DG21E4H

KC인증

R-R-Diu-DGE4H
상호명: 일품 주식회사, 제조자: 일품 주식회사, 제조국: 한국
모델명: DG21E4H

사용 환경

정상 동작 온도 범위 = -25 ~ 70 [°C]
이슬이 맺히지 않을 것, 먼지가 없을 것.

전원

정격 전압 = DC 24 [V] (동작 가능 범위 19 ~ 27 [V])
최대 소모 전류 = 300 [mA]

통신

물리 규격: TIA/EIA-485A (RS485)
선로상 최대 장치 수 = 64 node
ESD 보호 = 15 [kV]까지
데이터 프로토콜: MODBUS RTU protocol

DO 단자의 정격

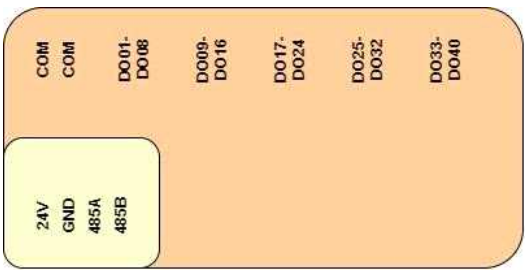
Transistor 출력: Sink type
전류: 0~0.5 [A], 전압: 0~50 [V]

DO 작동 표시

설정 OFF: LED OFF, 접점 개방
설정 ON: LED ON, 접점 단락

격리(Isolation)

전원(전원 단자와 RS485 단자)과 모든 DO 단자 사이의 격리
최대 격리 전압 = 1.5 [kV rms] (50~60 [Hz], 1 [분])



[그림1] E4H의 내부 격리

외형 치수

가로 145 [mm], 세로 90 [mm], 높이 41 [mm]

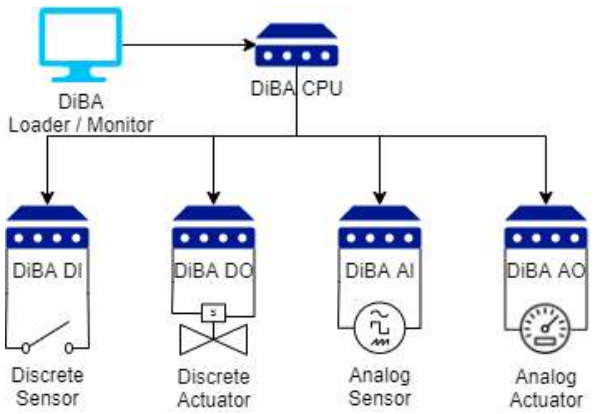
고정 방식

DIN rail에 장착 가능
나사 4개로 고정 가능 (가로 135 [mm], 세로 70 [mm])

1. 개요

DG21E4H는 PLC(Programmable Logic Controller)의 디지털 출력(Digital Output) 모듈입니다. DiBA(다이바) PLC는 [그림2]처럼 기능별 모듈들로 자동제어시스템을 구성하며, 사용자는 제어 대상의 크기와 특성에 따라 최적의 모듈 구성을 선택할 수 있습니다.

모델명 DG21E4H의 제품명은 MODBUS RTU DO입니다. 모델명은 일품표식(DG)과 출시연도, 대표모델명(E4H)으로 구성됩니다.

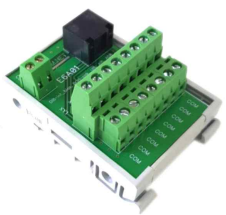
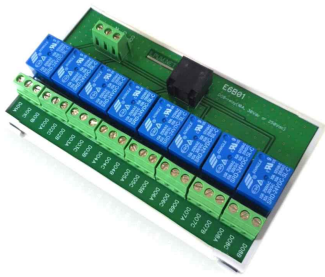


[그림2] 자동제어시스템의 구성

E4H는 MODBUS RTU slave로만 동작하므로 E5A(DiBA PLC CPU 모듈) 등의 MODBUS RTU master에 의해 제어됩니다.

