제품 매뉴얼

Smart I/O - I

업데이트 : 2021.10.14

Revision No.13 (2015.06.04 기준)

SMARTX® IEC SERIES

HNS
www.hnsts.co.kr
(주)에이치앤에스
Hardware & Software Total Solution
Expert Embedded Total Solution

주의사항

본 제품의 내용은 품질 향상을 위해서 사전 통보 없이 변경될 수 있습니다. 변경된 사용설명서는 저희 회사 홈페이지 www.hnsts.co.kr에서 확인 하시기 바랍니다.

본 제품을 사용하기 이전에 반드시 본 사용설명서를 충분히 읽어 본 뒤 사용하시기 바랍니다. 본 사용설명서를 충분히 읽어 보지 않은 상태에서 발생된 모든 피해는 당사에서 일체의 책임을 지지 않으므로 주의하십시오.
지정된 규격품 이외의 시스템을 사용하여 발생한 손상 및 본 사용설명서의 사용방법과 주의사항을 지키지 않아 시스템을 손상시켰을 때는 당사에서 책임지지 않으므로 주의하십시오.

1. 본 제품의 규격은 품질 개선을 위하여 임의로 사양이 변동될 수 있습니다.
2. 잘못된 배선은 제품이 파손되거나 오작동의 원인이 될 수 있습니다.
3. 외부 전원 또는 제품의 이상 발생시에 전체 제어 시스템을 보호하기 위해 IEC-Series의 외부에 보호 회로를 구성하여 사용하시기 바랍니다.
4. 오동작으로 인해 전체 시스템의 안전성 또는 인체에 심각한 문제를 초래할 수 있으며 잘못 취급 하였을 경우 사용자가 사망 또는 중상을 입는 위험상태가 발생할 수 있습니다.
5. 외부에 비상 정지 스위치, 보호 회로와 같은 시스템의 손상 및 오동 작으로부터 발생할 수 있는 피해로부터 보호할 수 있는 안전장치를 구성하여 장치를 설치하여 사용하시기 바랍니다. (잘못 취급하였을 경우 사용자가 상해를 입거나 또는 물적 손해가 발생하는 위험상태가 발생할 것으로 예상되는 경우 비상정지, 인터록 회로를 외부회로에서 구성해 주시기 바랍니다.)
6. 인체사고나 중대한 손해로 확대될 것으로 예측되는 용도로 사용하실 경우에는 이중 안전장치 등 안전대책을 세워 주시기 바랍니다.
7. 전선은 단자나사로 확실히 조여 주십시오. 접속이 불량일 경우 이상발열이나 고장의 원인이 됩니다.
8. 정격사양, 환경 등의 사양범위 이외에서는 사용하지 마십시오. 이상발열이나 고장의 원인이 됩니다.
9. 분해나 개조하지 마십시오. 감전이나 고장의 원인이 됩니다.
10. 전류가 흐르고 있는 동안에는 단자를 만지지 마십시오. 감전의 우려가 있습니다.

[쇼트/정전기 주의]

- 쇼트의 가능성이 존재하므로 작업 환경 주변에 전도 성이 높은 물체를 두지 않습니다.
- 작업환경이 건조한 경우 정전기가 발생할 수 있으므로 작업 전 절연 장갑을 착용합니다.
- 전기가 흐르는 전선의 피복이 벗겨져 있는 경우 절연체로 감아줍니다.

[전기 사용시 주의]

- AC(220V) 또는 DC(12V) 미 입력시 각 기능들이 정상 동작 안 함(ADC, DAC, RS232, Relay)
- Smart I/O전원연결은 DC 인 경우 12V(+,-극성주의), AC인 경우 220V를 사용하시기 바랍니다.
- DC 전원 사용시 극성이 맞게 되었는지 확인합니다. (+,-)

[점퍼, 결선]

- IEC-Series의 非 LITE/LITE 종류에 따른 Smart I/O의 점퍼 연결은 올바른지 확인합니다.
- IEC-Series와 Smart I/O 확장 케이블(Extension Port) 연결 시 포트와 케이블이 바뀌지 않도록 주의합니다.
- 각종 센서 사용시 전기를 인가하기 전에 최초 계획한대로 결선이 맞는지 확인합니다.

[Block 사용]

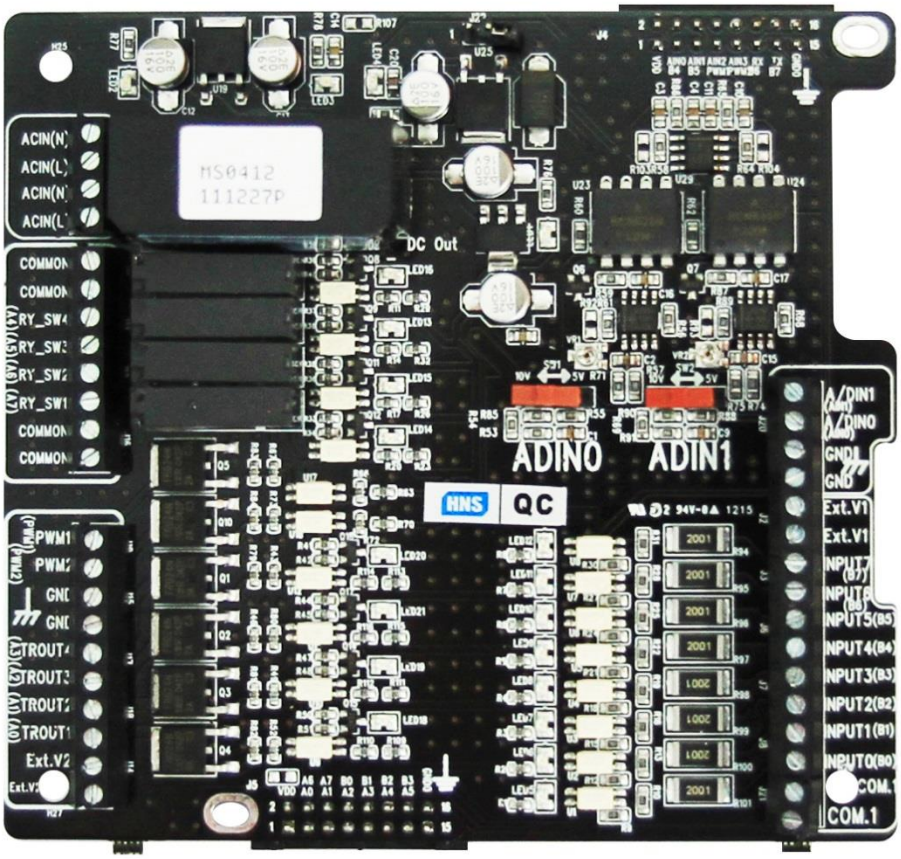
- Block 사용시 타사 제품과 함께 사용할 수 없습니다. (I/O보드와 호환되도록 설계됨)
- Block 사용시 전기적인 사양을 유의 하여 사용해야 합니다. (ex : ADC는 0~5V, 0~10V 입력)

목 차

Part - I
Smart I/O - I

Part- I . Smart I/O - I Base Board

1. Smart I/O - I 소개



Smart I/O 보드는 IEC-Series와 연동하여 적은 비용과 짧은 개발 기간으로 산업 환경에서 편리하게 적용 할 수 있는 솔루션이며, 기존의 HMI(Human Machine Interface)와 PLC(Programmable Logic Controller)의 기능을 하나의 제품으로 구성 할 수 있는 옵션 제품입니다.

1	PC 응용프로그램(C++, C#, Basic)은 가능한데 PLC의 래더 로직(Ladder Logic)을 모르시는 개발자 분들은 SmartX Framework를 통해 아주 쉽게 제어 가능합니다. (PLC를 모르시는 프로그래머에게 적합한 제품입니다.)
2	외부 기기(장치)들과 별도의 인터페이스회로 없이 바로 연결 하여 사용할 수 있습니다.
3	소형(간이) 자동화 시스템 구축을 간편하게 하실 수 있으며 비용 또한 절약 할 수 있습니다.
4	적은 비용으로 HMI와 PLC기능의 제품으로 구성 할 수 있습니다.

2. 지원되는 제품

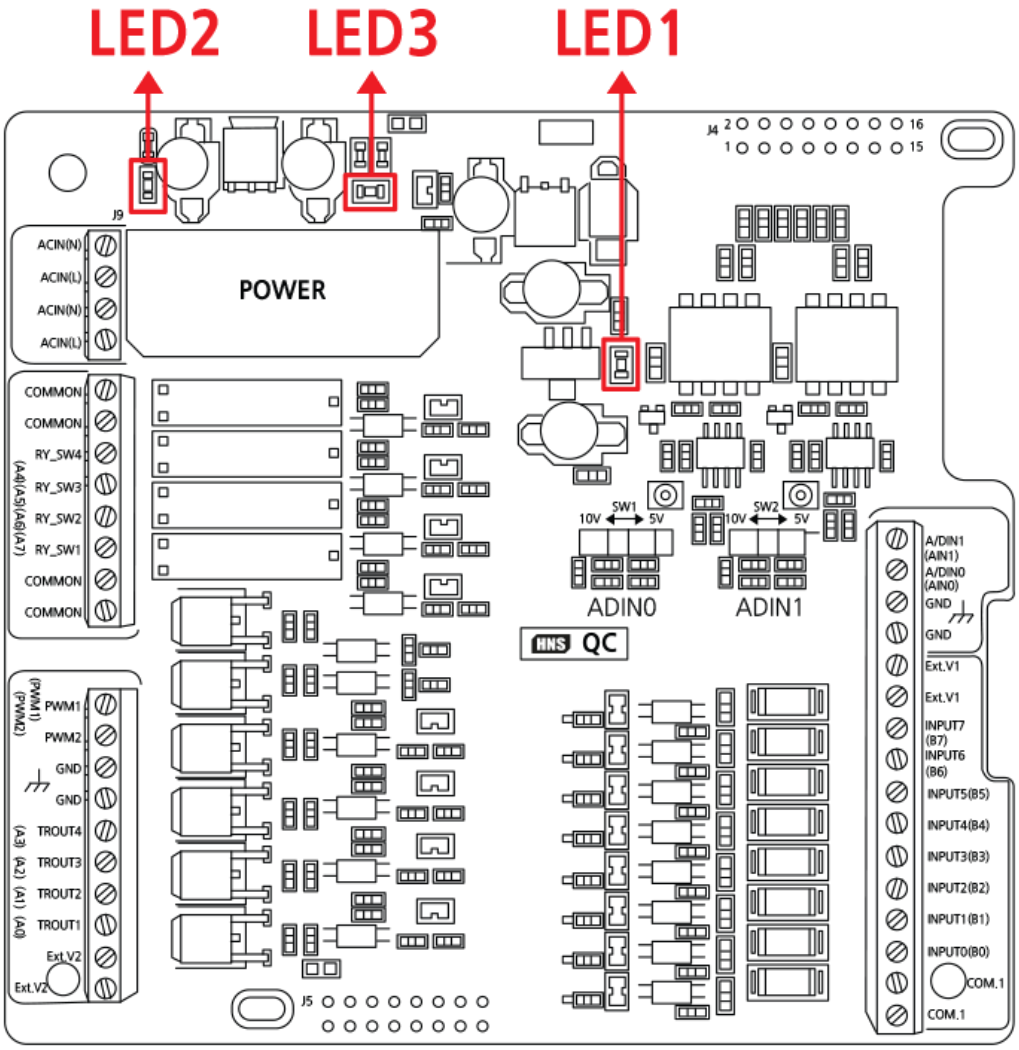
IEC Series 전 제품 사용 가능

IEC266 – Series 지원 제품명		
인치	IEC266-Series(非 Lite)	IEC266Lite-Series
4.3 inch	-	IEC266Lite-43, IEC266Lite-43[B1],[B2]
5.6 inch	-	IEC266Lite-56, IEC266Lite-56[B1],[B2]
7 inch	IEC266-07, IEC266-07[B1],[B2]	IEC266Lite-07, IEC266Lite-07[B1],[B2]
8 inch	IEC266-08, IEC266-08[B1],[B2]	-
10.2 inch	IEC266-102, IEC266-102[B1],[B2]	-

IEC667 – Series 지원 제품명		
인치	IEC667-Series(非 Lite)	IEC667Lite-Series
5.6 inch	-	IEC667Lite-56, IEC667Lite-56[B1],[B2]
7 inch	IEC667-07, IEC667-07[B1],[B2]	IEC667Lite-07, IEC667Lite-07[B1],[B2]
8 inch	IEC667-08, IEC667-08[B1],[B2]	IEC667Lite-08, IEC667Lite-08[B1],[B2]
10.2 inch	IEC667-102, IEC667-102[B1],[B2]	IEC667Lite-102, IEC667Lite-102[B1],[B2]
10.4 inch	IEC667-104, IEC667-104[B1],[B2]	IEC667Lite-104, IEC667Lite-104[B1],[B2]

IEC1000 – Series 지원 제품명		
인치	IEC1000-Series(非 Lite)	IEC1000Lite-Series
5.6 inch	-	IEC1000Lite-56, IEC1000Lite-56[B1],[B2]
7(6.95)inch	IEC1000-07N, IEC1000-07[B1],[B2]	IEC1000Lite -07, IEC1000Lite-07[B1],[B2]
8 inch	IEC1000-08, IEC1000-08[B1],[B2]	IEC1000Lite -08, IEC1000Lite-08[B1],[B2]
10.2 inch	IEC1000-102, IEC1000-102[B1],[B2]	IEC1000Lite-102, IEC1000Lite-102[B1],[B2]
10.4 inch	IEC1000-104, IEC1000-104[B1],[B2]	IEC1000Lite-104, IEC1000Lite-104[B1],[B2]
15 inch	IEC1000-150	-
XGA (모니터별도)	IEC1000XGA-I	-

3. 전원 입력 시 LED 상태



[주의] LED 장착위치는 품질개선을 위하여 임의로 위치사양이 변경될 수 있습니다.

