcp
  enet2.ip_addr="192.168.1.111" #OnRonot IP
  enet2.lport=502
  enet2.rport=502
  if (enet2.state() < 1)
    enet2.open
    enet2.connect #tcp is the case
  else
    stop
  endif
  print "enet2.state", enet2.state()
elseif (mode == 0) # enet module close
  enet2.close
else # onRobot gripper operate
```

```
  arr[0] = 300 # force (0~400)
  arr[1] = grip # width (0~1100)
  arr[2] = 1 # control (1:grip, 8=stop, 16=offset grip)
  modbus enet2,sid=65,fc=16,addr=0,len=3,wait=3,var=arr
endif
end
```

3.1.2.2. Serial communication

The following shows a sample program when assuming that the onRobot gripper is controlled through serial communication. First, in the serial communication setting, port usage should be set as 〈MODBUS〉, and MODBUS operation should be set as 〈master〉.

0060.job

```
Hyundai Robot Job File; { version: 1.6, mech_type: "780(YL012-0D)", total_axis: 6, aux_axis: 0 }
call 61,1 # onRobot module open
call 61,2,0 # onRobot gripper hold
delay(3)
call 61,2,300 # onRobot gripper release
call 61,0 # onRobot module close
delay(3)
end
```


0061.job

```
Hyundai Robot Job File; { version: 1.6, mech_type: "780(YL012-0D)", total_axis: 6, aux_axis: 0 }
param mode,grip
if (mode == 1) # enet module open
  global sci2,arr
  if (arr==0)
    arr=Array(5)
  endif
  # onRobot gripper enet connect
  sci2=com.Sci(2) # serial port 2 object
elseif (mode == 0) # enet module close
  print “sci close”
else # onRobot gripper operate
  arr[0] = 300 # force (0~400)
  arr[1] = grip # width (0~1100)
  arr[2] = 1 # control (1:grip, 8=stop, 16=offset grip)
```

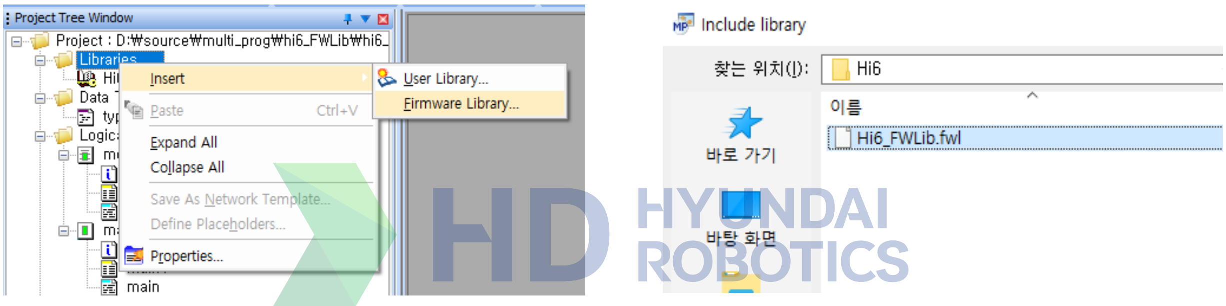
```
  modbus sci2,sid=65,fc=16,addr=0,len=3,wait=3,var=arr
endif
end
```

3.2 Embedded PLC

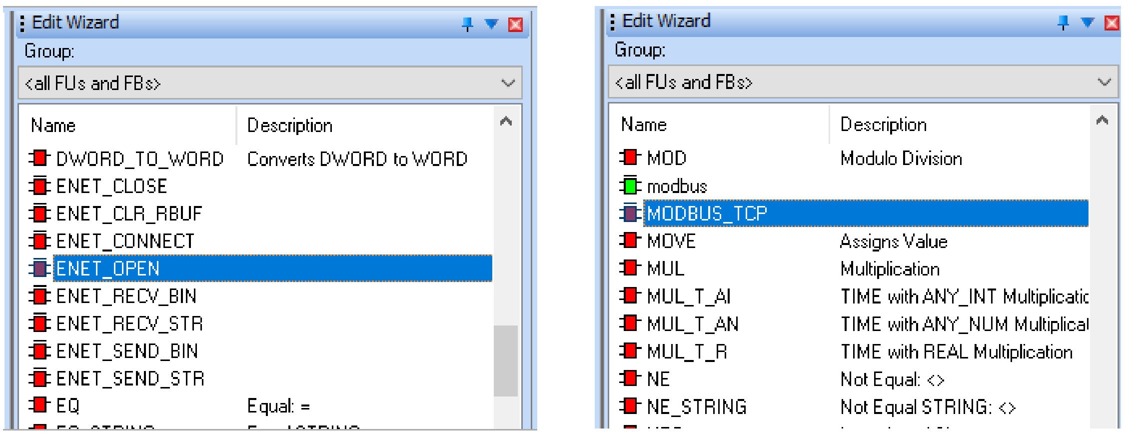
The built-in PLC ladder logic makes it possible to configure a Modbus master query and transmit it to the slave. You need knowledge about the built-in PLC and should refer to the related manuals to use this function.