E4H는 외부 장치를 제어하는 모듈로 PLC 시스템의 제어 정보와 외부 장치의 구동을 연계시킵니다. E4H의 DO 단자는 외부 장치의 구동 전원을 연결하거나(connect) 끊는(disconnect) 스위치입니다. E4H가 제어할 수 있는 직류 구동 전원의 범위는 전압 50[V] 이하이고, 전류 0.5[A] 이하입니다. E4H의 DO단자는 트랜지스터를 사용한 sink 출력이므로 교류 구동 전원은 사용할 수 없으며, COM 단자는 음(-)의 전압을 DO 단자는 양(+)의 전압을 인가해야 합니다. DO 단자의 제어 정보가 1(ON)이면 구동 전원을 연결하며 해당 LED가 켜지고, DO 단자의 제어 정보가 0(OFF)이면 구동 전원을 끊으며 해당 LED가 꺼집니다.

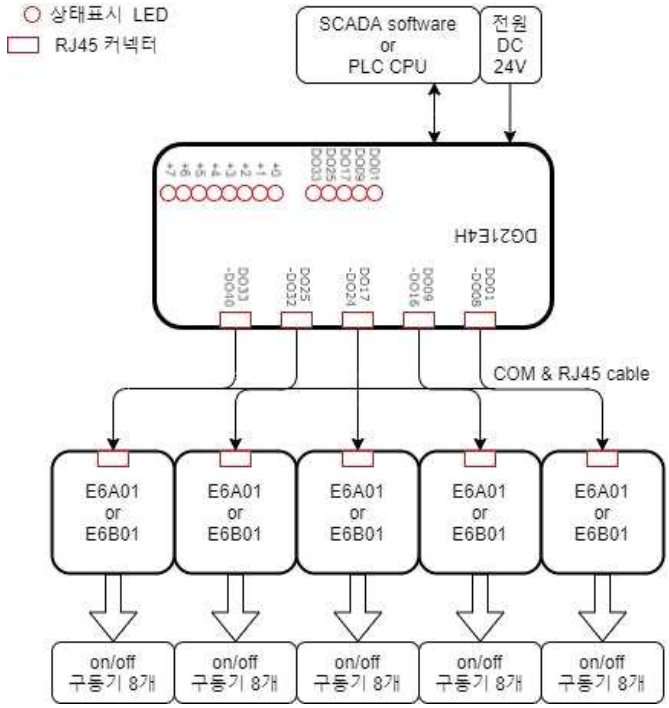
E4H의 DO 단자는 RJ45를 사용하므로 일반적인 외부 장치와는 터미널보드(RJ45는 direct cable 사용)를 통해 연결합니다. 터미널보드를 분리함으로써 단자의 수를 늘리고, DO 단자의 다양한 출력 특성에 맞는 터미널보드를 선택할 수 있습니다. 이 인터페이스는 특허 제10-2214702호로 보호받고 있습니다. 만약 E4H를 E4I(디지털 입력 모듈)의 DI 단자나 E4J(통합 입출력 모듈)의 DI 혹은 UI 단자와 바로 연결한다면, 터미널보드 없이 RJ45 direct cable만으로 연결하여 사용할 수 있습니다. 현재 E4H에 적용할 수 있는 터미널보드는 E6A01, E6B01이 있습니다.

터미널보드	E6A01	E6B01
사진		
특성	직결형 8단자	Relay형 8단자, 최대 10[A], 최대 30[Vdc] or 250[Vac]

자동제어시스템의 사용자가 안전하게 다양한 장치들을 제어하도록 E4H는 격리(isolation) 설계가 적용되어 있습니다([그림1] 참고). MODBUS RTU master와 연결되는 내부 영역(격리군1)은 전원과 RS485를 포함하고, 외부 영역(격리군2)은 DO 전체를 포함합니다.

2. 제품의 구성 및 연결 방법

E4H는 터미널보드 E6A01(직결형 8단자), E6B01(Relay형 8단자, 최대 10[A], 최대 30[Vdc] or 250[Vac])과 연결할 수 있습니다.



