# DG21E4I

## KC인증

R-R-Diu-DGE4I
상호명: 일품 주식회사, 제조자: 일품 주식회사, 제조국: 한국
모델명: DG21E4I


## 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

## DI 단자의 정격

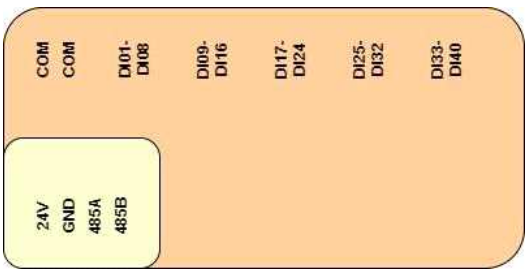
외부 전원 없이 접점만 연결 가능
외부 DO와 직결 (DI 감지용 전원을 장치 내부에 포함)

## DI 감지 표시

ON 감지시 LED 점등

## 격리(Isolation)

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




[그림1] E4I의 내부 격리

## 외형 치수

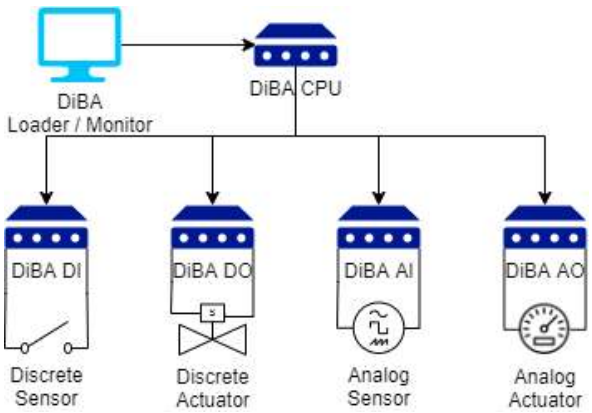
가로 145 [mm], 세로 90 [mm], 높이 41 [mm]

## 고정 방식

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

# 1. 개요

DG21E4I는 PLC(Programmable Logic Controller)의 디지털 입력(Digital Input) 모듈입니다. DiBA(다이바) PLC는 [그림2]처럼 기능별 모듈들로 자동제어시스템을 구성하며, 사용자는 제어 대상의 크기와 특성에 따라 최적의 모듈 구성을 선택할 수 있습니다.



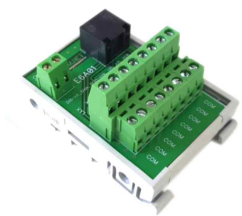

[그림2] 자동제어시스템의 구성

모델명 DG21E4I의 제품명은 MODBUS RTU DI입니다. 모델명은 일품표식(DG)과 출시연도, 대표모델명(E4I)으로 구성됩니다.

E4I는 MODBUS RTU slave로만 동작하므로 E5A(DiBA PLC CPU 모듈) 등의 MODBUS RTU master에 의해 제어됩니다.

E4I는 외부 장치를 감시하는 모듈로 다른 장치의 상태를 PLC 시스템에서 읽을 수 있는 정보로 변환합니다. E4I의 DI 단자는 외부 회로의 단락(short circuit) 혹은 개방(open circuit)을 감지하기 위해 내부의 전원을 사용하므로 외부 회로는 무전원의 접점만 사용 가능합니다. 즉, 다른 장치의 상태는 접점(일반적인 용어로 스위치)을 이용하여 전달됩니다. E4I의 DI는 단락 상태를 ON으로 읽고, 개방 상태를 OFF로 읽습니다. 이렇게 읽은 정보는 DI 별로 할당된 LED에 표시되며, ON일 때 LED가 켜지고, OFF일 때 LED가 꺼집니다. E4I는 DI로써의 기본적인 기능 외에도 최대 500 [PPS(Pulse Per Second)]의 속도로 rising edge(OFF에서 ON으로 변경되는 순간)의 수를 카운트합니다. 카운트 기능은 E4I를 이용하여 단순하게 현재의 상태를 확인하는 것뿐만 아니라, 상태 변경의 횟수도 함께 필요한 경우에 적절하게 사용될 수 있습니다.

