DG21E4H

KC certification

R-R-Diu-DGE4H

Manufacturer: Ilpoom Co., Ltd., Country of manufacture: Korea

Model name: DG21E4H

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 DO terminal

Transistor output: Sink type

Electric current: 0~0.5 [A], Voltage: 0~50 [V]

DO operation indication

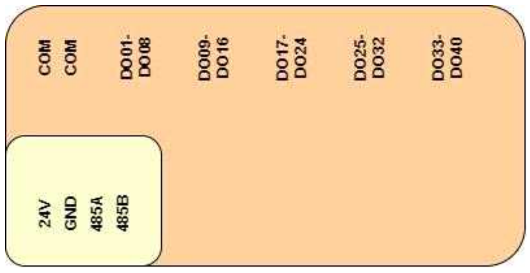
Setting OFF: LED OFF, contact open

Setting ON: LED ON, contact short circuit

Isolation

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

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



[Figure 1] Internal isolation of E4H

Dimensions

145 × 90 × 41 [mm]

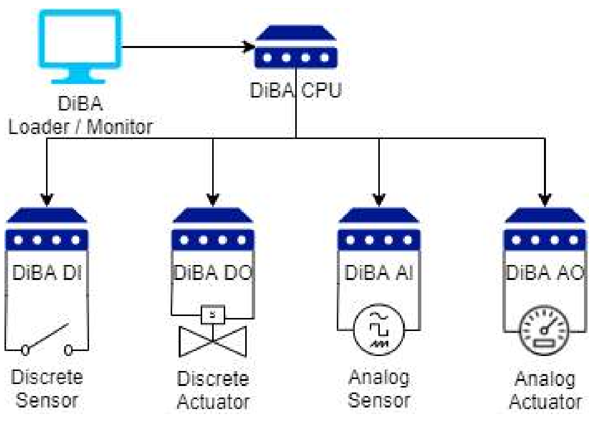
Fastening method

Can be mounted on DIN rail

Can be fixed with 4 screws (width 135 [mm], length 70 [mm])

1. Summary

DG21E4H is a digital output 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.
The product name of the model name DG21E4H is MODBUS RTU DO. The model name consists of the Ilpum company mark (DG), the year of release, and the representative model name (E4H).

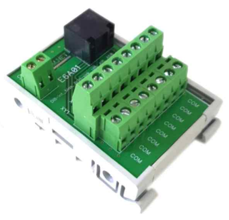
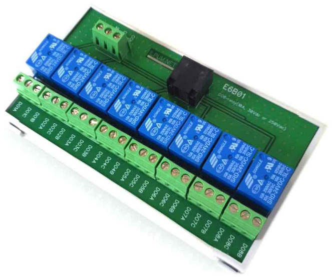


[Figure 2] Configuration of automatic control system

Since E4H operates only as a MODBUS RTU slave, it is controlled by a MODBUS RTU master such as E5A (DiBA PLC CPU module).

E4H is a module that controls external devices and links the control information of the PLC system with the operation of the external devices. The DO terminal of E4H is a switch that connects or disconnects the driving power of an external device. The range of DC drive power that E4H can control is 50 [V] or less voltage and 0.5 [A] or less of current. Since the DO terminal of E4H is a sink output using a transistor, AC drive power cannot be used. A negative (–) voltage must be applied to the COM terminal and a positive (+) voltage to the DO terminal. If the control information of the DO terminal is 1 (ON), the driving power is connected and the corresponding LED is turned on. If the control information of the DO terminal is 0 (OFF), the driving power is cut off and the corresponding LED is turned off.