[그림3] 구성 및 연결 방법

[그림4] 연결 예시 사진

E4H의 상태표시 LED는 5개의 그룹 LED와 8개의 단자 LED로 구성되어, 단자들의 상태를 RJ45 모듈러잭 단위로 묶어서 표시합니다. +0~+7의 8개의 단자 LED는 그룹을 나타내는 5개의 LED 중 점등되는 그룹의 DO 단자의 상태를 나타냅니다. 즉, DO01 LED가 켜지면 +0~+7의 LED는 DO01~DO08 단자의 상태를 나타내고, DO33 LED가 켜지면 +0~+7의 LED는 DO33~DO40 단자의 상태를 나타냅니다. 그룹 LED가 2초씩 순서대로 켜져서 RJ45 모듈러잭 단위로 소속된 단자들(8개)의 상태를 각각 2초간 표시하여, 40개의 모든 DO 단자의 상태를 10초간 돌아가며 보입니다.

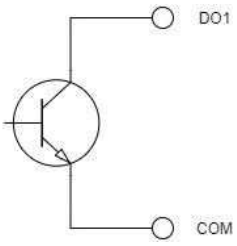
단자 LED / 그룹 LED	+0	+1	+2	+3	+4	+5	+6	+7
DO01	DO01	DO02	DO03	DO04	DO05	DO06	DO07	DO08
DO09	DO09	DO10	DO11	DO12	DO13	DO14	DO15	DO16
DO17	DO17	DO18	DO19	DO20	DO21	DO22	DO23	DO24
DO25	DO25	DO26	DO27	DO28	DO29	DO30	DO31	DO32
DO33	DO33	DO34	DO35	DO36	DO37	DO38	DO39	DO40

3. 회로 모델 및 배선

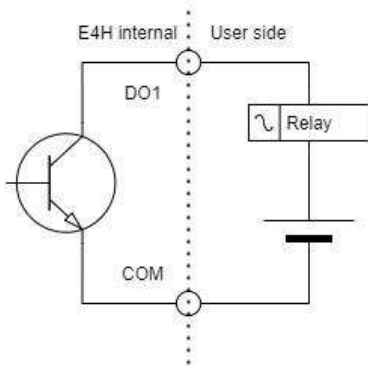
E4H는 40개의 DO를 가지고 있습니다. [그림5]는 그 중 DO01의 내부 회로를 이해하기 쉽게 표현한 것입니다. 나머지 DO도 동일한 형태를 가집니다. 터미널보드로 E6A01을 이용하여 외부 장치를 연결하는 경우, DO01은 XY1 단자에 배치됩니다.

[그림5]에서 DO01과 COM 단자는 설정값에 따라 개방(open, 끊음)되거나 단락(short, 연결)됩니다. E4H의 전원이 공급된 초기에 갖는 DO의 상태는 OFF이고, DO01과 COM 단자는 개방(open)되어 있습니다(power-on default).

[그림6]은 외부에 Relay를 장착한 사용 예를 보입니다.

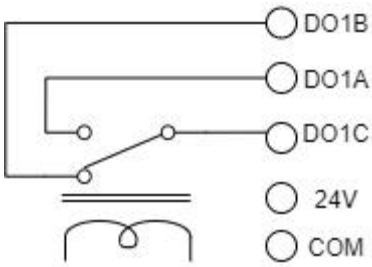


[그림5] DO의 내부 회로 모델

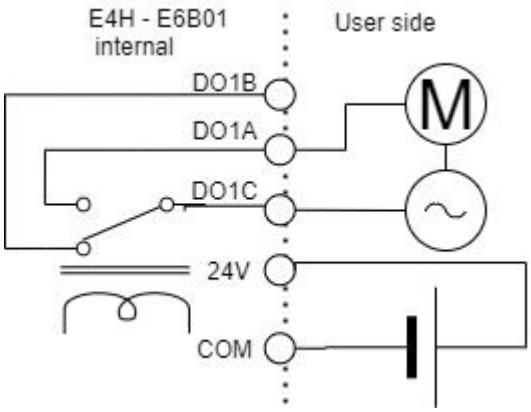


[그림6] DO 사용 예

터미널보드로 E6B01을 이용한 경우의 회로 모델은 [그림7]이고, 사용 예는 [그림8]입니다. 24[Vdc]로 구동되는 10[A] Relay에 의해 모든 단자가 개별 격리됩니다.