LED 연결상태(ON)	IEC Series 미 연결	IEC Series 연결
내부 Extension-Port 전원	-	LED3(DC 3.3V)
AC/DC 전원	LED1(DC 12V), LED2(DC 5V)	LED1(DC 12V), LED2(DC 5V) LED3(DC 3.3V)

[주의] Smart I/O - I 사용 중 입력전원(AC / DC)전원 입력 안 되는 경우 Smart I/O의 기능 정상 동작 여부 체크 결과

AC/DC 전원 입력없음	Input	FET	Relay	ADC	DAC	RS232
	동작 함(ON)	동작 안 함(OFF)			지원 안함	

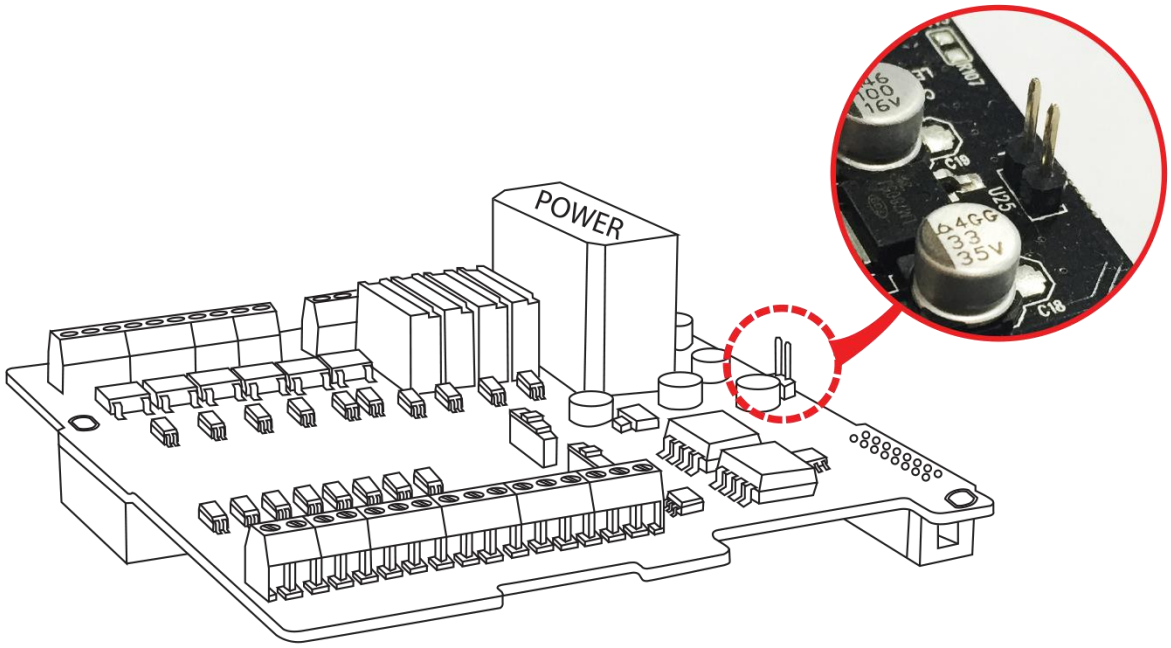
Smart I/O-I에 전원이 인가되지 않는 상태에서 특정 기능이 정상 동작하지 않을 수 있으니 위의 표를 참고하여 반드시 전원인가를 해주어야 합니다.

4. 점퍼 연결 주의사항

구입하신 IEC-Series 제품에 따라서 반드시 아래와 같이 점퍼를 설정하시고 연결 바랍니다.

연결하실 때 반드시 IEC-Series 전원을 OFF 하신 후 연결하여 사용하시기 바랍니다. **(★ 전원 ON상태에서 연결 시 고장의 원인이 됩니다.)**

[주의]	**Smart I/O-III에서는 IEC Lite-Series 제품과 연결을 지원하지 않습니다.** 따라서 IEC Lite-Series의 입력전원인 5V로 변경할 필요가 없어 점퍼가 별도로 존재하지 않습니다.



Smart I/O - I	IEC - Series 제품명	전원	점퍼
	IEC-Series	12V	점퍼 OFF(제거)
	IEC Lite-Series	5V	점퍼 ON(장착)

[주의]	잘못 된 연결 및 설정은 제품 오 동작 및 고장의 원인이 됩니다.

5. AC/DC 전원 연결하기

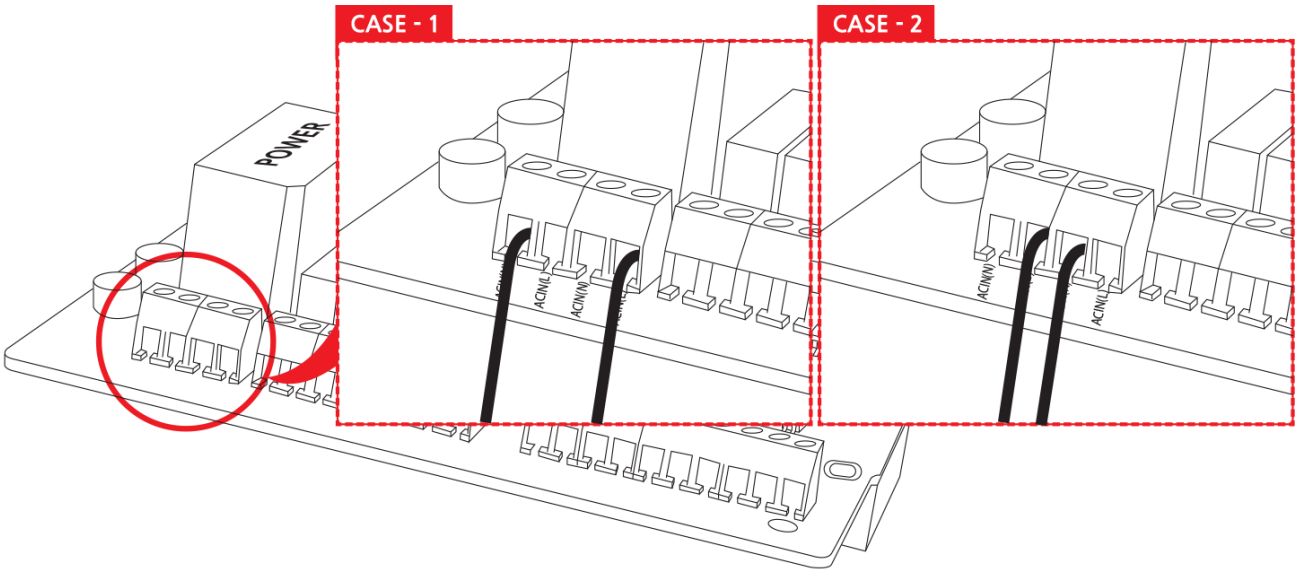
Smart I/O – I의 기능의 정상 동작을 위해 반드시 AC 또는 DC 외부입력전원이 필요합니다.

CASE-1) AC 전원 연결하기(기본 사양)

Smart I/O – I는 시스템 보호를 위하여 전원 절연(Isolation)을 목적으로 AC전원을 입력하도록 설계 되어 있습니다.
AC 전원(220V) 60HZ 입력하기 위해서는 아래의 그림을 참고하여 전원을 입력하시기 바랍니다.

[주의]	1. 제품에 전원(SMPS) 반드시 달려 있어야 합니다. 2. AC/DC 전원을 동시에 입력 할 수 없습니다. 3. 잘못된 배선으로 연결 시 폭발/상해/제품고장 위험이 있습니다. 4. AC전원 연결 시 위험하므로 반드시 OFF 된 상태에서 연결바랍니다. 5. 극성이 없으므로 N,L이 뒤바뀌어도 무관합니다. 6. 전선은 단자 나사로 확실히 조여 접촉불량이 발생되지 않습니다.

[STEP-1] 아래의 두가지 방법으로 하나를 선택하여 전원을 연결



▲ Smart I/O - I

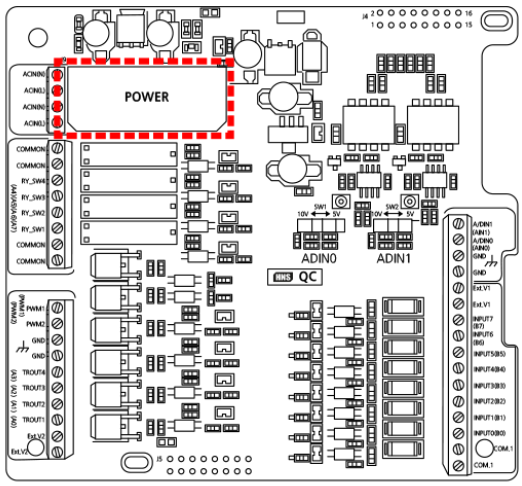
CASE-2) DC 전원 연결하기(별도 개조작업 필요)

Smart I/O – I는 시스템 보호를 위하여 전원 절연(Isolation)을 목적으로 AC전원을 입력하도록 설계 되어 있습니다. AC전원을 인가 하기 어려운 환경에서 DC전원을 인가 하도록 변경하기 위한 설명입니다. DC전원을 사용하실 경우 입력 전압은 반드시 아래의 표를 참고하여 입력하시기 바랍니다.

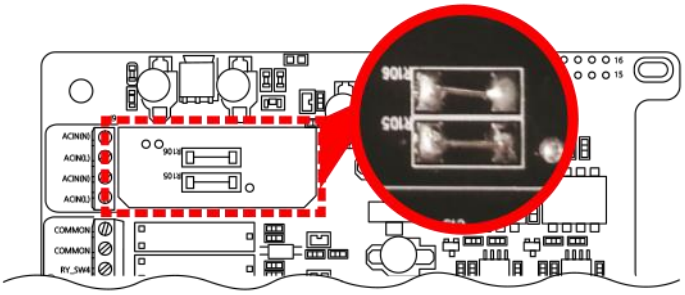
제품명	입력전압
Smart I/O – I, II	DC 12V, 300mA이상 사용 권장
Smart I/O – III	DC 12V, 800mA이상 사용 권장

[참고]	제품 구입시 DC전원으로 변경요청 하시면 DC전원으로 바로 사용할 수 있는 상태의 제품을 받으실 수 있습니다.

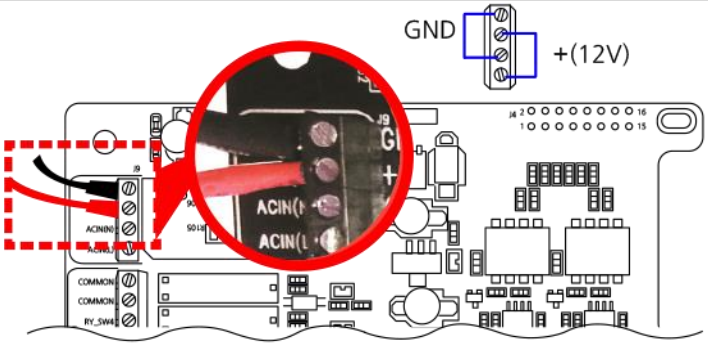
[STEP-1] SMPS(Power)을 제거



[STEP-2] SMPS을 제거한 부분에 사진과 같이 Shunt 저항을 장착할 수 있도록 되어 있으며, 이곳에 shunt(0Ω)저항 또는 점퍼선을 연결



[STEP-3] 전원연결은 AC와 달리 DC전원을 사용하실 경우 반드시 Board에 표시되어 있는 극성을 참고하여 연결



6. 입력/출력 특성

Smart I/O – I 내부에는 입력 8Ch, A/D 2Ch, 출력 8Ch, PWM 2Ch 로 다음과 같이 구성되어 있습니다.