3.2.1. Addition of native firmware library

In Libraries 〉 Insert 〉 Firmware Library, select the “Hi6_FWLib.fwl” file, and add it, as shown below.

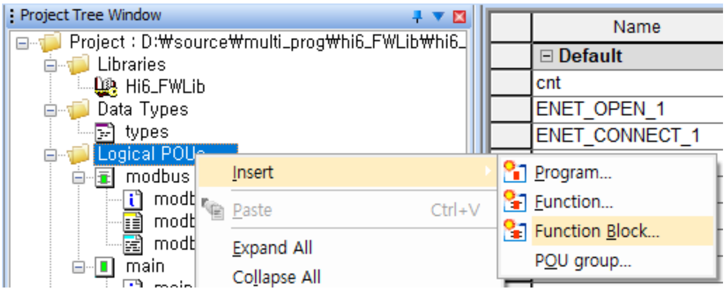


In the Edit Wizard, you can see that the function blocks related to ENET communication and those related to MODBUS_TCP have been added.

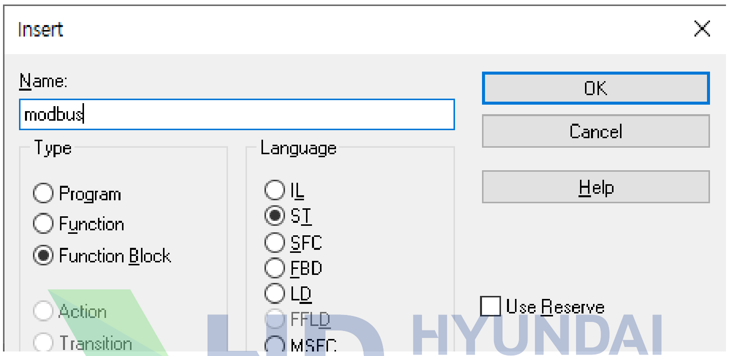


3.2.2. Addition of function blocks

In Logical POUs 〉 Insert 〉 Function Block, as shown in the figure below, you can add a function block.

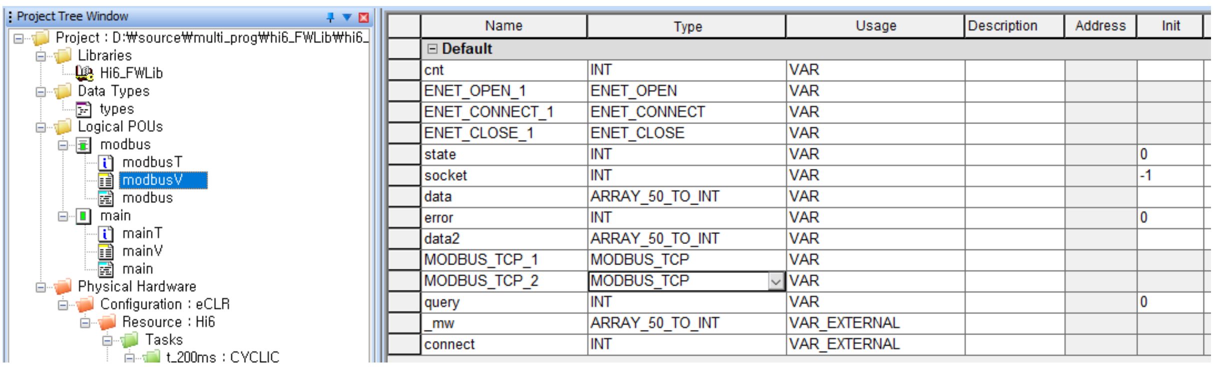


In this example, we have chosen to use the ST language under Modbus.