[그림7] E6B01 적용시 DO의 내부 회로 모델



[그림8] E6B01 적용시 DO 사용 예

4. 운용 기본 사항

E4H의 정보는 모두 MODBUS의 Holding Register 영역에 mapping되어 있고, 다른 영역으로는 접근할 수 없습니다. MODBUS RTU master가 E4H에게 보낸 요청을 정상적으로 처리할 수 없는 경우에 오류 응답을 합니다. 오류 응답에는 오류 코드가 포함되며, E4H가 사용하는 오류 코드는 다음과 같습니다.

오류 코드	오류 이름	오류 내용
1	Illegal Function	지원하지 않는 Function
2	Illegal Address	존재하지 않는 Register 혹은 읽기 전용에 대한 쓰기 요청
3	Illegal Value	유효 범위 밖의 값

E4H의 Dip Switch를 조작하여 통신 속도(baudrate)와 Slave ID를 설정합니다. Dip Switch를 E4H 본체 안쪽으로 밀면 ON, E4H 바깥쪽으로 밀면 OFF입니다. Baudrate은 다음과 같이 설정할 수 있습니다.

Dip Switch: Baudrate1	Dip Switch: Baudrate0	설정된 baudrate [bps]	공통 설정
OFF	OFF	9600	No Parity 8 Data Bits 1 Stop Bit
OFF	ON	19200	
ON	OFF	38400	
ON	ON	57600	

E4H의 Slave ID는 Dip Switch를 2진수로 읽은 값과 같습니다. Dip Switch가 ON이면 1, OFF면 0으로 보고, Address5 ~ Address0을 2^5(=32) ~ 2^0(=1)로 봐서 Slave ID를 계산합니다. 아래에 2가지 예를 들고, 표에 정리합니다. (2#은 2진수 표기에 대한 지시자입니다) Slave ID를 0으로 설정하면 E4H는 어떤 응답도 하지 않습니다.

(예1) Slave ID를 37로 설정하기. Address5 ~ Address0 = 2#100101

(예2) Slave ID를 1로 설정하기. Address5 ~ Address0 = 2#000001

Dip Switch 이름	자리의 값	(예1) 37 = 2#100101 = 1 x 2^5 + 0 x 2^4 + 0 x 2^3 + 1 x 2^2 + 0 x 2^1 + 1 x 2^0	(예2) 1 = 2#000001 = 0 x 2^5 + 0 x 2^4 + 0 x 2^3 + 0 x 2^2 + 0 x 2^1 + 1 x 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

E4H는 MODBUS slave로서 Holding Register만 제공합니다. Holding Register는 읽기와 쓰기가 모두 가능한 영역이지만, E4H가 Register를 제공하지 않는 주소에서는 읽기와 쓰기가 모두 불가능합니다. 또한, 읽기만 가능한 주소도 있으므로 MODBUS master는 아래의 표를 참고하여 접근해야 합니다. 표에 나타나지 않은 주소는 Register가 존재하지 않습니다.

주소	Read/Write	이름	값(= 의미)
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	사용 가능한 DO 수	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 모듈(E5A) 사용 예

E4H는 MODBUS master의 제어를 받아서 운용됩니다. 일품의 E5A(CPU 모듈)는 MODBUS master로 설정할 수 있고, 사용자가 지정한 작업을 저장하여 단독으로 시스템을 운용하는 능력을 가지고 있습니다. 사용자는 DST file을 작성하여 E5A가 작업할 내용을 지시할 수 있습니다. 더 자세한 내용은 「E5A 설명서」와 「DST 문법 설명서」를 참고하십시오.