① Smart I/O - I Base Board

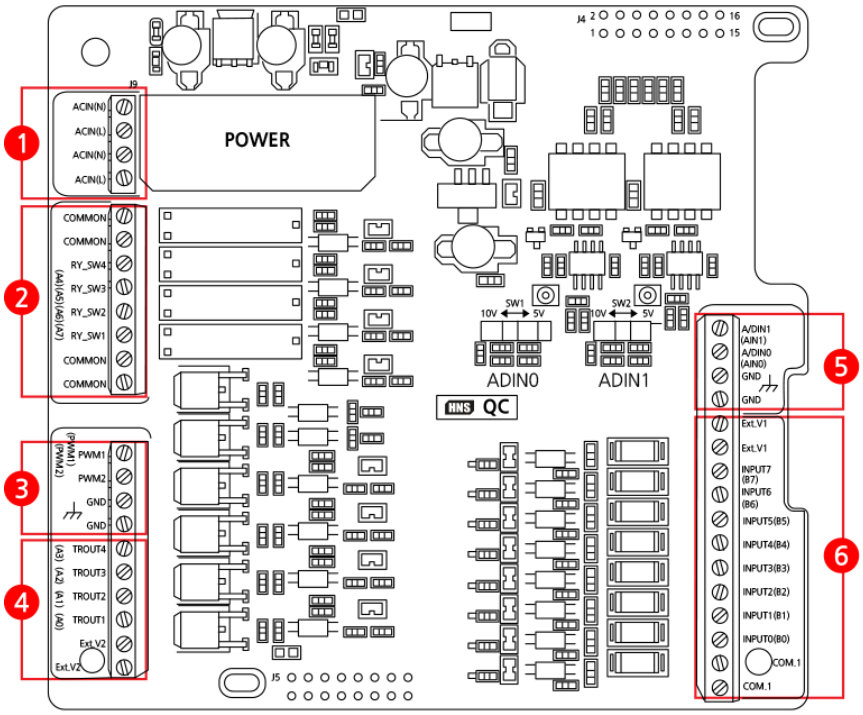
Direction	Type	Name	Channel	Rating	Notice
INPUT	DC입력(무극성)	INPUT0 ~ 7	8Ch	DC 12V ~ DC 24V	PORTB0 ~ 7
	A/D 입력	A/D IN	2Ch	5V / 10V	IEC266 – 10bit IEC667/1000 – 12bit
OUTPUT	FET 출력	TROUT1 ~ 4	4Ch	VSS = 55V, ID = 17A	N-Channel
	RELAY 출력	RY_SW1 ~ 4	4Ch	5A 250VAC 5A 30VDC	–
	PWM(FET)출력	PWM1 ~ 2	2Ch	VSS = 55V, ID = 17A	N-Channel

[주의]	Smart I/O – I을 사용하기 위해서는 AC IN에 전원을 인가하여야 합니다. (AC Input ➔ AC90V ~ AC240V 50/60Hz)
	AC 전원 대신 DC전원을 인가하여 사용할 수 있습니다. 당사로 문의 후 제품을 저희 회사로 보내주시기 바랍니다. (부품을 수정해야 합니다.) **[자료참고] 5. AC/DC 변환방법 및 주의사항**
	IEC667-Series에서 Port-B, Port-D, Port-F, Port-G는 입력으로 사용하는 것을 권장합니다. Port-B 0, 1, 2, 3, 4, 5 Pin은 전원 인가 후 9.3s 후 High로 출력되며, Port-D 0 ~ 7 Pin , Port-F 0 ~7 Pin, Port-G 7 Pin은 전원 인가 후 1.4s 후 High로 출력됩니다. 제시한 포트는 Pull Down 저항을 걸어서 사용하는 경우에는 문제가 되지 않으며, 자세한 사항은 [SmartX Framework 프로그래밍 가이드] ➔ [SmartGPIO] ➔ [Port 초기 상태 값]을 참고하시기 바랍니다.

② Smart I/O - I의 INPUT과 OUTPUT IEC-Series 연결포트 정보

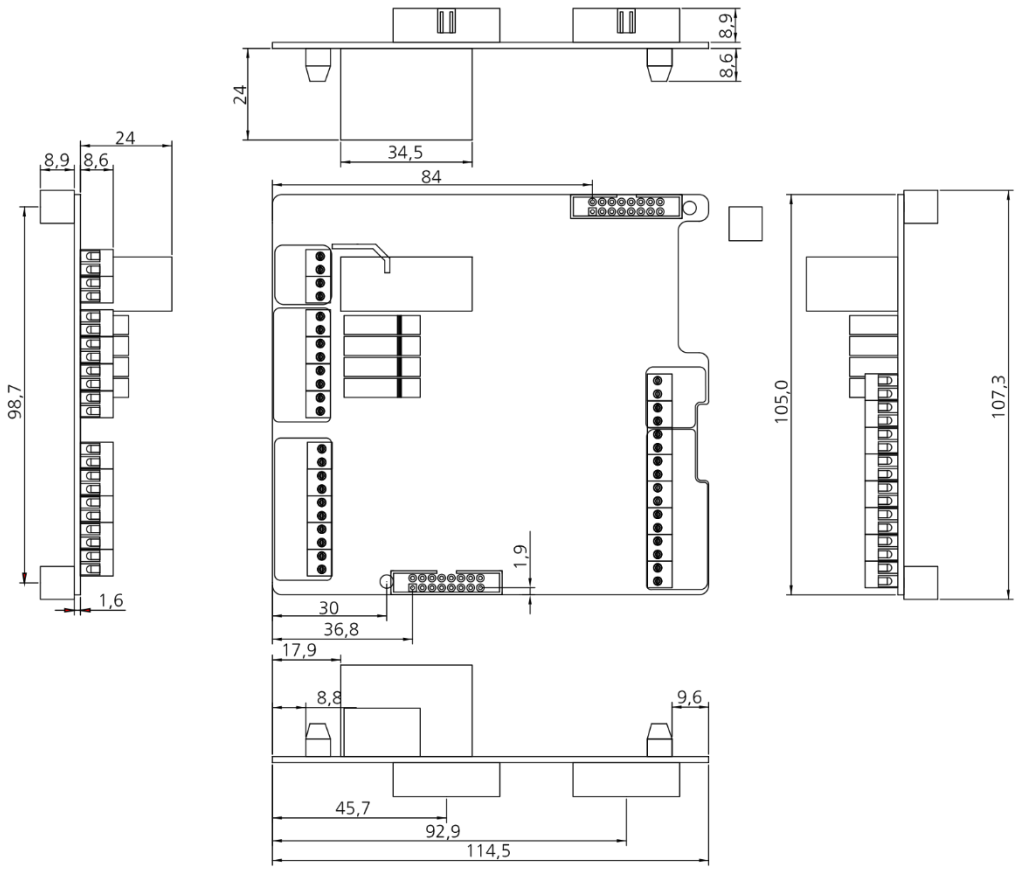
INPUT	DC입력 Smart I/O	INPUT0	INPUT1	INPUT2	INPUT3	INPUT4	INPUT5	INPUT6	INPUT7
	PORT NAME	PORTB0	PORTB1	PORTB2	PORTB3	PORTB4	PORTB5	PORTB6	PORTB7
	ADC입력	A/D IN0	A/D IN1	–	–	–	–	–	–
	PORT NAME	AIN0	AIN1	–	–	–	–	–	–
OUTPUT	FET출력 Smart I/O	TROUT1	TROUT2	TROUT3	TROUT4	–	–	–	–
	PORT NAME	PORTA0	PORTA1	PORTA2	PORTA3	–	–	–	–
	RELAY 출력 Smart I/O	RY_SW1	RY_SW2	RY_SW3	RY_SW4	–	–	–	–
	PORT NAME	PORTA7	PORTA6	PORTA5	PORTA4	–	–	–	–
	PWM 출력	PWM1	PWM2	–	–	–	–	–	–
	PORT NAME	PWM1	PWM2	–	–	–	–	–	–

7. 각부 명칭



1. AC90V~240V 50Hz/60Hz
2. Relay S/W 1~4
3. PWM 1~2
4. FETOUT 1~4
5. A/D IN/ 1~2
6. INPUT 0~7

8. 외형치수



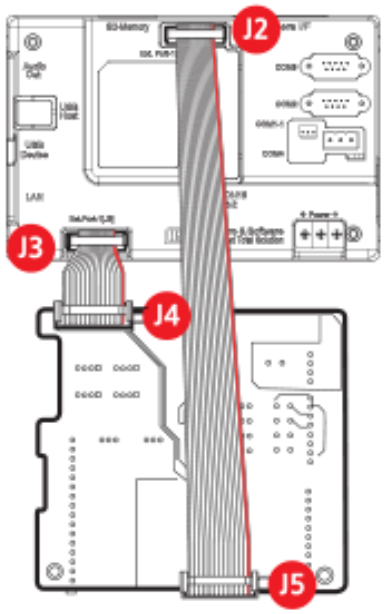
[참고] 본 제품의 도면은 **[홈페이지]-[자료실]-[도면 및 승인원]에서 다운로드** 하실 수 있습니다.

9. IEC-Series와 Smart I/O - I 제품 연결하기(케이블 연결)

[주의]	연결하실 때 반드시 IEC-Series 전원을 OFF 하신 후 연결하여 사용하시기 바랍니다. (전원 ON 상태에서 연결 시 고장의 원인이 됩니다.)
	Extension Port-I/II IDC Cable Set의 경우 50cm 이하의 길이로 사용 권장합니다. 50cm 이상으로 길이를 연장하는 경우 제품의 정상적인 동작을 보장하지 못합니다.

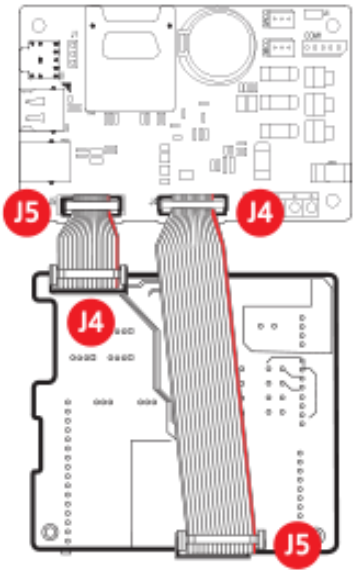
[연결사용 안내]

※ IEC266 – Series



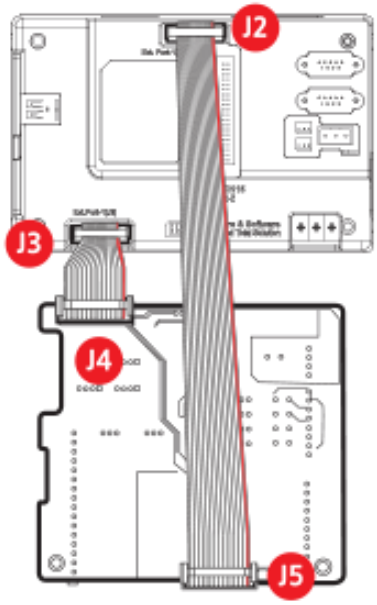
IEC-Series	연결	Smart I/O-I
J2	→	J5
J3	→	J4

※ IEC266Lite – Series



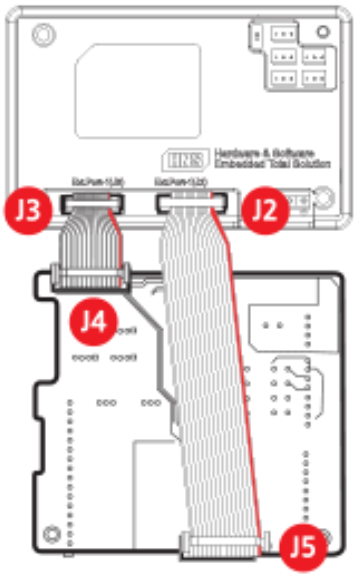
IEC-Series	연결	Smart I/O-I
J5	→	J4
J4	→	J5

※ IEC667/1000/XGA – I Series



IEC-Series	연결	Smart I/O-I
J2	→	J5
J3	→	J4

※ IEC667/1000Lite – Series



IEC-Series	연결	Smart I/O-I
J2	→	J5
J3	→	J4

[MEMO]