3.2.3. Addition of variables

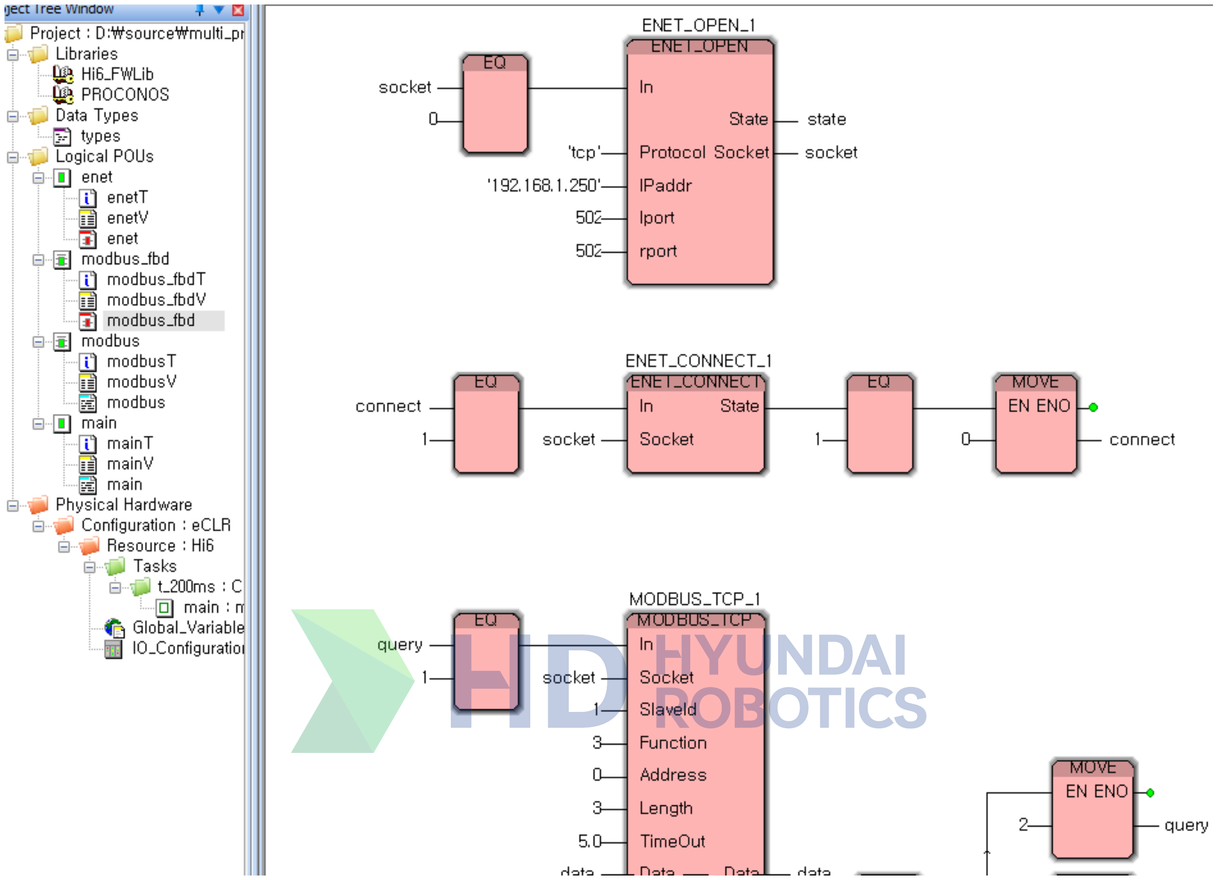
In the variable management screen, you can register variables, as shown in the figure below.