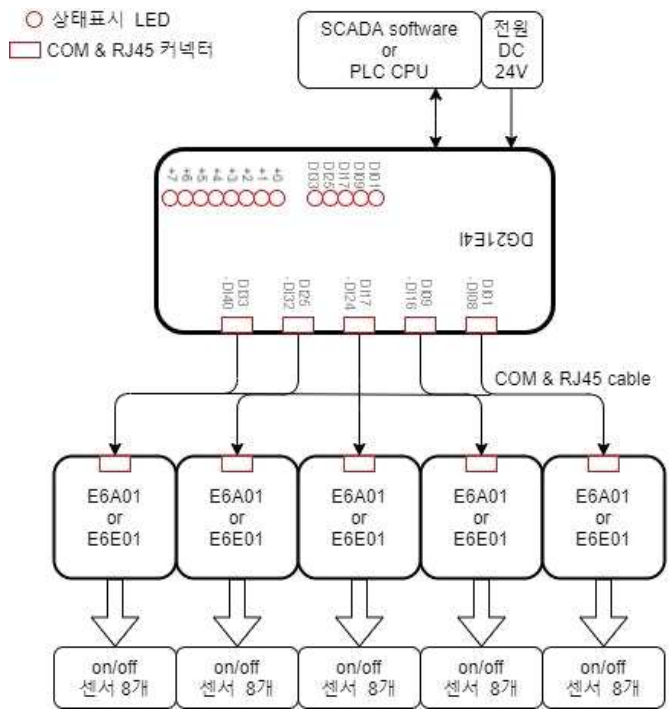
E4I의 DI 단자는 RJ45를 사용하므로 일반적인 외부 장치와는 터미널보드(RJ45는 direct cable 사용)를 통해 연결합니다. 터미널보드를 분리함으로써 단자의 수를 늘리고, DI 단자의 다양한 출력 특성에 맞는 터미널보드를 선택할 수 있습니다. 이 인터페이스는 특허 제10-2214702호로 보호받고 있습니다. 만약 E4I를 E4H(디지털 출력 모듈)의 DO 단자나 E4J(통합 입출력 모듈)의 DO 단자와 바로 연결한다면, 터미널보드 없이 RJ45 direct cable만으로 연결하여 사용할 수 있습니다. 현재 E4I에 적용할 수 있는 터미널보드는 E6A01, E6E01이 있습니다.

| 터미널보드 | E6A01 | E6E01 |
|---|---|---|
| 사진 |  |  |
| 특성 | 직결형 8단자 | 개별 격리형 8단자, 외부전원전압 10~30[Vdc], 극성무관 |

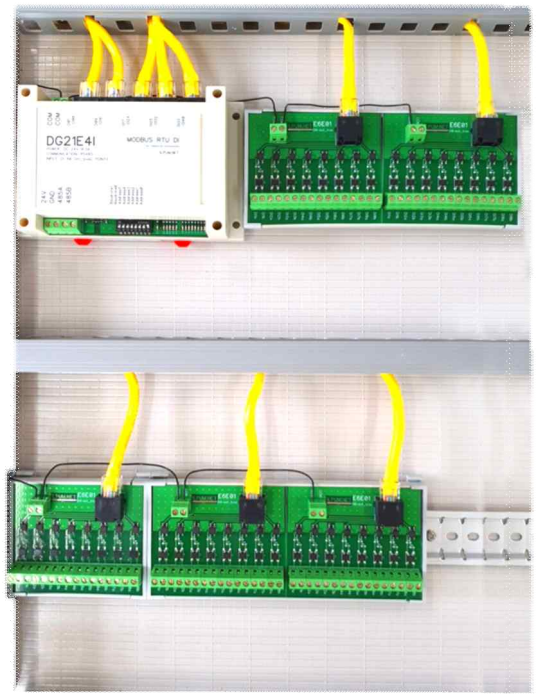
자동제어시스템의 사용자가 안전하게 다양한 장치들을 제어하도록 E4I는 격리(isolation) 설계가 적용되어 있습니다([그림1] 참고). MODBUS RTU master와 연결되는 내부 영역(격리군1)은 전원과 RS485를 포함하고, 외부 영역(격리군2)은 DI 전체를 포함합니다.