Part -II
Smart I/O - I 기능소개

Part-II. Smart I/O - I 기능소개

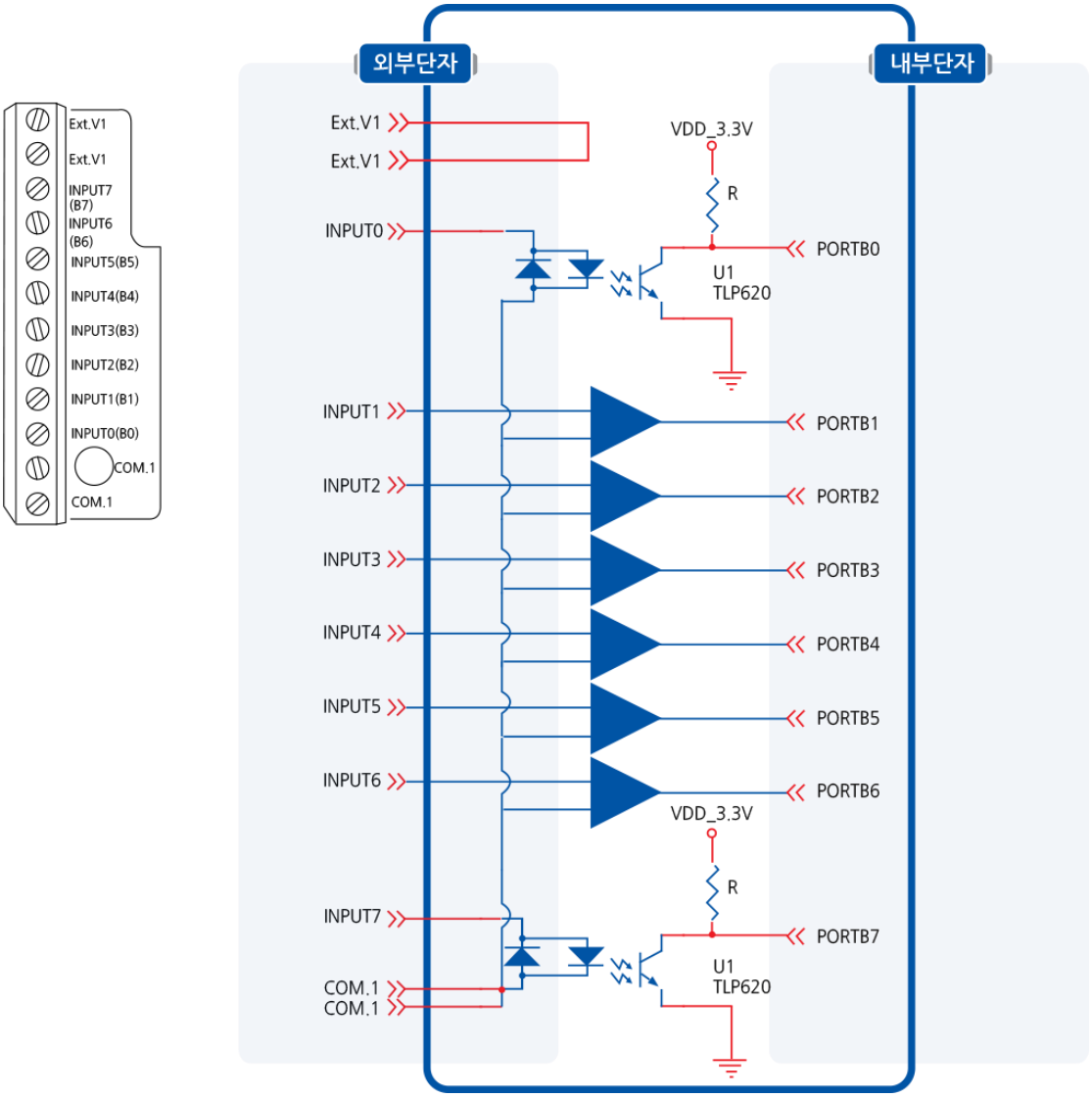
1. INPUT 입력단자

1) INPUT 입력단자 소개

8 채널의 입력을 처리할 수 있으며 극성 없이 입력이 가능하며 High Logic은 DC12 ~ 24V를 사용할 수 있습니다.

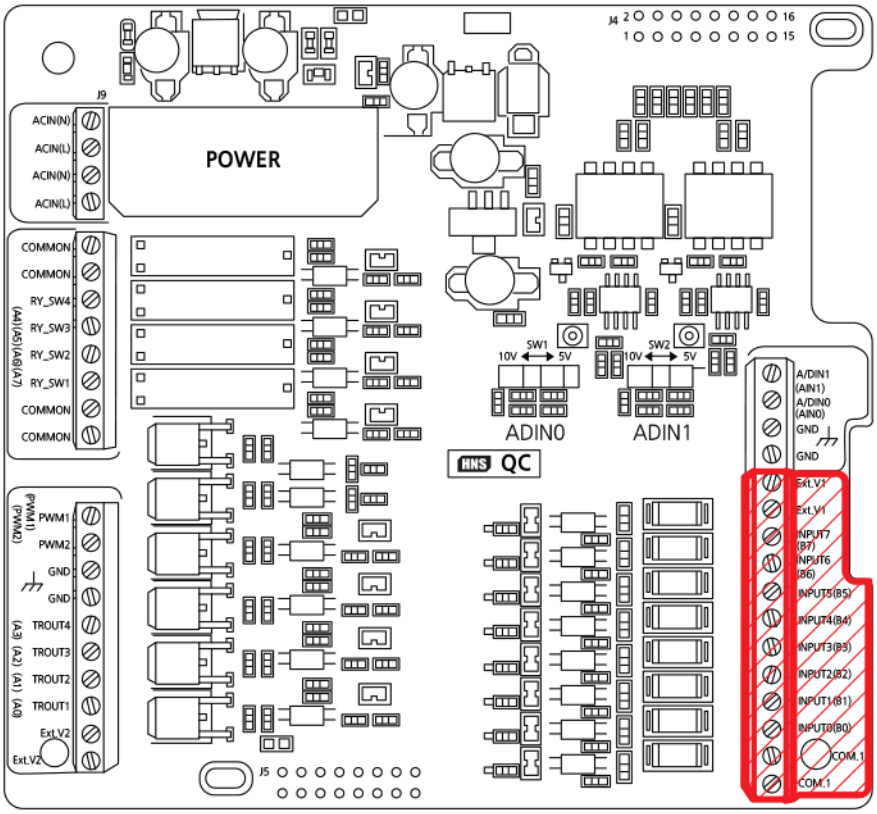
외부입력단자	INPUT0	INPUT1	INPUT2	INPUT3	INPUT4	INPUT5	INPUT6	INPUT7
내부 Extension Port 연결단자	PORTB0	PORTB1	PORTB2	PORTB3	PORTB4	PORTB5	PORTB6	PORTB7

Direction	외부상태(입출력)	PortData 값(True/False)
입력	High(12~24V)	False(0)
	Low(0V)	True(1)



[주의]	포토커플러 입력은 무극성이며 COM.1 내부는 공통으로 묶여 있습니다.
	전원은 DC 12 ~ 24V까지 사용가능하며, COM.1 / INPUT0 ~ 7전원은(+/-) 상관없이 반대로만 연결하면 됩니다.