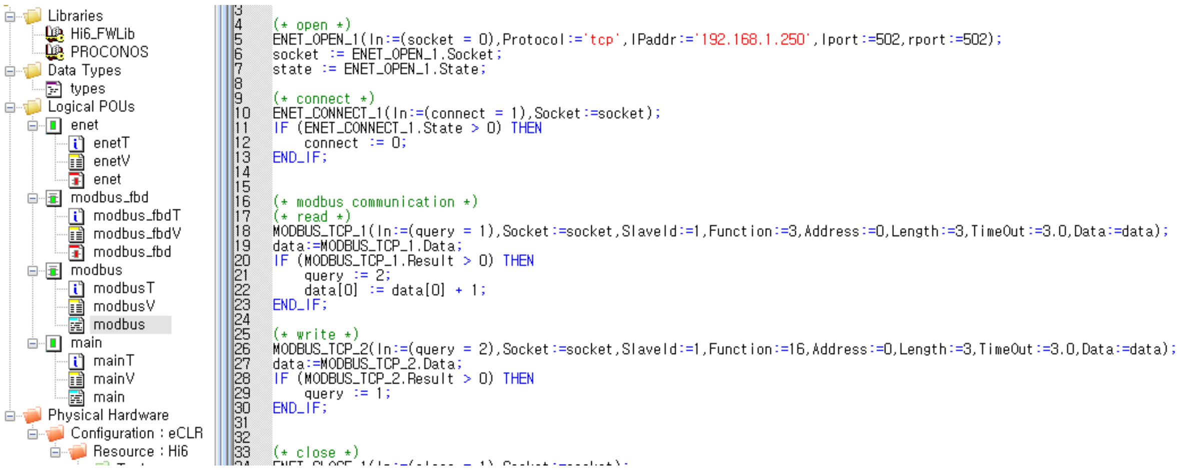
3.2.4. Creation of function blocks

As shown in the following figure, the user can write a program on the program writing screen.

(FBD language)

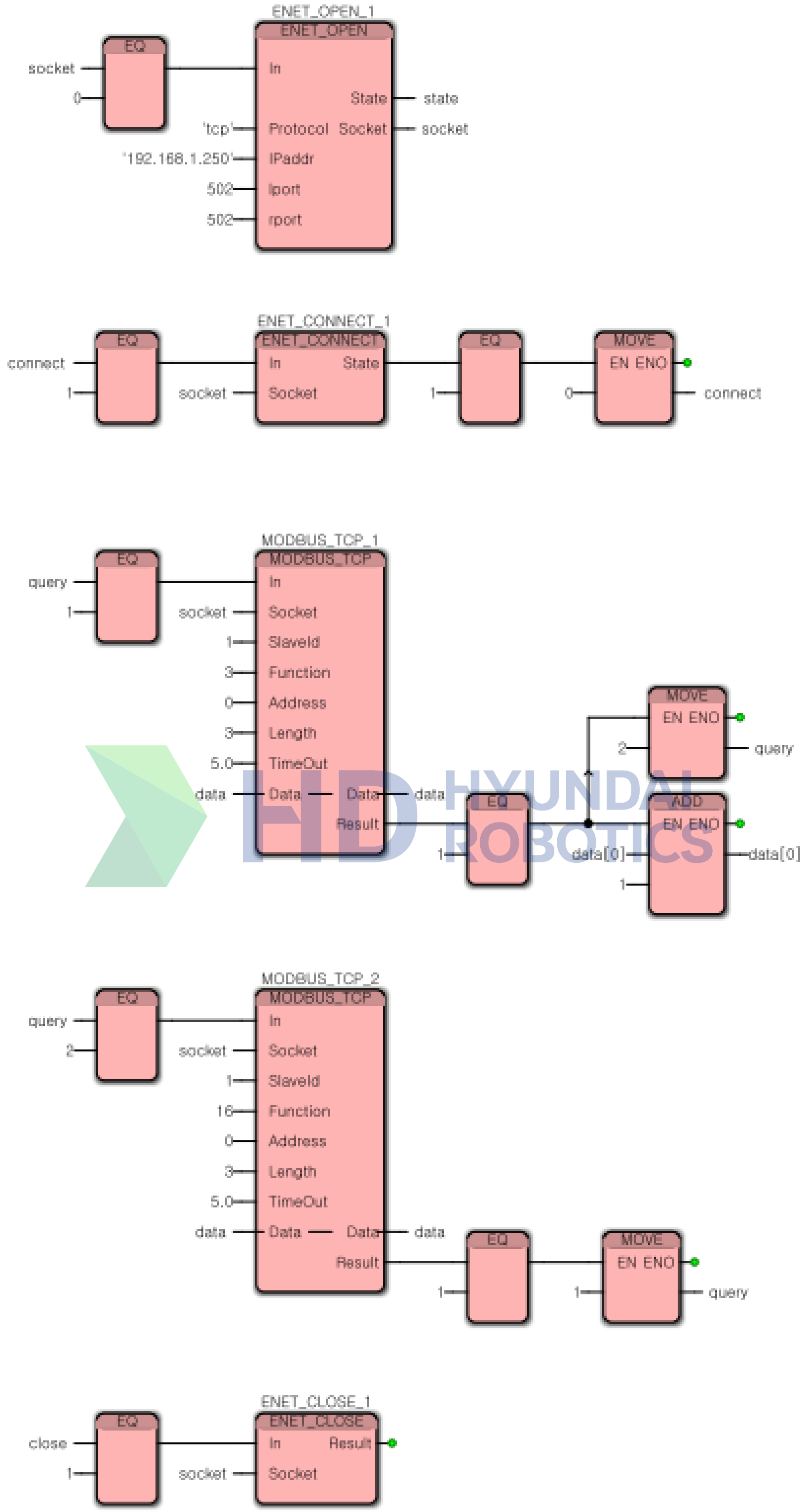


(ST language)



3.2.5. Sample function blocks

(FBD language)

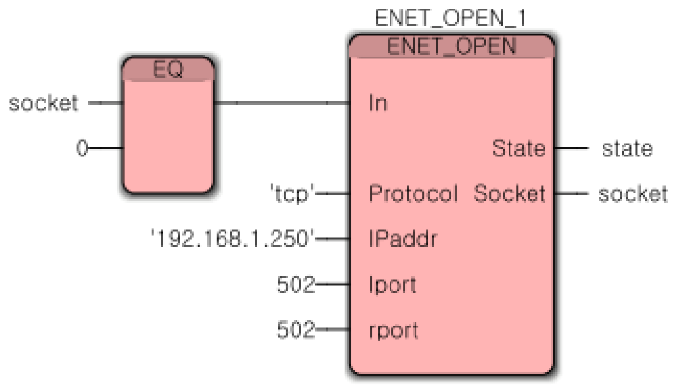


(ST language)

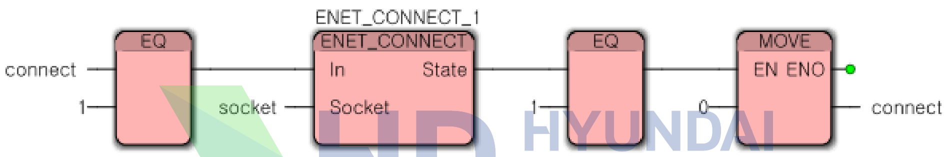
```
(* open *)
ENET_OPEN_1(In:=(socket = 0),Protocol:='tcp',IPaddr:='192.168.1.250',lport:=502,rport:=502);
socket := ENET_OPEN_1.Socket;
state := ENET_OPEN_1.State;

(* connect *)
ENET_CONNECT_1(In:=(connect = 1),Socket:=socket);
IF (ENET_CONNECT_1.State > 0) THEN
    connect := 0;
END_IF;


(* modbus communication *)
(* read *)
MODBUS_TCP_1(In:=(query = 1),Socket:=socket,SlaveId:=1,Function:=3,Address:=0,Length:=3,TimeOut:=3.0
,Data:=data);
data:=MODBUS_TCP_1.Data;
IF (MODBUS_TCP_1.Result > 0) THEN
    query := 2;
    data[0] := data[0] + 1;
END_IF;

(* write *)
MODBUS_TCP_2(In:=(query = 2),Socket:=socket,SlaveId:=1,Function:=16,Address:=0,Length:=3,TimeOut:=3.
