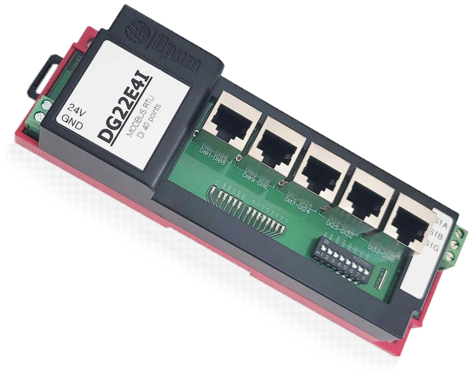
DG22E4I

KC certification

R-R-Diu-DGE4I

Manufacturer: Ilpum Corporation

Model name: DG22E4I



Operating environment

Normal operating temperature range = -25 ~ 70 [°C]

No dew, no dust.

Power

Rated voltage = DC 24 [V] (operating range 19 ~ 27 [V])

Maximum current consumption = 300 [mA]

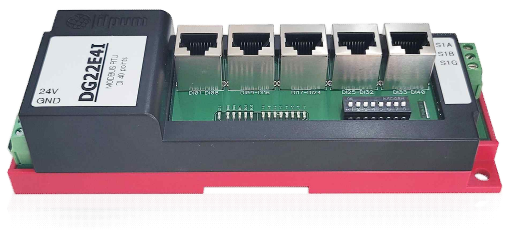
Communication

Physical standard: TIA/EIA-485A (RS485)

Maximum number of devices on the track = 64 node

ESD protection = up to 15 [kV]

Data protocol: MODBUS RTU protocol



Rating of DI terminal

Points can be connected without any external power

Have direct connection with external DO (includes power for DI detection inside the device)

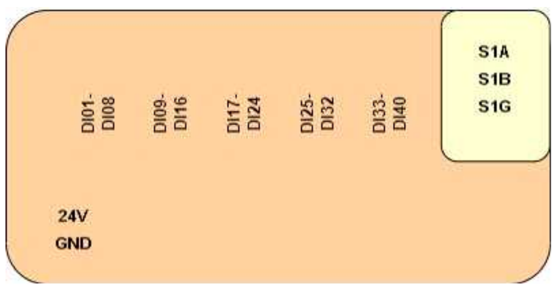
DI detection indication

LED lights up when ON is detected

Isolation

Isolation between power supply (power terminal and RS485 terminal) and all DI terminals

Maximum isolation voltage = 1.5 [kV rms] (50~60 [Hz], 1 [min])



[Figure 1] Internal isolation of E4I

Dimensions

Width 145 [mm], Depth 90 [mm], Height 41 [mm]

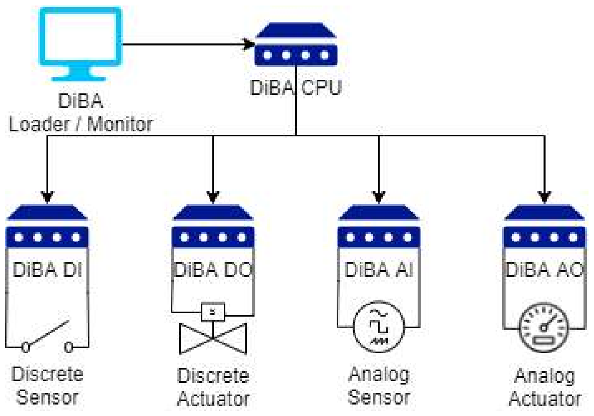
Fastening method

Can be mounted on DIN rail

Can be fixed with 4 screws on the wall (hole distance: x axis = 135 [mm], y axis = 70 [mm])

1. Summary

DG22E4I is a digital input module of PLC (Programmable Logic Controller). DiBA PLC composes an automatic control system with modules for each function as shown in [Figure 2], and the user can select the optimal module configuration according to the size and characteristics of the control object.



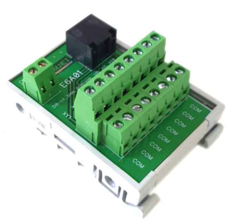
[Figure 2] Configuration of automatic control system

The product name of the model name DG22E4I is MODBUS RTU DI. The model name consists of the Ilpum Corporation mark (DG), the year of release, and the representative model name (E4I).

Since E4I operates only as a MODBUS RTU slave, it is controlled by a MODBUS RTU master such as E5A (DiBA PLC CPU module).