2) INPUT 입력단자 위치

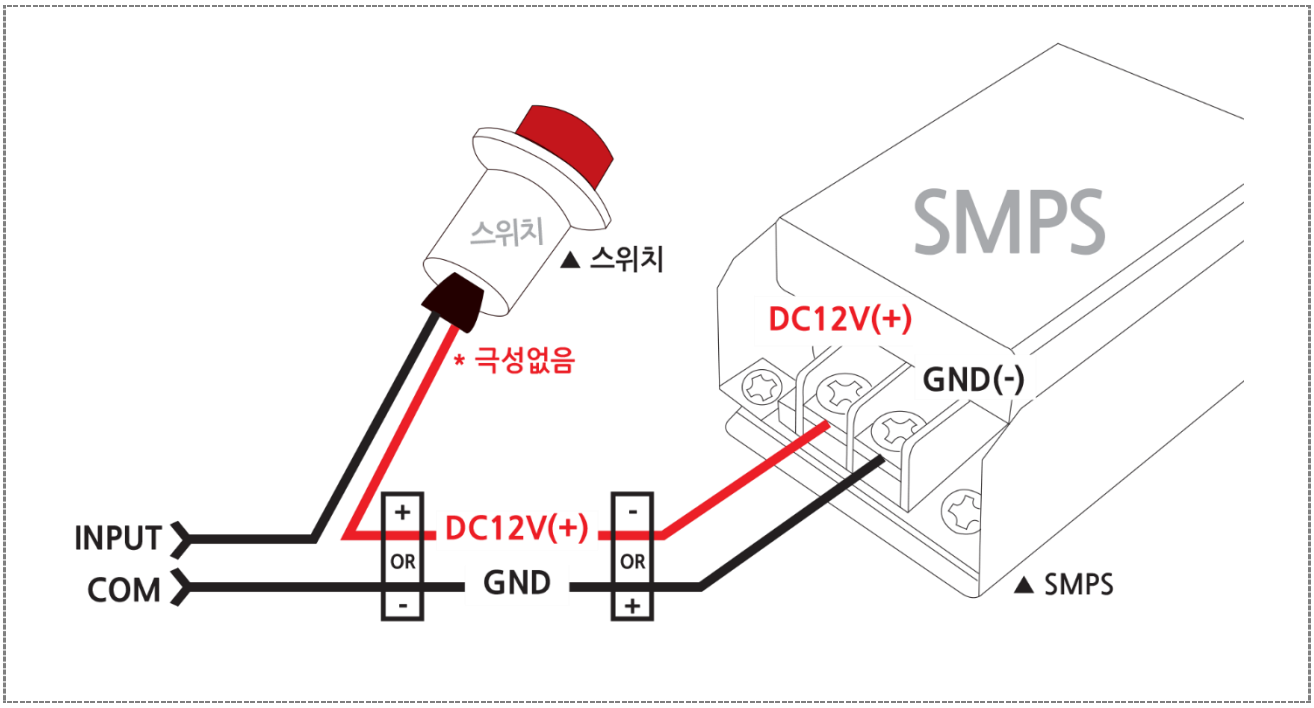


3) INPUT 입력단자 응용방법

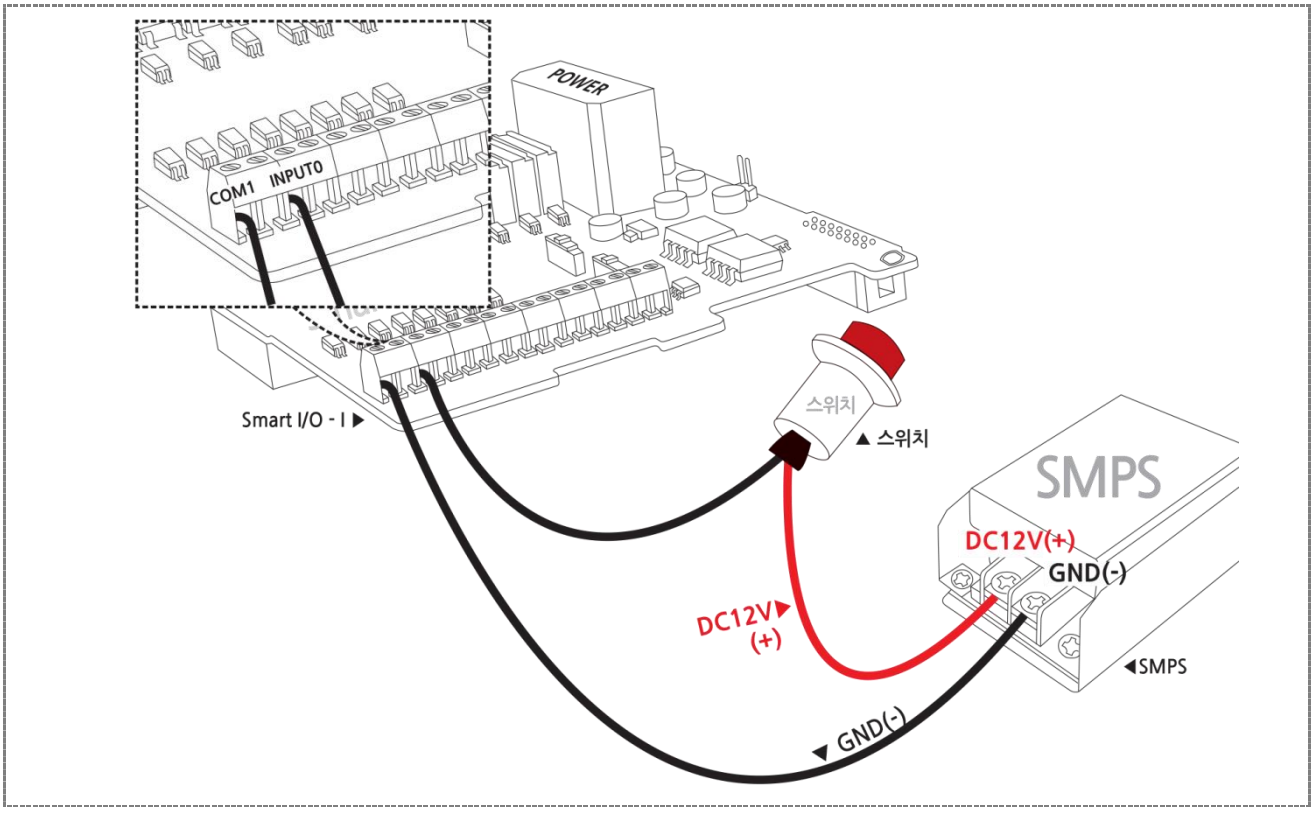
3-1) 스위치 연결방법

아래그림을 참고하여 결선하시고, Smart I/O – INPUT의 IN0/INPUT0 – 스위치 항목에서 확인 가능

① 결선도



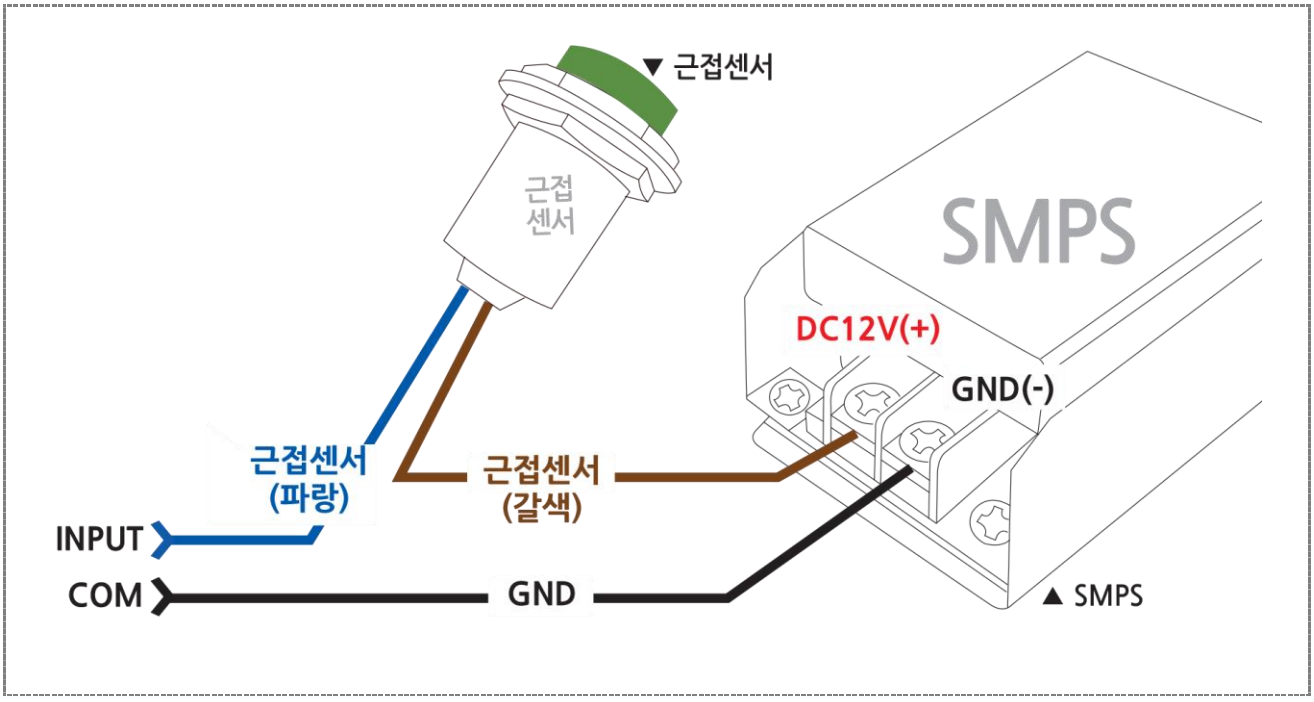
② Smart I/O - I Base Board 결선도 예



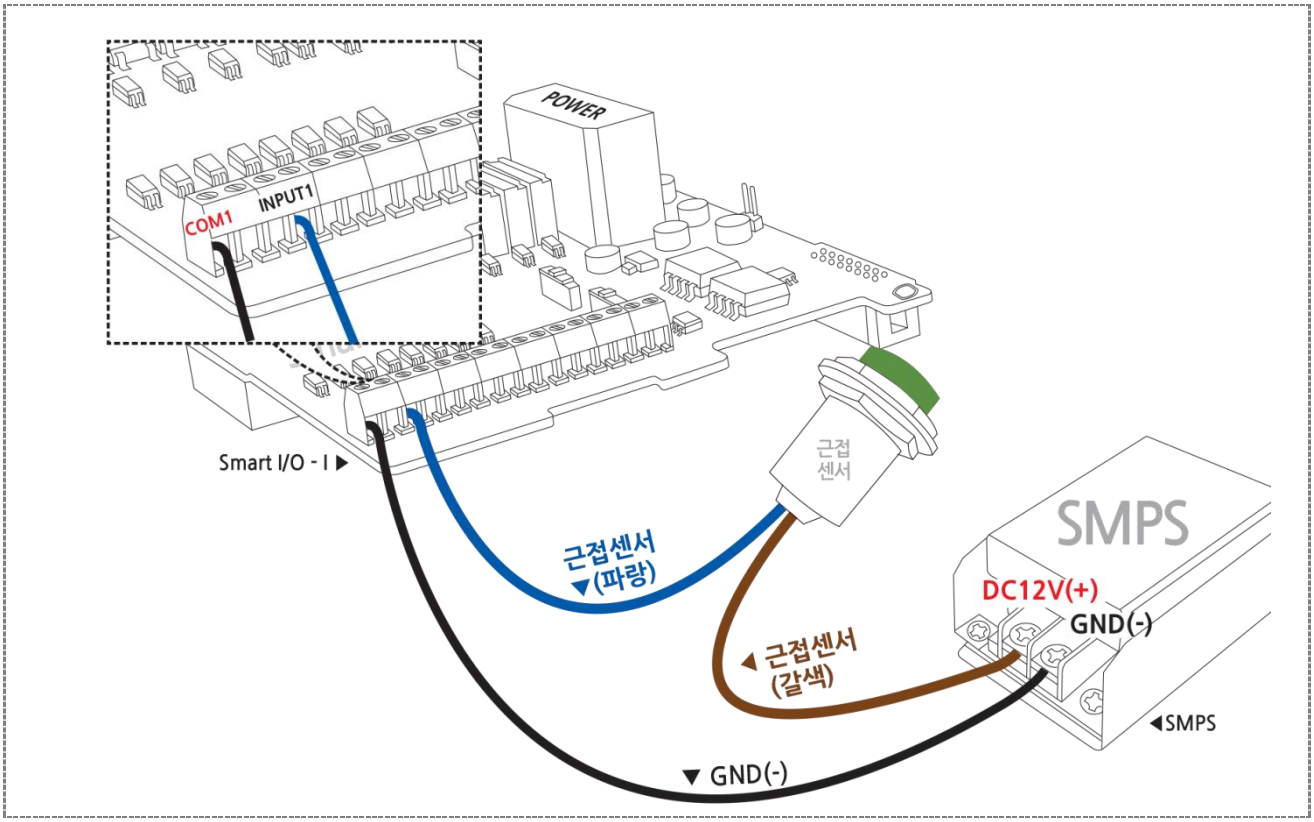
3-2) 2선식 센서 연결방법(-COM)

아래와 같이 결선을 하시고, Smart I/O_Input의 IN1/INPUT1 – 2선식 근접센서 항목에서 확인가능

① 결선도



② Smart I/O - I Base Board 결선도 예

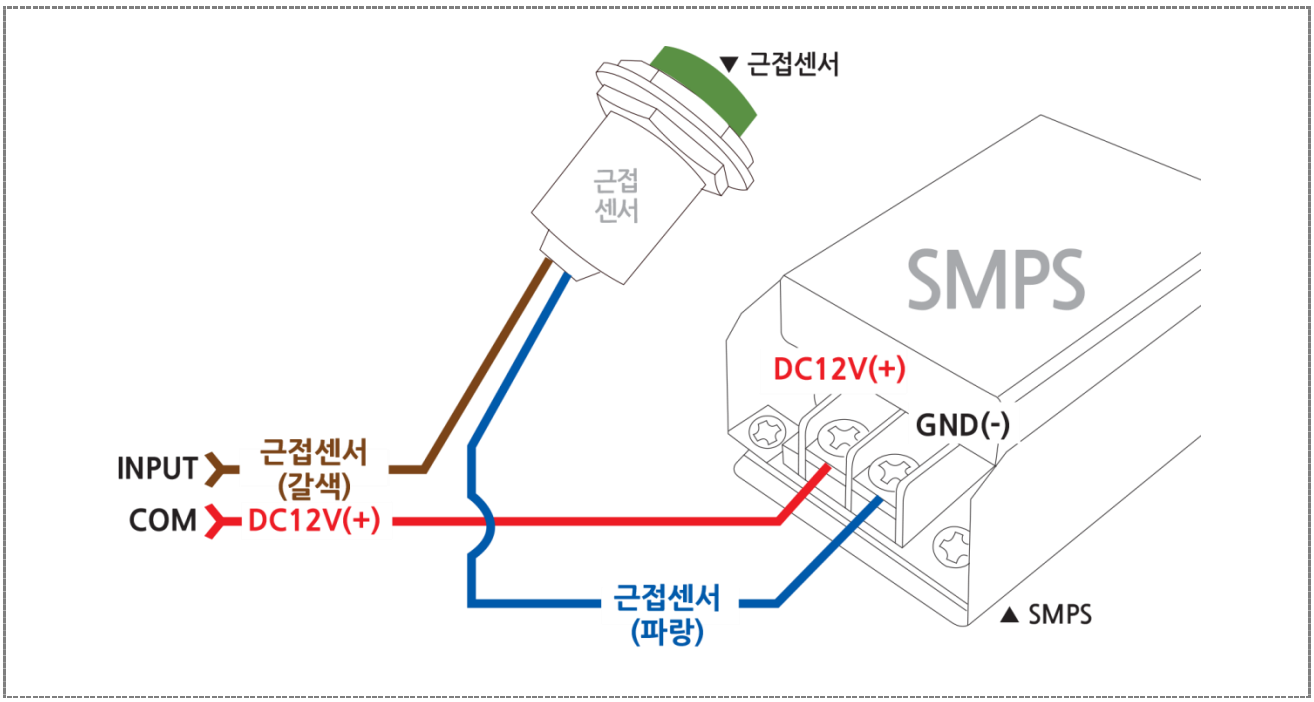


[참조]	오토닉스(PRT18-8DO) 데이터 시트 참조

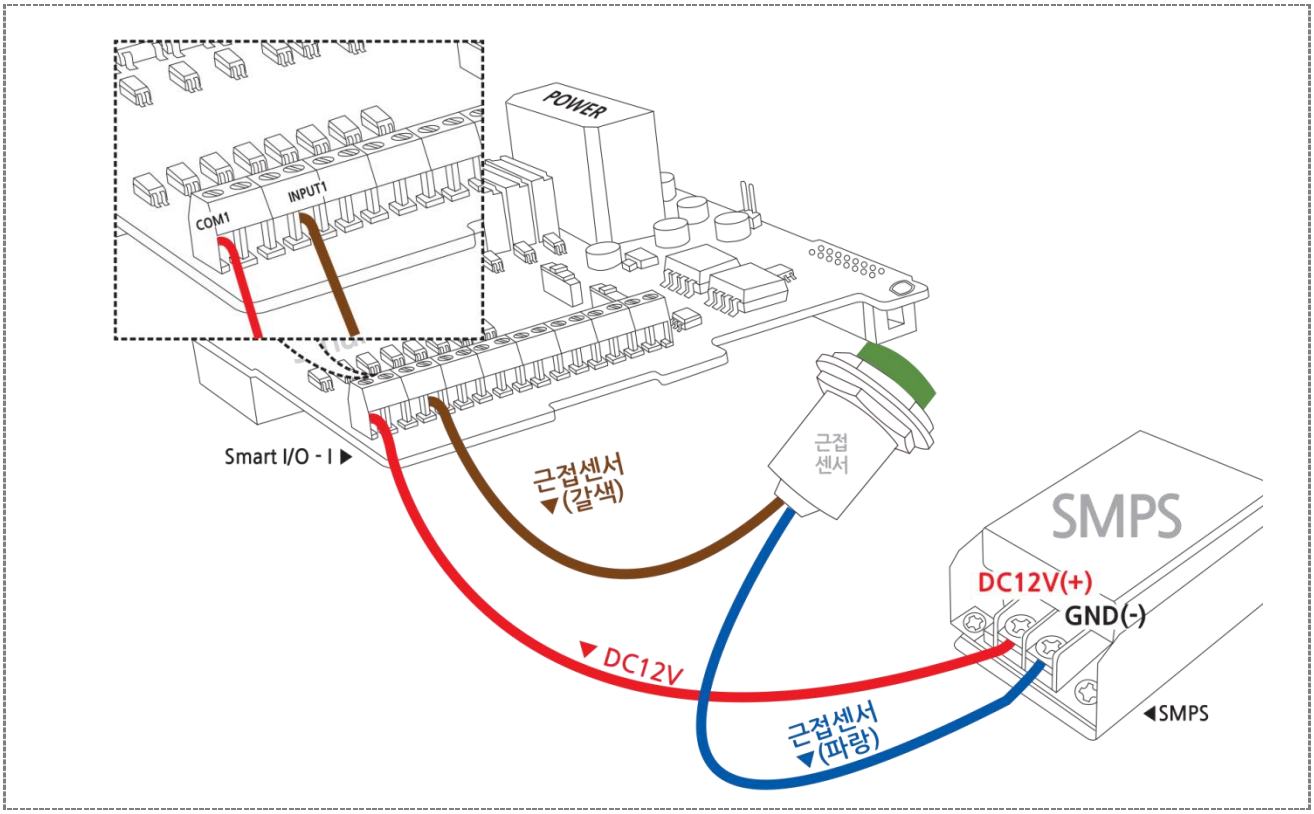
3-3) 2선식 센서 연결방법(+COM)