0,Data:=data);
data:=MODBUS_TCP_2.Data;
IF (MODBUS_TCP_2.Result > 0) THEN
    query := 1;
END_IF;


(* close *)
ENET_CLOSE_1(In:=(close = 1),Socket:=socket);
```

3.2.6. Description of sample function blocks

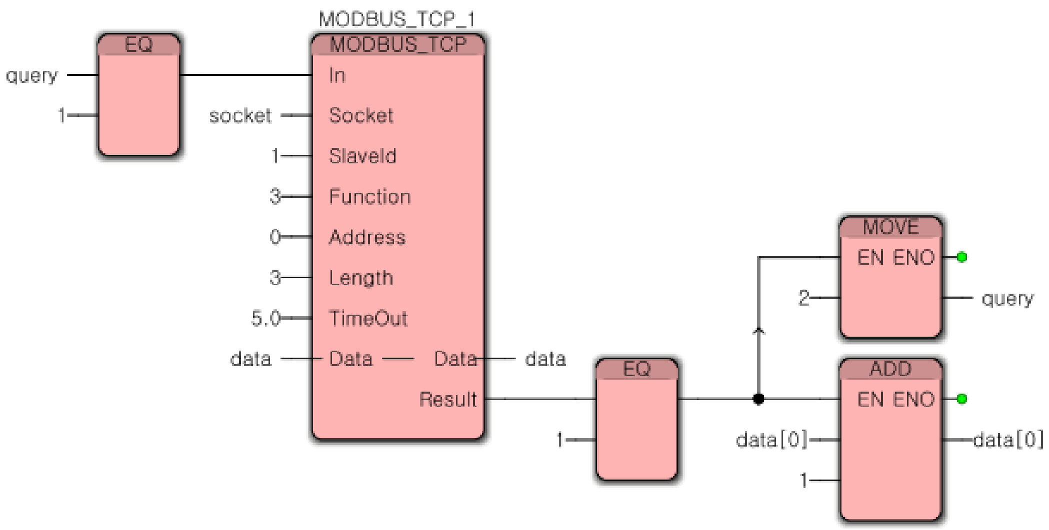
- Once the PLC is run, the socket for ENET communication will be automatically opened because the socket variable has been already initialized to "0."
- Regarding IPaddr, you must designate the IP address of the counterpart device that you need to connect to.



- In case the connect variable is set to "1," the operation of connection to the slave device will be performed, and then the connect variable will be changed to "0."

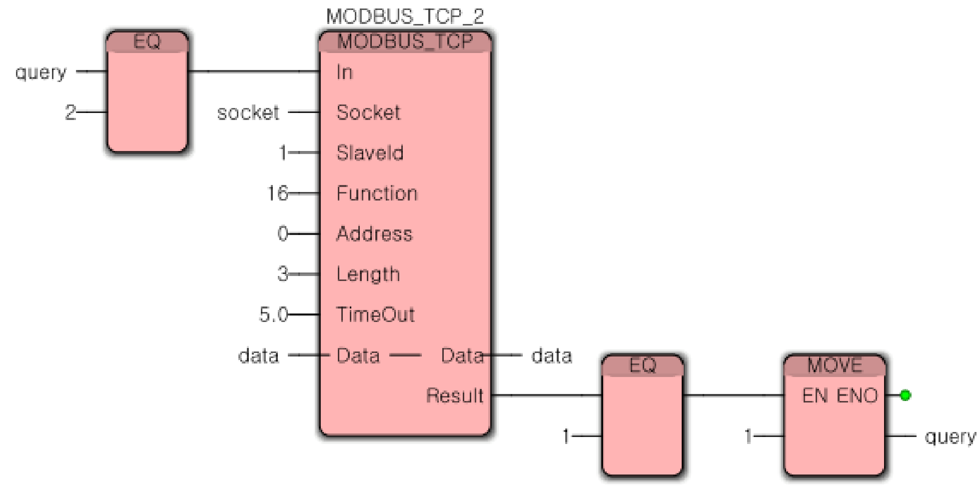


- In case the query variable is set to "1," 3 data results will be obtained from the address 0 of the slave in compliance with Function:=3, Address:=0, and Length:=3 and then transferred to the array variable of data (read).
- In case the result is "1," the query will be set to "2," and the variable value of data[0] will be increased by 1.