E4I is a module that monitors external devices and converts the status of other devices into information that can be read by the PLC system. Since the DI terminal of E4I uses internal power to detect a short circuit or an open circuit of an external circuit, only non-powered points can be used for the external circuit. That is, the status of another device is communicated using points (switches in general terms). The DI of E4I reads the short-circuit status as ON, and the open status reads it as OFF. The read information is displayed on the LED allocated for each DI. When it is ON, the LED turns on, and when it is OFF, the LED turns off. In addition to the basic function of DI, E4I counts the number of rising edges (the moment it changes from OFF to ON) at the maximum rate of 500 [PPS (Pulse Per Second)]. The count function can be used appropriately not only to simply check the current status using E4I, but also to check the number of status changes when it is necessary.

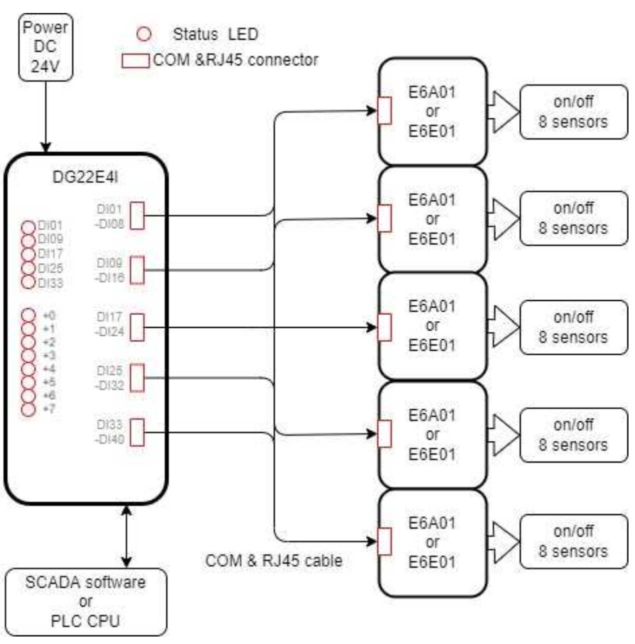
As the DI terminal of E4I uses RJ45, it is connected to a general external device through a terminal board (RJ45 uses a direct cable). By separating the terminal board, the number of terminals can be increased and a terminal board suitable for the various output characteristics of the DI terminal can be selected. This interface is protected by Patent No. 10-2214702. If you connect the E4I directly to the DO terminal of E4H (digital output module) or the DO terminal of E4J (integrated input/output module), you can use it by connecting only with an RJ45 direct cable without a terminal board. Currently, there are E6A01 and E6E01 terminal boards applicable to E4I.

Terminal board	E6A01	E6E01
Photograph		
Characteristic	Direct connection type 8 terminals	Individual isolated type 8 terminals, Outer power voltage 10~30[Vdc], No polarity

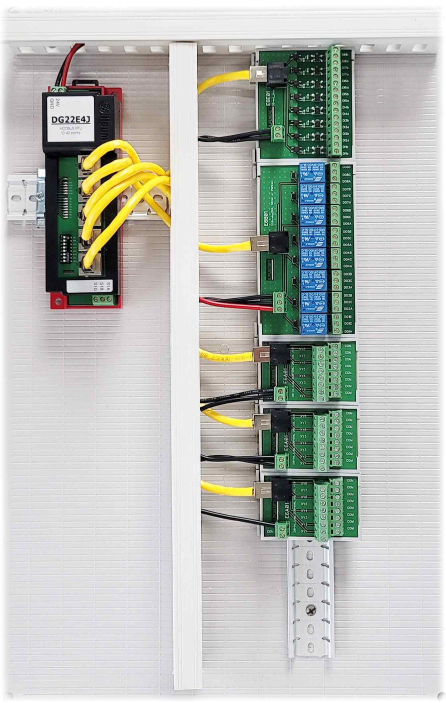
In order for the user of the automatic control system to safely control various devices, the E4I has an isolation design (refer to [Figure 1]). The inner area (isolated group 1) connected to the MODBUS RTU master contains power and RS485, and the outer area (isolated group 2) contains the entire DI.

2. Composition of the product

E4I can be connected to terminal board E6A01 (8 terminals for direct connection), E6E01 (Individual isolation type 8 terminals, Outer power voltage 10~30[Vdc], No polarity).