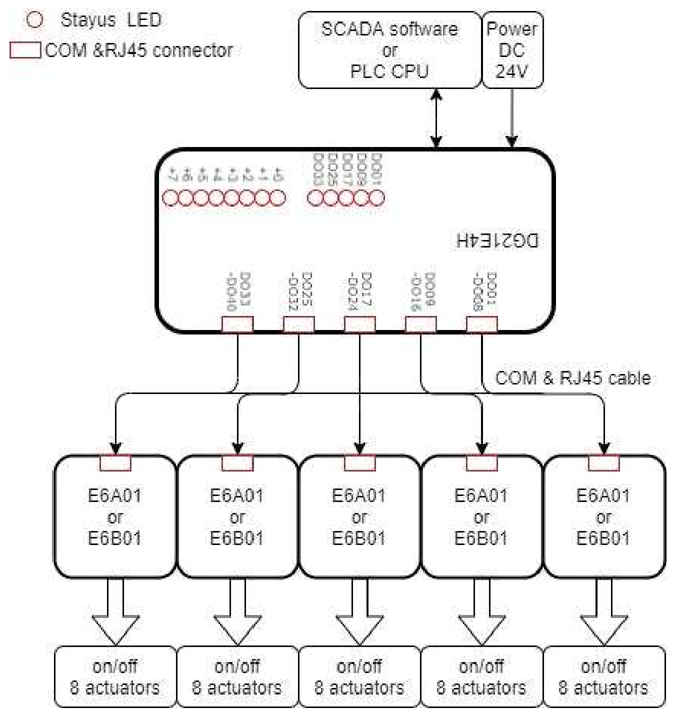
Since the DO terminals of E4H use RJ45, they are connected to general external devices through terminal boards (RJ45 uses a direct cable). By separating the terminal boards, the number of terminals can be increased and terminal boards suitable for the various output characteristics of the DO terminals can be selected. This interface is protected by Patent No. 10-2214702. If you connect the E4H directly to the DI terminal of E4I (digital input module) or the DI or UI 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 E6B01 terminal boards applicable to E4H.

Terminal board	E6A01	E6B01
Photograph		
Characteristic	Direct connection type 8 terminals	Relay type 8 terminals, Max. 10[A], Max. 30[Vdc] or 250[Vac]

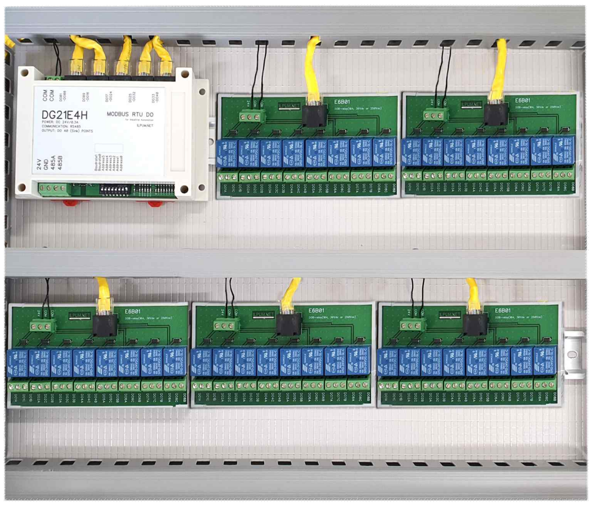
In order for the user of the automatic control system to safely control various devices, the E4H has an isolation design (refer to [Figure 1]). The inner area (isolation group 1) connected to the MODBUS RTU master contains power and RS485, and the outer area (isolation group 2) contains the entire DO.

2. Composition of the product

E4H can be connected with the terminal board E6A01 (8 terminals for direct connection), E6B01 (8 terminals for relay type, maximum 10 [A], maximum 30 [Vdc] or 250 [Vac]).



[Figure 3] How to connect the product



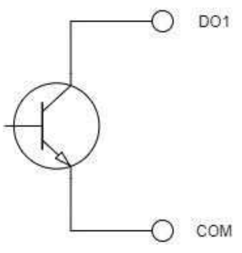
[Figure 4] Connection example picture

The status LED of E4H 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 DO terminals of the group that is lit among the 5 LEDs indicating the group. That is, when DO01 LED is on, +0~+7 LEDs indicate the status of DO01~DO08 terminals, and when DO33 LED is on, +0~+7 LEDs indicate the status of DO33~DO40 terminals. The group LEDs turn on in sequence for 2 seconds to indicate the status of the 8 terminals belonging to the RJ45 modular jack for 2 seconds each, and the status of all 40 DO terminals is shown in turn for 10 seconds.