아래의 DST file은 E4H의 몇몇 기능을 E5A가 운용하는 예를 보입니다. E4H의 Slave ID는 1입니다. DO 단자에는 아무것도 연결하지 않고, E5A 설명서의 text SCADA를 이용하여 E4H의 DO 설정을 변경합니다. E4H의 동작은 LED를 통해 확인할 수 있습니다.

```
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
  (* 통신 설정: 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

  (* 명령 확인 *)
  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: (* 이름 및 버전 출력 *)
      vStr0 := CONCAT(gNameIO, ', ', STR_FROM_INT(gVerIO));
    1: (* DO1 상태 출력 *)
      vStr0 := CONCAT('DO1 = ', STR_FROM_BOOL(gStaIO AND 1));
```

```
      2: (* DO2 상태 출력 *)
        vStr0 := CONCAT('DO2 = ', STR_FROM_BOOL(gStaIO AND 2));
      3: (* DO3 상태 출력 *)
        vStr0 := CONCAT('DO3 = ', STR_FROM_BOOL(gStaIO AND 4));
      4: (* DO4 상태 출력 *)
        vStr0 := CONCAT('DO4 = ', STR_FROM_BOOL(gStaIO AND 8));
      5: (* DO5 상태 출력 *)
        vStr0 := CONCAT('DO5 = ', STR_FROM_BOOL(gStaIO AND 16#10));
      6: (* DO6 상태 출력 *)
        vStr0 := CONCAT('DO6 = ', STR_FROM_BOOL(gStaIO AND 16#20));
      7: (* DO7 상태 출력 *)
        vStr0 := CONCAT('DO7 = ', STR_FROM_BOOL(gStaIO AND 16#40));
      8: (* DO8 상태 출력 *)
        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 설정 *)
      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;

  (* 통신 요청: 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
```

```
  (* 통신 설정: SET, MODBUS RTU master mode, 9600/N/8/1 *)
  extLine2.gConf(GET_SET := 1, MODE := 1, RATE := 9600, PARITY := 0, DATABITS := 8,
STOPBIT := 1);
  (* 통신 대상: Slave ID = 1, Slave Area = holding register *)
  extLine2.gComm.SLAVE_ID := 1;
  extLine2.gComm.SLAVE_AREA := 3;
  extLine2.gComm.EC := 2; (* 초기화를 위해 일부러 비정상 종료로 지정. *)
END_PROGRAM

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

  (* 통신 결과가 나올 때까지 기다림. *)
  IF extLine2.gComm.EC = 1 THEN RETURN; END_IF;

  IF extLine2.gComm.EC <> 0 THEN (* 통신 비정상 종료 *)
    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
```

사용할 수 있는 명령은 다음과 같습니다.

?0	제품명과 버전을 출력합니다.
?1	DO1의 상태를 출력합니다.
?2	DO2의 상태를 출력합니다.
?3	DO3의 상태를 출력합니다.
?4	DO4의 상태를 출력합니다.
?5	DO5의 상태를 출력합니다.
?6	DO6의 상태를 출력합니다.
?7	DO7의 상태를 출력합니다.
?8	DO8의 상태를 출력합니다.
!10	DO1을 OFF로 설정합니다.
!11	DO1을 ON으로 설정합니다.
!20	DO2를 OFF로 설정합니다.
!21	DO2를 ON으로 설정합니다.
!30	DO3을 OFF로 설정합니다.
!31	DO3을 ON으로 설정합니다.
!40	DO4를 OFF로 설정합니다.
!41	DO4를 ON으로 설정합니다.
!50	DO5을 OFF로 설정합니다.
!51	DO5을 ON으로 설정합니다.
!60	DO6을 OFF로 설정합니다.
!61	DO6을 ON으로 설정합니다.
!70	DO7을 OFF로 설정합니다.
!71	DO7을 ON으로 설정합니다.
!80	DO8을 OFF로 설정합니다.
!81	DO8을 ON으로 설정합니다.