[Figure 3] How to connect the product



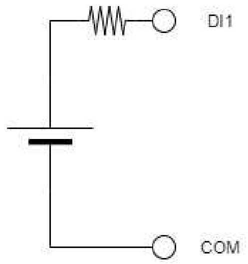
[Figure 4] Connection example picture (DG22E4J)

The status LED of E4I consists of 5 group LEDs and 8 terminal LEDs, and displays the status of terminals grouped by RJ45 modular jack unit. The 8 terminal LEDs from +0 to +7 indicate the status of the DI terminal of the group that is lit among the 5 LEDs indicating the group. That is, when DI01 LED is on, +0~+7 LEDs indicate the status of DI01~DI08 terminals, and when DI33 LED is on, +0~+7 LEDs indicate the status of DI33~DI40 terminals. The group LED lights up in sequence for 2 seconds and displays the status of each of the 8 terminals belonging to the RJ45 modular jack for 2 seconds, showing the status of all 40 DI terminals in turn for 10 seconds.

Terminal LED / Group LED	+0	+1	+2	+3	+4	+5	+6	+7
DI01	DI01	DI02	DI03	DI04	DI05	DI06	DI07	DI08
DI09	DI09	DI10	DI11	DI12	DI13	DI14	DI15	DI16
DI17	DI17	DI18	DI19	DI20	DI21	DI22	DI23	DI24
DI25	DI25	DI26	DI27	DI28	DI29	DI30	DI31	DI32
DI33	DI33	DI34	DI35	DI36	DI37	DI38	DI39	DI40

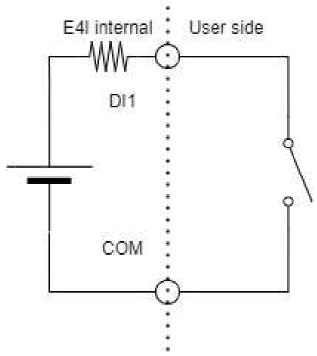
3. Circuit model and wiring

E4I has 40 DIs. [Figure 5] is an easy-to-understand representation of the internal circuit of DI01. The rest of DIs have the same shape. DI supply is isolated from the supply that drives the E4I.



[Figure 5] DI's internal circuit model

[Figure 6] is an example of connecting a switch to DI. The current state of [Figure 6] is that the switch is off (open), and DI01 of E4I is recognized as OFF. When the circuit is connected (shorted) by pressing the switch, the E4I recognizes DI01 as ON.



[Figure 6] Example of DI use

4. Operational Basics

All E4I information is mapped to the Holding Register area of MODBUS and cannot be accessed in other areas. It responds with an error when the MODBUS RTU master cannot process the request sent to E4I normally. The error response contains an error code, and the error codes used by E4I are:

Error code	Error name	Error content
1	Illegal Function	Function not supported
2	Illegal Address	Non-existent Register or write request to read-only
3	Illegal Value	Value outside the valid range

Operate E4I's Dip Switch to set the baudrate and Slave ID. Push the Dip Switch to the inside of the E4I main body to turn it ON, and push it to the outside of the E4I to turn it OFF. Baudrate can be set as follows:

Dip Switch: Baudrate1	Dip Switch: Baudrate0	Set baudrate [bps]	Common settings
OFF	OFF	9600	No Parity 8 Data Bits 1 Stop Bit
OFF	ON	19200	
ON	OFF	38400	
ON	ON	57600	

Slave ID of E4I is the same as the value read by Dip Switch in binary. If Dip Switch is ON, it is regarded as 1, if it is OFF, it is regarded as 0, and Slave ID is calculated by considering Address5 ~ Address0 as 2^5(=32) ~ 2^0(=1). Two examples are given below and summarized in a table. (2# is an indicator for binary notation) If Slave ID is set to 0, E4I does not respond at all.

(Example 1) Set Slave ID to 37. Address5 ~ Address0 = 2#100101

(Example 2) Set Slave ID to 1. Address5 ~ Address0 = 2#000001

Dip Switch name	Place value	(Example 1) 37 = 2#100101 $= 1x2^5+0x2^4+0x2^3+1x2^2+0x2^1+1x2^0$	(Example 2) 1 = 2#000001 $= 0x2^5+0x2^4+0x2^3+0x2^2+0x2^1+1x2^0$
Address5	2^5	1 = ON	0 = OFF
Address4	2^4	0 = OFF	0 = OFF
Address3	2^3	0 = OFF	0 = OFF
Address2	2^2	1 = ON	0 = OFF
Address1	2^1	0 = OFF	0 = OFF
Address0	2^0	1 = ON	1 = ON

5. MODBUS Protocol Memory Map

E4I provides only Holding Register as MODBUS slave. Holding Register is an area where both reading and writing are possible, but both reading and writing are impossible at addresses that E4I does not provide registers. Also, since there are addresses that can only be read, MODBUS master should access it referring to the table below. Addresses not shown in the table do not have a Register.