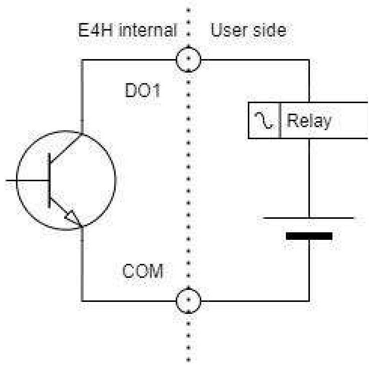
Terminal LED / Group LED	+0	+1	+2	+3	+4	+5	+6	+7
DO01	DO01	DO02	DO03	DO04	DO05	DO06	DO07	DO08
DO09	DO09	DO10	DO11	DO12	DO13	DO14	DO15	DO16
DO17	DO17	DO18	DO19	DO20	DO21	DO22	DO23	DO24
DO25	DO25	DO26	DO27	DO28	DO29	DO30	DO31	DO32
DO33	DO33	DO34	DO35	DO36	DO37	DO38	DO39	DO40

3. Circuit model and wiring

E4H has 40 DOs. [Figure 5] is an easy-to-understand representation of DO01's internal circuit. The rest of the DOs have the same shape. When connecting an external device using terminal board E6A01, DO01 is placed on the XY1 terminal.
In [Figure 5], DO01 and COM terminals are open (disconnected) or shorted (connected) depending on the setting value. When the power of E4H is initially supplied, the DO state is OFF, and DO01 and COM terminals are open (power-on default).
[Figure 6] shows an example of using an external relay.

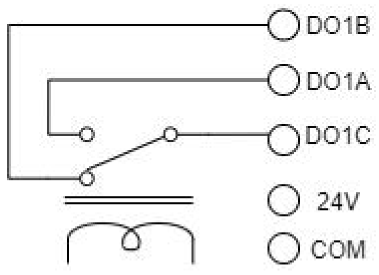


[Figure 5] DO's internal circuit model

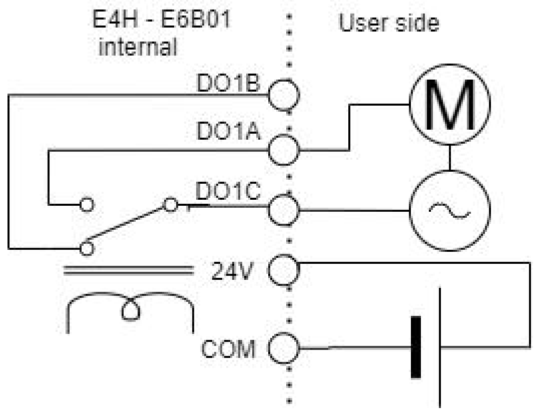


[Figure 6] Example of DO use

The circuit model when the terminal board E6B01 is used is [Figure 7], and the example of use is [Figure 8]. All terminals are individually isolated by a 10[A] relay driven by 24[Vdc].



[Figure 7] DO internal circuit model when E6B01 is applied



[Figure 8] Example of DO use when E6B01 is applied

4. Operational Basics

All E4H 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 E4H normally. The error response contains an error code, and the error code used by E4H is:

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

Operate the E4H Dip Switch to set the baudrate and Slave ID. Push the Dip Switch to the inside of the E4H main body to turn ON, and push it to the outside of the E4H 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 E4H 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, E4H 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 = $1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	(Example 2) 1 = 2#000001 = $0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
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

E4H 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 E4H does not provide registers. Also, since there are addresses that can only be read, the MODBUS master should access it referring to the table below. Registers do not exist for addresses not shown in the table.

Address	Read/Write	Name	Value(= Meaning)

0	R/W	DO 01~16	0~65535 = bit mapped in the word
			Bit 0 (LSB): DO 01 {1 is ON, 0 is OFF}
			…
			Bit 15: DO 16 {1 is ON, 0 is OFF}

1	R/W	DO 17~32	0~65535 = bit mapped in the word
			Bit 0 (LSB): DO 17 {1 is ON, 0 is OFF}
			…
			Bit 15: DO 32 {1 is ON, 0 is OFF}

2	R/W	DO 33~40	0~255 = bit mapped in the word
			Bit 0 (LSB): DO 33 {1 is ON, 0 is OFF}
			…
			Bit 7: DO 40 {1 is ON, 0 is OFF}

9000	R	Number of available DOs	40
9900	R	Design Year	2021
9901	R	Family Number	69
9902	R	Product Number	4
9903	R	Compatibility number	72
9990	R	Version	1
9991	R	Lot	0~199

6. CPU module (E5A) usage example

E4H 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 information, refer to the “E5A Manual” and the “DST Reference Manual”.