아래와 같이 결선을 하시고, Smart I/O_Input의 IN1/INPUT1 – 2선식 근접센서 항목에서 확인가능

① 결선도



② Smart I/O - I Base Board 결선도 예

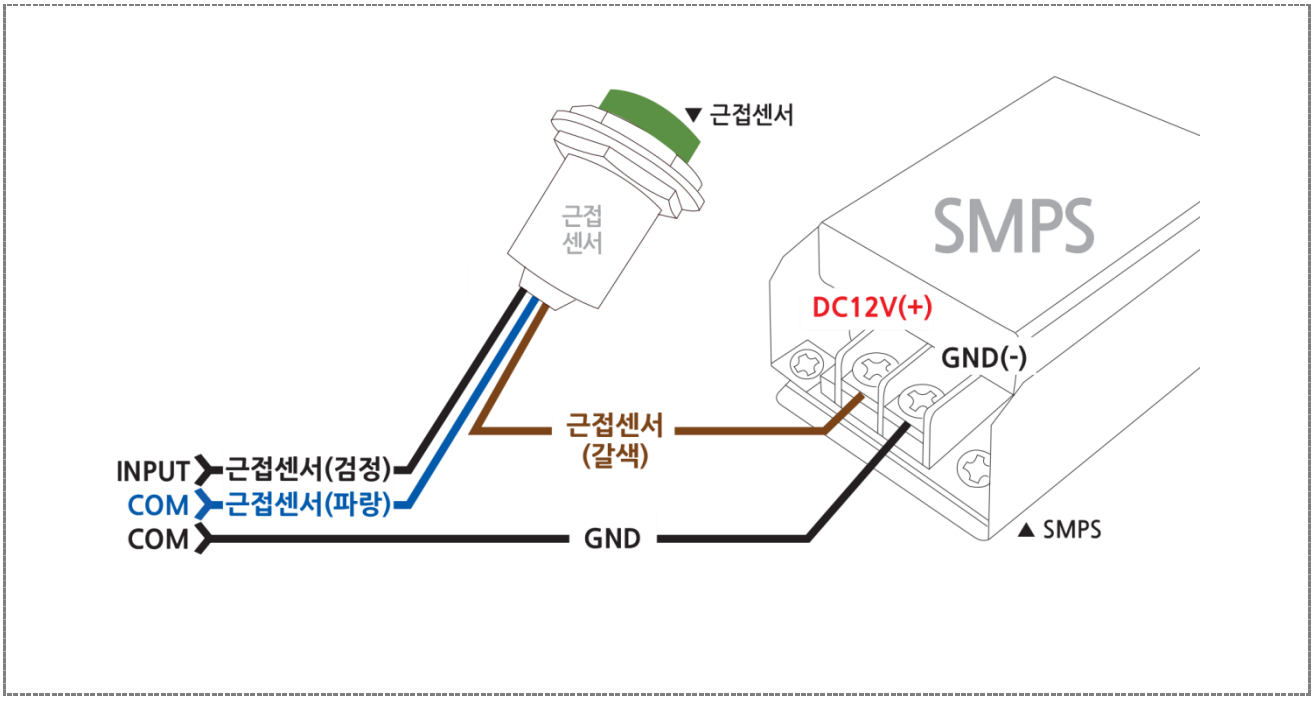


[참조]	오토닉스(PRT18-8DO) 데이터 시트 참조

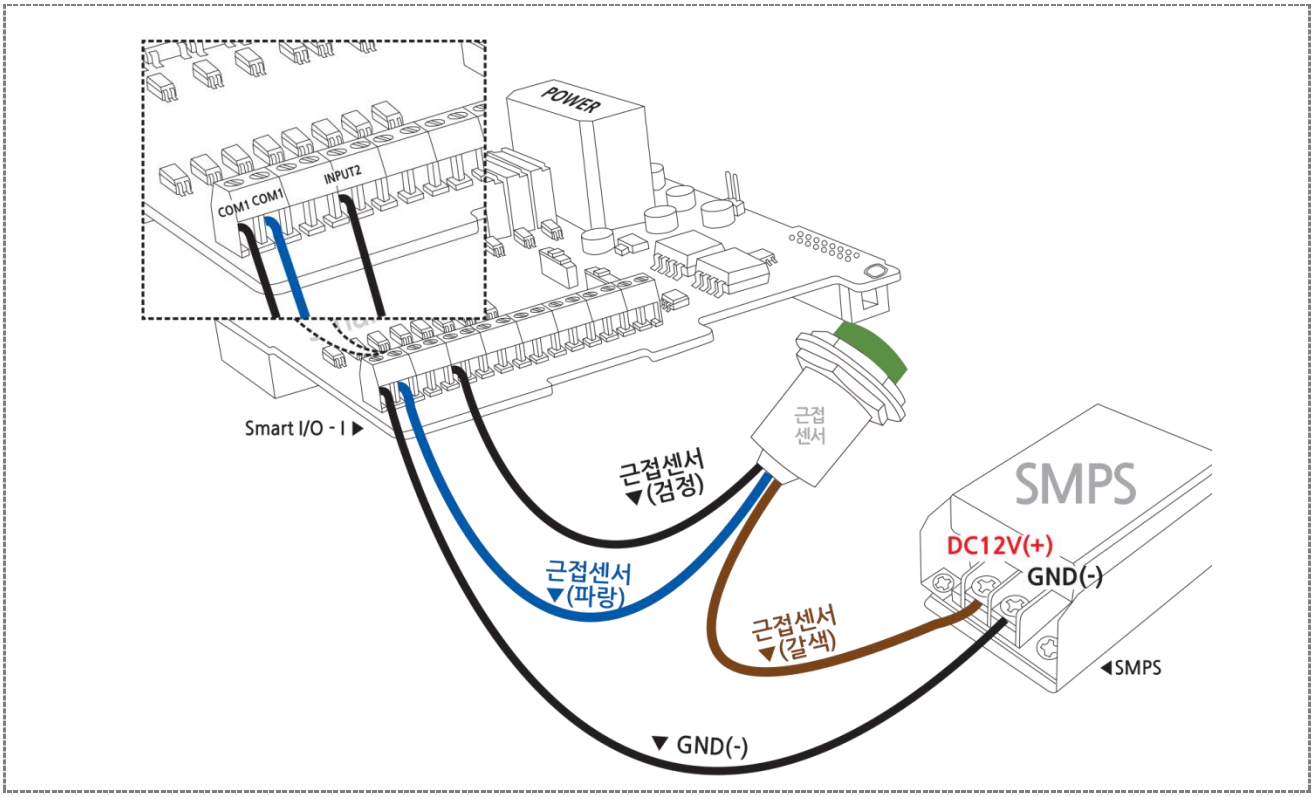
3-4) 3선식 센서 연결방법(PNP형)

아래와 같이 결선을 하시고, Smart I/O_INPUT의 IN2/INPUT2 – 3선식 PNP근접센서 항목에서 확인가능

① 결선도



② SmartI/O - I Base Board 결선도 예

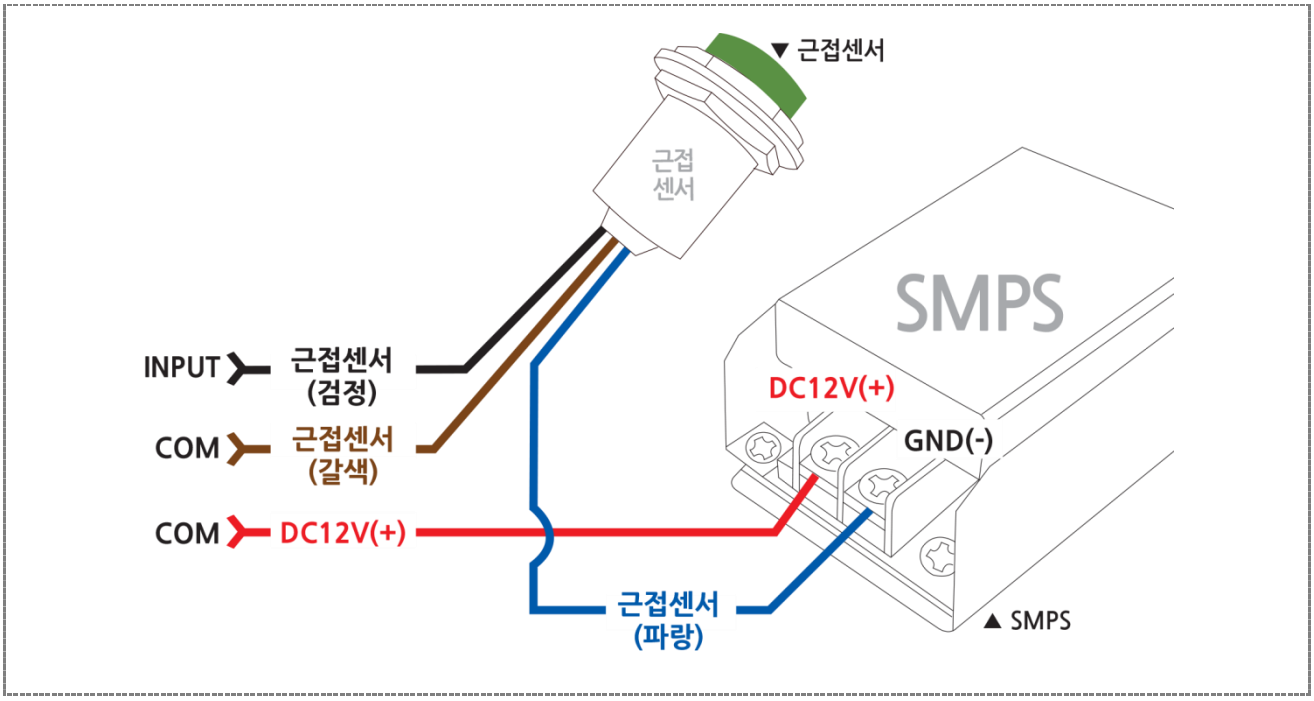


[참조]	오토닉스(PRT18-8DP) 데이터 시트 참조

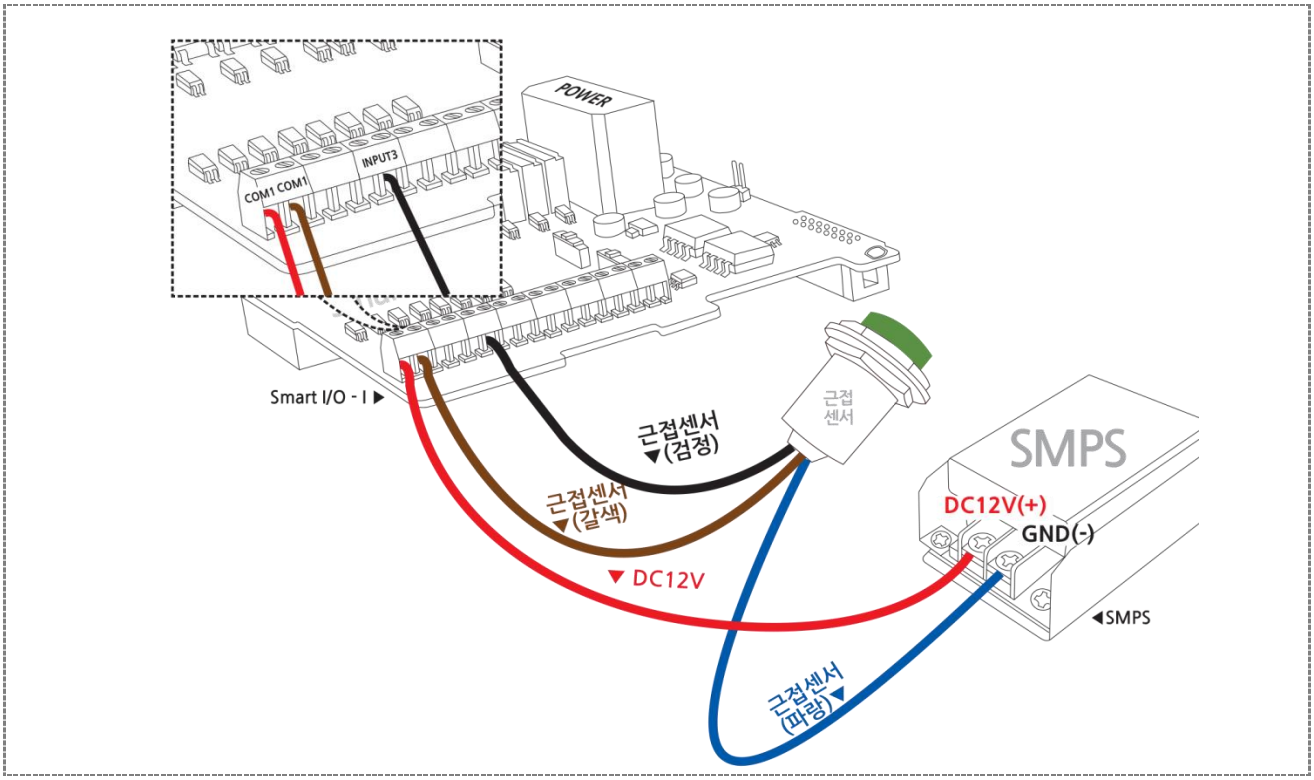
3-5) 3선식 센서 연결방법(NPN형)