## 2. 제품의 구성 및 연결 방법

E4I는 터미널보드 E6A01(직결형 8단자), E6E01(개별 격리형 8단자, 외부전원전압 10~30[Vdc], 극성무관)과 연결할 수 있습니다.




[그림3] 구성 및 연결 방법



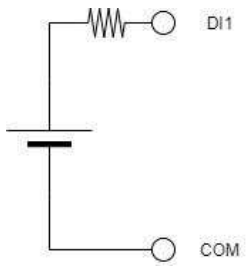
[그림4] 연결 예시 사진

E4I의 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 DI 단자의 상태를 나타냅니다. 즉, DI01 LED가 켜지면 +0~+7의 LED는 DI01~DI08 단자의 상태를 나타내고, DI33 LED가 켜지면 +0~+7의 LED는 DI33~DI40 단자의 상태를 나타냅니다. 그룹 LED가 2초씩 순서대로 켜져서 RJ45 모듈러잭 단위로 소속된 단자들(8개)의 상태를 각각 2초간 표시하여, 40개의 모든 DI 단자의 상태를 10초간 돌아가며 보입니다.

| 단자 LED / 그룹 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 |

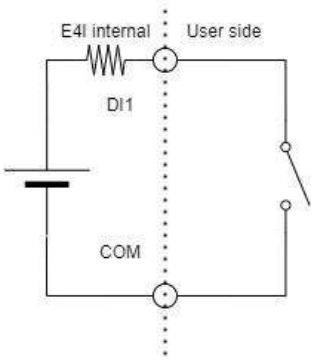
## 3. 회로 모델 및 배선

E4I는 40개의 DI를 가지고 있습니다. [그림5]는 그 중 DI01의 내부 회로를 이해하기 쉽게 표현한 것입니다. 나머지 DI도 동일한 형태를 가집니다. DI 전원은 E4I를 구동하는 전원과 격리되어 있습니다.




[그림5] DI의 내부 회로 모델

[그림6]은 스위치를 DI에 연결한 예입니다. [그림6]의 현재 상태는 스위치가 떨어진(개방된) 상태이고, E4I의 DI01은 OFF로 인식합니다. 스위치를 눌러서 회로가 연결(단락)되면 E4I는 DI01을 ON으로 인식합니다.




[그림6] DI 사용 예

## 4. 운용 기본 사항

E4I의 정보는 모두 MODBUS의 Holding Register 영역에 mapping되어 있고, 다른 영역으로는 접근할 수 없습니다. MODBUS RTU master가 E4I에게 보낸 요청을 정상적으로 처리할 수 없는 경우에 오류 응답을 합니다. 오류 응답에는 오류 코드가 포함되며, E4I가 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 |

E4I의 Dip Switch를 조작하여 통신 속도(baudrate)와 Slave ID를 설정합니다. Dip Switch를 E4I 본체 안쪽으로 밀면 ON, E4I 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<br>8 Data Bits<br>1 Stop Bit |
| OFF | ON | 19200 | |
| ON | OFF | 38400 | |
| ON | ON | 57600 | |

E4I의 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 E4I는 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<br>$= 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ | (예2) 1 = 2#000001<br>$= 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ |
|---|---|---|---|
| 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