Adress	Read/ Write	Name	Value (= Meaning)
10	R	DI 01~16	0~65535 = bit mapped in the word Bit 0 (LSB): DI 1 {1 is ON, 0 is OFF} … Bit 15: DI 16 {1 is ON, 0 is OFF}
11	R	DI 17~32	0~65535 = bit mapped in the word Bit 0 (LSB): DI 17 {1 is ON, 0 is OFF} … Bit 15: DI 32 {1 is ON, 0 is OFF}
12	R	DI 33~40	0~255 = bit mapped in the word Bit 0 (LSB): DI 33 {1 is ON, 0 is OFF} … Bit 7: DI 40 {1 is ON, 0 is OFF}
100	R/W	DI 01 Counter	0~65535 = Number of occurrences of rising edge Up counter (increment by 1, 65535 cycles to 0) If you change the value, counts from that value.
101	R/W	DI 02 Counter	0~65535 = Number of occurrences of rising edge (same as above)
102	R/W	DI 03 Counter	0~65535 = Number of occurrences of rising edge (same as above)
103	R/W	DI 04 Counter	0~65535 = Number of occurrences of rising edge (same as above)
104	R/W	DI 05 Counter	0~65535 = Number of occurrences of rising edge (same as above)
105	R/W	DI 06 Counter	0~65535 = Number of occurrences of rising edge (same as above)
106	R/W	DI 07 Counter	0~65535 = Number of occurrences of rising edge (same as above)
107	R/W	DI 08 Counter	0~65535 = Number of occurrences of rising edge (same as above)
108	R/W	DI 09 Counter	0~65535 = Number of occurrences of rising edge (same as above)
109	R/W	DI 10 Counter	0~65535 = Number of occurrences of rising edge (same as above)
110	R/W	DI 11 Counter	0~65535 = Number of occurrences of rising edge (same as above)
111	R/W	DI 12 Counter	0~65535 = Number of occurrences of rising edge (same as above)
112	R/W	DI 13 Counter	0~65535 = Number of occurrences of rising edge (same as above)
113	R/W	DI 14 Counter	0~65535 = Number of occurrences of rising edge (same as above)
114	R/W	DI 15 Counter	0~65535 = Number of occurrences of rising edge (same as above)
115	R/W	DI 16 Counter	0~65535 = Number of occurrences of rising edge (same as above)
116	R/W	DI 17 Counter	0~65535 = Number of occurrences of rising edge (same as above)
117	R/W	DI 18 Counter	0~65535 = Number of occurrences of rising edge (same as above)

Adress	Read/ Write	Name	Value (= Meaning)
118	R/W	DI 19 Counter	0~65535 = Number of occurrences of rising edge (same as above)
119	R/W	DI 20 Counter	0~65535 = Number of occurrences of rising edge (same as above)
120	R/W	DI 21 Counter	0~65535 = Number of occurrences of rising edge (same as above)
121	R/W	DI 22 Counter	0~65535 = Number of occurrences of rising edge (same as above)
122	R/W	DI 23 Counter	0~65535 = Number of occurrences of rising edge (same as above)
123	R/W	DI 24 Counter	0~65535 = Number of occurrences of rising edge (same as above)
124	R/W	DI 25 Counter	0~65535 = Number of occurrences of rising edge (same as above)
125	R/W	DI 26 Counter	0~65535 = Number of occurrences of rising edge (same as above)
126	R/W	DI 27 Counter	0~65535 = Number of occurrences of rising edge (same as above)
127	R/W	DI 28 Counter	0~65535 = Number of occurrences of rising edge (same as above)
128	R/W	DI 29 Counter	0~65535 = Number of occurrences of rising edge (same as above)
129	R/W	DI 30 Counter	0~65535 = Number of occurrences of rising edge (same as above)
130	R/W	DI 31 Counter	0~65535 = Number of occurrences of rising edge (same as above)
131	R/W	DI 32 Counter	0~65535 = Number of occurrences of rising edge (same as above)
132	R/W	DI 33 Counter	0~65535 = Number of occurrences of rising edge (same as above)
133	R/W	DI 34 Counter	0~65535 = Number of occurrences of rising edge (same as above)
134	R/W	DI 35 Counter	0~65535 = Number of occurrences of rising edge (same as above)
135	R/W	DI 36 Counter	0~65535 = Number of occurrences of rising edge (same as above)
136	R/W	DI 37 Counter	0~65535 = Number of occurrences of rising edge (same as above)
137	R/W	DI 38 Counter	0~65535 = Number of occurrences of rising edge (same as above)
138	R/W	DI 39 Counter	0~65535 = Number of occurrences of rising edge (same as above)
139	R/W	DI 40 Counter	0~65535 = Number of occurrences of rising edge (same as above)
9000	R	Number of DIs available	40
9900	R	Design Year	2021
9901	R	Family Number	69
9902	R	Product Number	4
9903	R	Compatibility Number	73
9990	R	Version	1
9991	R	Lot	0~199