아래와 같이 결선을 하시고, Smart I/O_INPUT의 IN3/INPUT3 – 3선식NPN근접센서 항목에서 확인가능

① 결선도



② SmartI/O - I Base Board 결선도 예



[참조]	오토닉스(PRT18-8DN) 데이터 시트 참조

4) 언어별 주요소스 코드

C#

```
// PORTB의 상태변경이 감지되었을 때 발생되는 이벤트
private void smartGPIO1_EvtPortBDatasChange(object sender, EventArgs e)
{
    int iPortDatas;
    // 이벤트 인자로 받은 데이터를 포트의 상태를 얻기 위해 형변환한다.
    SmartX.PORTDataEvtArgs PortDatas;
    // EventArgs e 에는 포트의 데이터를 직접 접근할 수 없으므로 SmartX.PORTDataEvtArgs로 변환한다.
    // ePortDatas.iPortDatas 필드가 있는데 이 값을 읽으면 포트의 상태를 얻을 수 있다.
    PortDatas = (SmartX.PORTDataEvtArgs)e;

    // iPortDatas = 포트0~8번까지의 값(0~255)
    iPortDatas = PortDatas.iPortDatas;

    // Port 입력 상태에 따른 처리 코드로 개발자가 직접 작성
    SmartGPIO_Parsing(iPortDatas);
}
```

VB.NET

```
' PORTB의 상태변경이 감지되었을 때 발생되는 이벤트
Private Sub smartGPIO1_EvtPortBDatasChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
smartGPIO1.EvtPortBDatasChange
    Dim iPortDatas As Integer
    Dim PortDatas As SmartX.PORTDataEvtArgs
    PortDatas = e
    iPortDatas = PortDatas.iPortDatas
    SmartGPIO_Parsing(iPortDatas)
End Sub
```

C++

```
void CSmartGPIOEVCDlg::OnTimer(UINT nIDEvent)
{       // Timer 식별자가 3인경우. SetTimer(3, 2000, NULL);에서 식별자 지정
    if(nIDEvent == 3)
    {
        int iPortDatas;
        iPortDatas = pf_GetPortBDatas();
        //포트의 상태가 변경 될때만 처리
        if(m_iOldPortBData != iPortDatas)
        {
            m_iOldPortBData = iPortDatas;
            // Port 입력 상태에 따른 처리 코드로 개발자가 직접 작성
            PortB_Parsing(iPortDatas);
        }
    }
    CDialog::OnTimer(nIDEvent);
}
```

5) Application 응용예제

SmartX Framework	SmartGPIO
소스파일	Smart I/O_Input
소스경로	홈페이지 [자료실] → [Application Note] → [Smart I/O 활용]
기 능	스위치, 근접센서 등 입력기능
응용분야	스위치, 외부입력장치
준비사항	▪ 스위치 : 스위치 ▪ 근접2선식(Nomal Open)센서 : ㈜오토닉스(PRT18-8DO) ▪ 근접3선식(PNP) 센서 : ㈜오토닉스(PR18-8DP) ▪ 근접3선식(NPN) 센서 : ㈜오토닉스(PR18-8DN)

❶ 이미지/카운트 초기화 버튼 ❷ 이미지 표시창 ❸ 카운트 표시

[동작설명]

▶ 프로그램이 시작되면서 외부에서 각각의 포트 별로 입력신호(LOW → HIGH)가 발생하면 ②의 이미지가 변경되며 ③에 입력 카운트가 1 증가합니다.

▶ 만약 프로그램 실행 도중에 ③의 카운트를 0으로 초기화 하고 싶은 경우 ①의 Initial 버튼을 클릭하면 초기화 됩니다.

5-1) C#예제 전체소스 코드

본 예제의 회로도에서 INPUT0은 스위치, INPUT1은 2선식 근접센서, INPUT2는 3선식 PNP 근접센서, INPUT3는 3선식 NPN근접센서로 연결되어 있습니다. 여기서 설명하는 INPUT0은 스위치(SW)이며 스위치 이외의 근접센서(2선식, 3선식)에 관한 설명은 Application Notes의 Smart I/O_Input 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
// 폼 로드시 실행 됨
private void Form1_Load(object sender, EventArgs e)
{
    SmartGPIO_Parsing(255);            // 초기 Data값 All High Setting
    smartGPIO1.PORTBDIRS = 0;          // PORTB GPIO 방향설정 (ALL INPUT)
    smartGPIO1.PORTBDATAS = 0;         // PORTB DATA 설정      (ALL LOW)
    smartGPIO1.PortBWatchStart();      // PORTB DATA변경 이벤트 스타트
    …중략…
}
```

[STEP-2] Port B의 상태가 변경되면 smartGPIO1_EvtPortBDatas Change 이벤트가 호출

```
// PORTB의 상태변경이 감지되었을 때 발생되는 이벤트
private void smartGPIO1_EvtPortBDatasChange(object sender, EventArgs e)
{
    int iPortDatas;
    // 이벤트 인자로 받은 데이터를 포트의 상태를 얻기 위해 형변환한다.
    SmartX.PORTDataEvtArgs PortDatas;
    // EventArgs e 에는 포트의 데이터를 직접 접근할 수 없으므로
    // SmartX.PORTDataEvtArgs로 변환한다.
    // ePortDatas.iPortDatas 필드가 있는데 이 값을 읽으면 포트의 상태를 얻을 수 있다.
    PortDatas = (SmartX.PORTDataEvtArgs)e;
    // iPortDatas = 포트0~8번까지의 값(0~255)
    iPortDatas = PortDatas.iPortDatas;
    // PORTB의 이벤트 처리함수
    SmartGPIO_Parsing(iPortDatas);
}
```

[STEP-3] Port B의 이벤트 처리함수

```
// PORTB의 상태변경 이벤트 처리함수
private void SmartGPIO_Parsing(int iPortDatas)
{
    // Input0 체크. 스위치(SW)
    if ((iPortDatas & 0x01) == 0x01)                    // PortB의 0번 비트가 1이면
    {
        LblCnt0.Text = InputCount[0].ToString();        // 카운터를 표시만 해줌
        SD0.SetBackimage = IMG_SW_1;                    // OFF 이미지 표시
    }
    else            // PortB의 0번 비트가 0이면 < 버튼이나 센서가 작동>
    {
        LblCnt0.Text = (InputCount[0] += 1).ToString();     // 카운터를 +1해서 표시해줌
        SD0.SetBackimage = IMG_SW_0;                        // ON 이미지표시
    }
… 중략 …
```

5-2) VB.NET 예제 전체소스 코드

본 예제의 회로도에서 INPUT0은 스위치, INPUT1은 2선식 근접센서, INPUT2는 3선식 PNP 근접센서, INPUT3는 3선식 NPN근접센서로 연결되어 있습니다. 여기서 설명하는 INPUT0은 스위치(SW)이며 스위치 이외의 근접센서(2선식, 3선식)에 관한 설명은 Application Notes의 Smart I/O_Input 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
// 폼 로드시 실행 됨
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    SmartGPIO_Parsing(255)                ' 초기Data값All High Setting
    smartGPIO1.PORTBDIRS = 0              ' PORTB GPIO 방향설정(ALL INPUT)
    smartGPIO1.PORTBDATAS = 0             ' PORTB DATA 설정     (ALL LOW)
    smartGPIO1.PortBWatchStart()          ' PORTB DATA변경 이벤트 스타트
    …중략…
End Sub
```

[STEP-2] Port B의 상태가 변경되면 smartGPIO1_EvtPortBDatas Change 이벤트가 호출

```
' PORTB의 상태 변경이 감지되었을 때 발생되는 이벤트
Private Sub smartGPIO1_EvtPortBDatasChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
smartGPIO1.EvtPortBDatasChange
    Dim iPortDatas As Integer
    ' 이벤트인자로 받은 데이터를 포트의 상태를 얻기 위해 형변환한다.
    Dim PortDatas As SmartX.PORTDataEvtArgs

    ' EventArgs e 에는포트의 데이터를 직접 접근할 수 없으므로
    ' SmartX.PORTDataEvtArgs로 변환한다.
    ' ePortDatas.iPortDatas 필드가 있는 데이값을 읽으면 포트의 상태를 얻을수있다.
    PortDatas = e

    ' iPortDatas = 포트~8번까지의 값(0~255)
    iPortDatas = PortDatas.iPortDatas
    ' PORTB의 이벤트 처리 함수
    SmartGPIO_Parsing(iPortDatas)
End Sub
```

[STEP-3] Port B의 이벤트 처리함수

```
' PORTB의 상태변경 이벤트 처리함수
Private Sub SmartGPIO_Parsing(ByVal iPortDatas As Integer)
    ' Input0 체크
    If ((iPortDatas And &H1) = &H1) Then                    ' PortB의 0번비트가 1이면
        LblCnt0.Text = InputCount(0)                        ' 카운터를 표시만해 줌
        SD0.SetBackimage = IMG_SW_1                         ' OFF 이미지표시
    Else                                                    ' PortB의 0번비트가 0이면 <버튼이나 센서가 작동>
        InputCount(0) = InputCount(0) + 1
        LblCnt0.Text = InputCount(0)                        ' 카운터를 +1해서 표시해 줌
        SD0.SetBackimage = IMG_SW_0                         ' ON 이미지표시
    End If
… 중략 …
```

5-3) C++ 예제 전체소스 코드

CPP 예제 소스 코드는 별도로 제공하지 않습니다. SmartX Framework 관련 예제를 참고하시기 바랍니다.

자료위치 안내 : 자사홈페이지(www.hnsts.co.kr) → 자료실 → SmartX 관련자료 → SmartX Framework 예제파일 → SmartX_Example_C++ → SmartGPIOEVC

[표1] Port 핀과 모드, iPortDatas 값 설명 (0번 비트가 1인 경우)

Port 핀	7	6	5	4	3	2	1	0
모드	0	0	0	0	0	0	0	1
iPortDatas 값	0x01(16진수)							

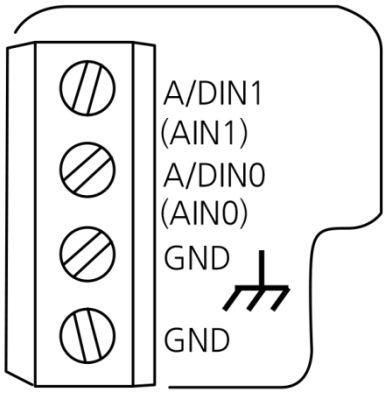
[참고] GPIO관련 자세한 설명은 SmartX Programming Guide의 SmartGPIO편을 참고 바랍니다.

2. A/D(Analog to Digital) 입력단자

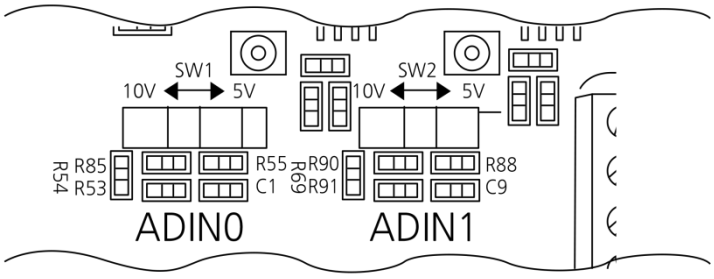
1) A/D(Analog to Digital) 입력단자 소개

A/D INPUT은 Analog Optocoupler를 사용한 것이 특징입니다. 따라서 입력 측(A/D INPUT)과 IEC-Series의 AIN0/AIN1간 전원분리(Isolation)가 되어 있습니다. A/D 입력전압은 DC 0 ~ 5V 또는 DC 0 ~ 10V를 사용하며 IEC266은 10bit(1024), IEC667은 12bit(4096), IEC1000은 12bit(4096)의 해상도를 가지고 있습니다. A/D IN은 IEC-Series의 AIN0/AIN1에 다음과 같이 연결 되어 있습니다.