E4I는 MODBUS slave로서 Holding Register만 제공합니다. Holding Register는 읽기와 쓰기가 모두 가능한 영역이지만, E4I가 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) |
|---|---|---|---|
| 10 | R | DI 01~16 | 0~65535 = bit mapped in the word<br>Bit 0 (LSB): DI 1 {1 is ON, 0 is OFF}<br>...<br>Bit 15: DI 16 {1 is ON, 0 is OFF} |
| 11 | R | DI 17~32 | 0~65535 = bit mapped in the word<br>Bit 0 (LSB): DI 17 {1 is ON, 0 is OFF}<br>...<br>Bit 15: DI 32 {1 is ON, 0 is OFF} |
| 12 | R | DI 33~40 | 0~255 = bit mapped in the word<br>Bit 0 (LSB): DI 33 {1 is ON, 0 is OFF}<br>...<br>Bit 7: DI 40 {1 is ON, 0 is OFF} |
| 100 | R/W | DI 01 Counter | 0~65535 = rising edge 발생 횟수<br>Up counter (1씩 증가, 65535는 0으로 순환)<br>값을 변경하면, 그 값부터 count. |
| 101 | R/W | DI 02 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 102 | R/W | DI 03 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 103 | R/W | DI 04 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 104 | R/W | DI 05 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 105 | R/W | DI 06 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 106 | R/W | DI 07 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 107 | R/W | DI 08 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 108 | R/W | DI 09 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 109 | R/W | DI 10 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 110 | R/W | DI 11 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 111 | R/W | DI 12 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 112 | R/W | DI 13 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 113 | R/W | DI 14 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 114 | R/W | DI 15 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 115 | R/W | DI 16 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 116 | R/W | DI 17 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 117 | R/W | DI 18 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 118 | R/W | DI 19 Counter | 0~65535 = rising edge 발생 횟수 (상동) |

| 주소 | Read/Write | 이름 | 값(= 의미) |
|---|---|---|---|
| 119 | R/W | DI 20 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 120 | R/W | DI 21 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 121 | R/W | DI 22 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 122 | R/W | DI 23 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 123 | R/W | DI 24 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 124 | R/W | DI 25 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 125 | R/W | DI 26 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 126 | R/W | DI 27 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 127 | R/W | DI 28 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 128 | R/W | DI 29 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 129 | R/W | DI 30 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 130 | R/W | DI 31 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 131 | R/W | DI 32 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 132 | R/W | DI 33 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 133 | R/W | DI 34 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 134 | R/W | DI 35 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 135 | R/W | DI 36 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 136 | R/W | DI 37 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 137 | R/W | DI 38 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 138 | R/W | DI 39 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 139 | R/W | DI 40 Counter | 0~65535 = rising edge 발생 횟수 (상동) |
| 9000 | R | 사용 가능한 DI 수 | 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 모듈(E5A) 사용 예

E4I는 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 E4I의 몇몇 기능을 E5A가 운용하는 예를 보입니다. E4I의 Slave ID는 1이고, 손으로 누르는 스위치를 DI1과 DI2에 연결합니다. E5A 설명서의 text SCADA를 이용하여 E4I의 정보, 스위치의 누름 상태, 누름 횟수 등을 확인하고, 누름 횟수를 지울 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, 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
  (* 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;
  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: (* 이름 및 버전 출력 *)
        vStr1 := CONCAT(gNameIO, ', ', STR_FROM_INT(gVerIO));
      1: (* DI1 상태 출력 *)
        vStr1 := CONCAT('DI1 = ', STR_FROM_BOOL(gStaIO AND 1), ', Count = ',
STR_FROM_WORD(gCntIO_1));
      2: (* DI2 상태 출력 *)
        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 지움 *)
        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 지움 *)
        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;

  (* 통신 요청: 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
  (* 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;
    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));
```

```
        (* DI 1 Counter *)
        extLine2.gComm(READ_WRITE := 0, SLAVE_ADDR := 10, DATA_COUNT := 1, MEM_ADDR
:= ADDROF(gCntIO_1));
        (* DI 2 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
```

사용할 수 있는 명령은 다음과 같습니다.

| | |
|---|---|
| ?0 | 제품명과 버전을 출력합니다. |
| ?1 | DI1의 상태를 출력합니다. |
| ?2 | DI2의 상태를 출력합니다. |
| !10 | DI1의 Counter를 지웁니다. |
| !20 | DI2의 Counter를 지웁니다. |