The DST file below shows an example of how E5A operates some functions of E4H. E4H's Slave ID is 1. Do not connect anything to the DO terminal, and use the text SCADA of the E5A manual to change the DO setting of E4H. The operation of the E4H can be checked through the LED.

```
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 : 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;
   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 := MID(vStr0, 1, -1);
   vStr0 := LEFT(vStr0, 1);
   vPosL := STR_TO_INT(vStr1);
   IF vStr0 = '?' THEN
      CASE vPosL OF
      0: (* Name and version output *)
         vStr0 := CONCAT(gNameIO, ', ', STR_FROM_INT(gVerIO));
      1: (* Status output of DO1 *)
         vStr0 := CONCAT('DO1 = ', STR_FROM_BOOL(gStaIO AND 1));
      2: (* Status output of DO2 *)
        vStr0 := CONCAT('DO2 = ', STR_FROM_BOOL(gStaIO AND 2));
      3: (* Status output of DO3 *)
        vStr0 := CONCAT('DO3 = ', STR_FROM_BOOL(gStaIO AND 4));
      4: (* Status output of DO4 *)
        vStr0 := CONCAT('DO4 = ', STR_FROM_BOOL(gStaIO AND 8));
      5: (* Status output of DO5 *)
        vStr0 := CONCAT('DO5 = ', STR_FROM_BOOL(gStaIO AND 16#10));
      6: (* Status output of DO6 *)
        vStr0 := CONCAT('DO6 = ', STR_FROM_BOOL(gStaIO AND 16#20));
      7: (* Status output of DO7 *)
        vStr0 := CONCAT('DO7 = ', STR_FROM_BOOL(gStaIO AND 16#40));
      8: (* Status output of DO8 *)
        vStr0 := CONCAT('DO8 = ', STR_FROM_BOOL(gStaIO AND 16#80));
      END_CASE;
    ELSIF vStr0 = '!' THEN
      vPosR := vPosL MODULO 10;
      vPosL := vPosL / 10;
      IF vPosL > 0 & vPosL <= 8 THEN (* DO setting *)
        vLen := SHL(1, vPosL - 1);
        IF vPosR = 0 THEN
          gWriteIO := gStaIO AND (NOT vLen);
        ELSIF vPosR = 1 THEN
          gWriteIO := gStaIO OR vLen;
        END_IF;
        extLine2.gComm(READ_WRITE := 1, SLAVE_ADDR := 0, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gWriteIO));
        vStr0 := CONCAT('Set DO ', STR_FROM_INT(vPosL));
      END_IF;
    END_IF;

    IF LEN(vStr0) < 2 THEN
      vStr0 := ': Invalid Command';
    END_IF;

    (* Communication request: SEND & RECV *)
    gUsrSend.buffer := CONCAT('$l$r', vStr0, '$l$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;
  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] <> 69) THEN
      extLine2.gComm.EC := 2;
      RETURN;
    END_IF;
    gNameIO := 'DG16E4E';
    extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 9990, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gVerIO));
  ELSE
      IF gVerIO <> 1 THEN
        extLine2.gComm.EC := 2;
        RETURN;
      END_IF;
      (* DO 1~8 *)
      extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 0, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gStaIO));
  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 DO1.
?2	Outputs the status of DO2.
?3	Outputs the status of DO3.
?4	Outputs the status of DO4.
?5	Outputs the status of DO5.
?6	Outputs the status of DO6.
?7	Outputs the status of DO7.
?8	Outputs the status of DO8.
!10	Set DO1 to OFF.
!11	Set DO1 to ON.
!20	Set DO2 to OFF.
!21	Set DO2 to ON.
!30	Set DO3 to OFF.
!31	Set DO3 to ON.
!40	Set DO4 to OFF.
!41	Set DO4 to ON.
!50	Set DO5 to OFF.
!51	Set DO5 to ON.
!60	Set DO6 to OFF.
!61	Set DO6 to ON.
!70	Set DO7 to OFF.
!71	Set DO7 to ON.
!80	Set DO8 to OFF.
!81	Set DO8 to ON.