6. CPU module (E5A) usage example

E4I is operated under the control of MODBUS master. E5A (CPU module) of Ilpum Corporation can be set as MODBUS master, and has the ability to operate the system independently by storing user-specified tasks. Users can write DST file to instruct E5A what to do. For more details, refer to “E5A Manual” and “DST Reference Manual”.

The DST file below shows an example of how E5A operates some functions of E4I. E4I's Slave ID is 1, and a hand-pressed switch is connected to DI1 and DI2. By using text SCADA in the E5A manual, you can check the information of E4I, the pressed state of the switch, the number of times it is pressed, and delete the number of times it is pressed.

```
CONFIGURATION nameOfConf
  TYPE T_UDF:
    STRUCT
      length : BYTE;
      buffer : STRING;
    END_STRUCT
  END_TYPE

  VAR_GLOBAL
    gUsrSend, gUsrRecv : T_UDF;
    gNameIO : STRING := 'Unknown';
    gVerIO : INT := 0;
    gStaIO, gCntIO_1, gCntIO_2 : UINT := 0;
    gReadIO : ARRAY[4] OF UINT;
    gWriteIO, gIdxIO : UINT;
  END_VAR

  RESOURCE extLine1 ON ETH_1
    VAR_GLOBAL
      gConf : CONF_ETH1;
      gCommUsr : COMM_ETHUSR;
    END_VAR

    TASK taskInit (SINGLE := TRUE, PRIORITY := 1);
    TASK taskSync (INTERVAL := t#1s, PRIORITY := 5);

    PROGRAM pgm1Init WITH taskInit : Prog1Init();
    PROGRAM WITH taskSync : Prog1Sync();
  END_RESOURCE

  RESOURCE extLine2 ON SER_1
```

```
      VAR_GLOBAL
        gConf : CONF_SER1;
        gComm : COMM_SER1;
      END_VAR

      TASK taskInit (SINGLE := TRUE, PRIORITY := 2);
      TASK taskSync (INTERVAL := t#1s, PRIORITY := 6);

      PROGRAM pgm2Init WITH taskInit : Prog2Init();
      PROGRAM WITH taskSync : Prog2Sync();
    END_RESOURCE
END_CONFIGURATION

PROGRAM Prog1Init
    (* Communication settings: SET, user defined server mode, IPv4/23 *)
    extLine1.gConf(GET_SET := 1, MODE := 2, IPV4_6 := 0, PORT := 23);
    extLine1.gCommUsr.RECV_ADDR := ADDROF(gUsrRecv);
    extLine1.gCommUsr.RECV_LEN := SIZEOF(gUsrRecv.buffer) / 8;
    extLine1.gCommUsr(RECV_SEND := 0);
END_PROGRAM

PROGRAM Prog1Sync
    VAR
      vLen, vPosL, vPosR : INT;
      vStr0, vStr1 : STRING;
    END_VAR

    (* Command confirmation *)
    vLen := gUsrRecv.length;
    IF vLen < 3 THEN RETURN; END_IF;
    vStr0 := gUsrRecv.buffer;
    vPosL := FIND(vStr0, '$l');
    vPosR := FIND(vStr0, '$r');
    IF vLen < 10 THEN
      IF (vPosL < 0) & (vPosR < 0) THEN RETURN; END_IF;
    END_IF;

    vStr1 := vStr0;
    vStr0 := LEFT(vStr0, 1);
    vStr1 := DELETE(vStr1, 0, 1);
    vPosL := STR_TO_INT(vStr1);
    IF vStr0 = '?' THEN
```

```
      CASE vPosL OF
        0: (* Name and version output *)
          vStr1 := CONCAT(gNameIO, ', ', STR_FROM_INT(gVerIO));
        1: (* Status output of DI1 *)
          vStr1 := CONCAT('DI1 = ', STR_FROM_BOOL(gStaIO AND 1), ', Count = ',
STR_FROM_WORD(gCntIO_1));
        2: (* Status output of DI2 *)
          vStr1 := CONCAT('DI2 = ', STR_FROM_BOOL(gStaIO AND 2), ', Count = ',
STR_FROM_WORD(gCntIO_2));
        ELSE