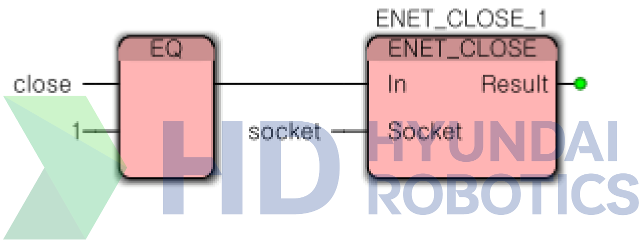


- In case the query variable is set to “2,” the value set in the data array variable will be set as the 3 data to the address 0 of the slave in compliance with Function:=16, Address:=0, and Length:=3 (write).

- In case the result is “1,” the query will be set to “1.”



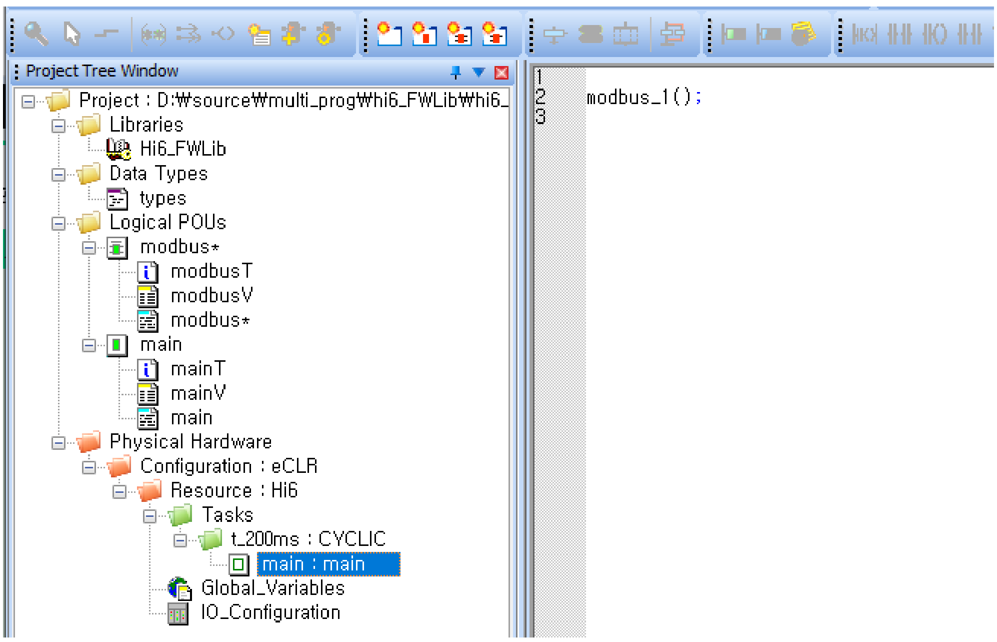
- In case the close variable is set to “1,” the socket will be closed.



3.2.7. Operation of sample function blocks

- The function blocks written as samples will be called and executed from main, which is a program POU.

- Main, a program POU, will be executed every 200 ms in the Cycle task.



Customer support

Contact: 1670-5041 | Email: robotics@hyundai-robotics.com

Operating hours: Weekdays (Monday–Friday) 09:00–18:00 | Closed on weekends and holidays

For any inquiries about our products or services, please contact our customer support team.

GRC: 477, Bundangsuseo-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13553, Korea

Daegu: 50, Techno sunhwan-ro 3-gil, Yuga-eup, Dalseong-gun, Daegu, Republic of Korea

Ulsan: Room 201-5, Automobile & Shipping Technology Hall, 21, Maegoksaneop-ro, Buk-gu, Ulsan, Republic of Korea

Joongbu: 161, Songgok-gil, Yeomchi-eup, Asan-si, Chungcheongnam-do, Republic of Korea

Gwangju: Room 101, Building B, 170-3, Pyeongdongsandan-ro, Gwangsan-gu, Gwangju, Republic of Korea

ARS 1588-9997 | 1 Robot Sales 2 Service Sales 3 Consultation for Purchase 4 Customer Support 5 Inquiry for Investment 6 Inquiries for Recruitment and General Matters

www.hyundai-robotics.com