          vStr1 := ': Invalid Command';
      END_CASE;
    ELSIF vStr0 = '!' THEN
      CASE vPosL OF
        10: (* DI 1 Counter Clear *)
          gWriteIO := 0;
          extLine2.gComm(READ_WRITE := 1, SLAVE_ADDR := 10, DATA_COUNT := 1,
MEM_ADDR := ADDROF(gWriteIO));
          vStr1 := 'Clear DI 1 Counter';
        20: (* DI 2 Counter Clear *)
          gWriteIO := 0;
          extLine2.gComm(READ_WRITE := 1, SLAVE_ADDR := 20, DATA_COUNT := 1,
MEM_ADDR := ADDROF(gWriteIO));
          vStr1 := 'Clear DI 2 Counter';
        ELSE
          vStr1 := ': Invalid Command';
      END_CASE;
    ELSE
      vStr1 := ': Invalid Command';
    END_IF;

    (* Communication request: SEND & RECV *)
    gUsrSend.buffer := CONCAT('$l$r', vStr1, '$l$r');
    extLine1.gCommUsr.SEND_ADDR := ADDROF(gUsrSend);
    extLine1.gCommUsr.SEND_LEN := LEN(gUsrSend.buffer);
    extLine1.gCommUsr(RECV_SEND := 2);
END_PROGRAM

PROGRAM Prog2Init
    (* Communication settings: SET, MODBUS RTU master mode, 9600/N/8/1 *)
    extLine2.gConf(GET_SET := 1, MODE := 1, RATE := 9600, PARITY := 0, DATABITS := 8,
STOPBIT := 1);
```

```
  (* Communication target: Slave ID = 1, Slave Area = holding register *)
  extLine2.gComm.SLAVE_ID := 1;
  extLine2.gComm.SLAVE_AREA := 3;
  extLine2.gComm.EC := 2; (* Designate as abnormal termination for initialization. *)
END_PROGRAM

PROGRAM Prog2Sync
  VAR
    vLen, vPosL, vPosR : INT;
  END_VAR

  (* Wait for communication result. *)
  IF extLine2.gComm.EC = 1 THEN RETURN; END_IF;

  IF extLine2.gComm.EC <> 0 THEN (* Communication abnormal termination *)
    gIdxIO := 0;
    gNameIO := 'Unknown';
    gVerIO := 0;
    gStaIO := 0;
    gCntIO_1 := 0;
    gCntIO_2 := 0;
  END_IF;

  CASE gIdxIO OF
    0: (* Design Year ~ Compatibility Number *)
      extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 9900, DATA_COUNT := 4,
MEM_ADDR := ADDROF(gReadIO[0]));
    1: (* Version *)
      IF (gReadIO[0] <> 2016) | (gReadIO[1] <> 69) | (gReadIO[2] <> 4) | (gReadIO[3] <> 65) THEN
        extLine2.gComm.EC := 2;
        RETURN;
      END_IF;
      gNameIO := 'DG16E4A';
      extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 9990, DATA_COUNT := 1,
MEM_ADDR := ADDROF(gVerIO));
  ELSE
      IF gVerIO <> 2 THEN
        extLine2.gComm.EC := 2;
        RETURN;
      END_IF;
      (* DI 1~16 *)
      extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 0, DATA_COUNT := 1, MEM_ADDR
```

```
:= ADDROF(gStaIO));
        (* DI1 Counter *)
        extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 10, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gCntIO_1));
        (* DI2 Counter *)
        extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 20, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gCntIO_2));
  END_CASE;
  IF gIdxIO < 2 THEN
    gIdxIO := gIdxIO + 1;
  END_IF;
END_PROGRAM
```

The available commands are:

?0	Print the product name and version.
?1	Outputs the status of DI1.
?2	Outputs the status of DI2.
!10	Clear Counter of DI1.
!20	Clear Counter of DI2.