외부입력단자	A/D IN0	A/D IN1
내부 Extension Port 연결단자	AIN 0	AIN 1



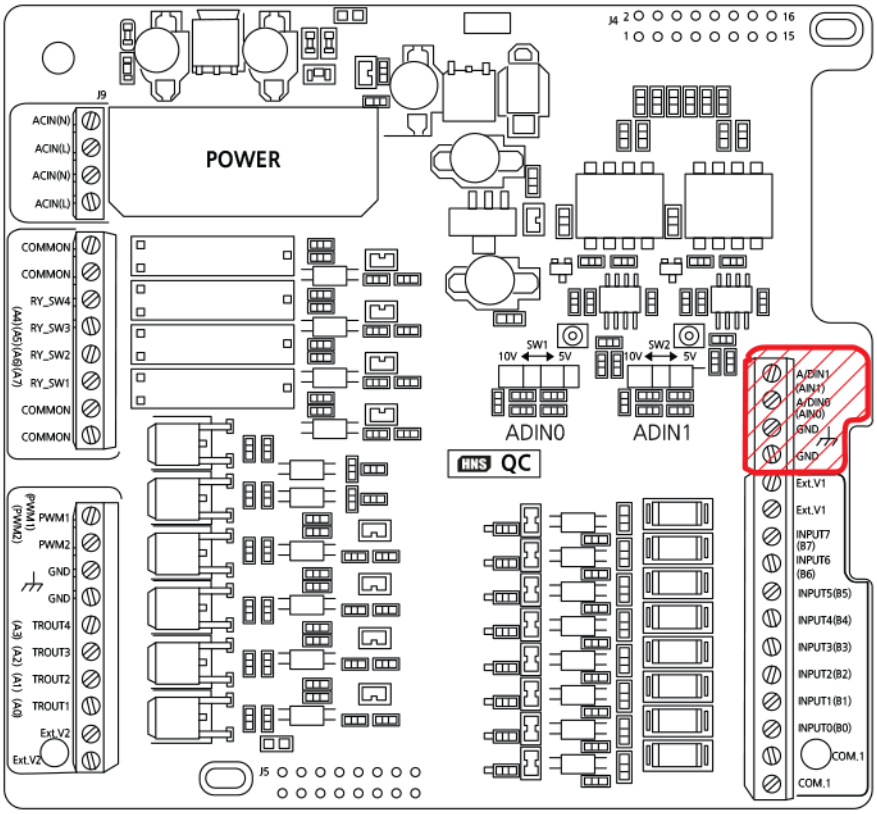
[A/D IN 커넥터]



[입력 선택 스위치 (10V ⇔ 5V)]

[주의]	포토커플러 입력은 무극성이며 COM.1 내부는 공통으로 묶여 있습니다.
	외부입력전원(멀티 탭) 사용시 접지가 있는 것을 사용해야 합니다. 접지가 없는 경우 ADC의 입력 값이 많이 흔들릴 수 있습니다. Extension 2 케이블의 길이가 길어지는 경우 노이즈의 영향으로 신호가 왜곡될 수 있습니다

2) A/D(Analog to Digital) 입력단자 위치

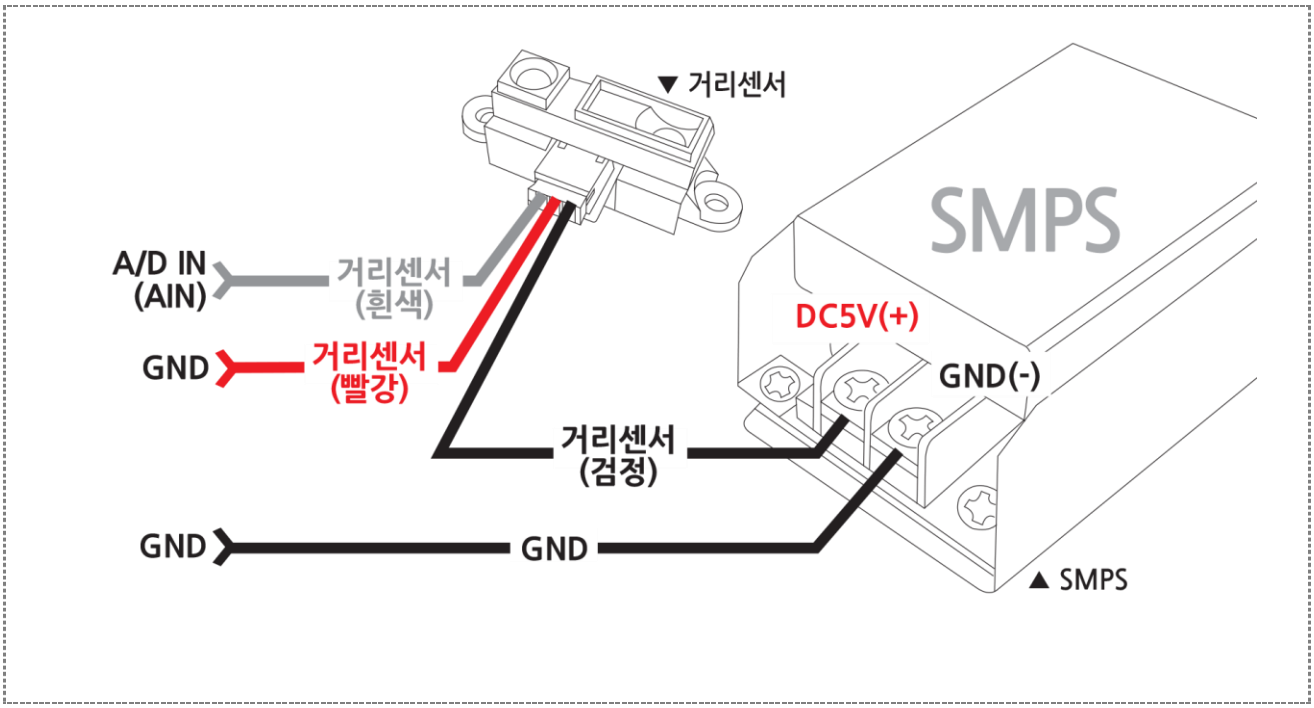


3) A/D(Analog to Digital) 입력단자 응용방법

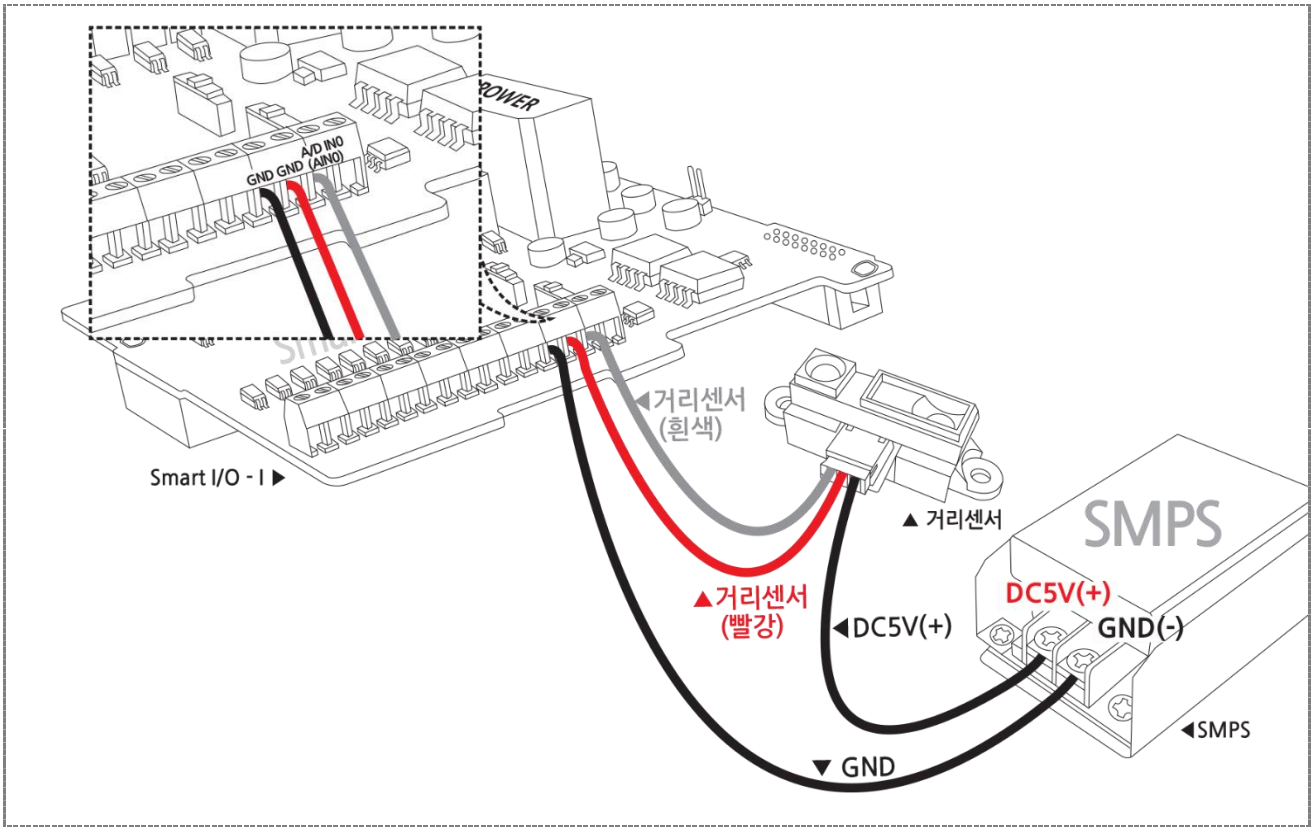
3-1) 거리센서 연결방법

아래와 같이 결선을 하시고, Smart I/O_ADC의 ADC0항목에서 확인가능

① 결선도



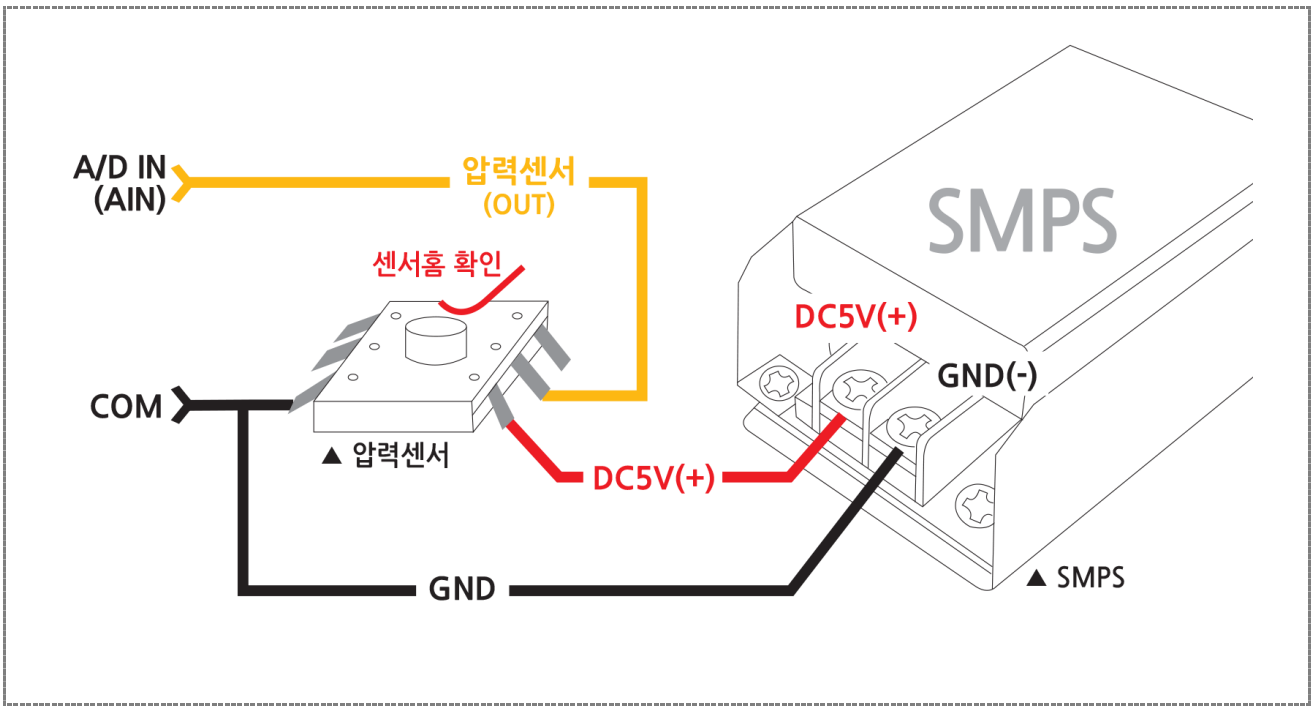
② SmartI/O - I Base Board 결선도 예



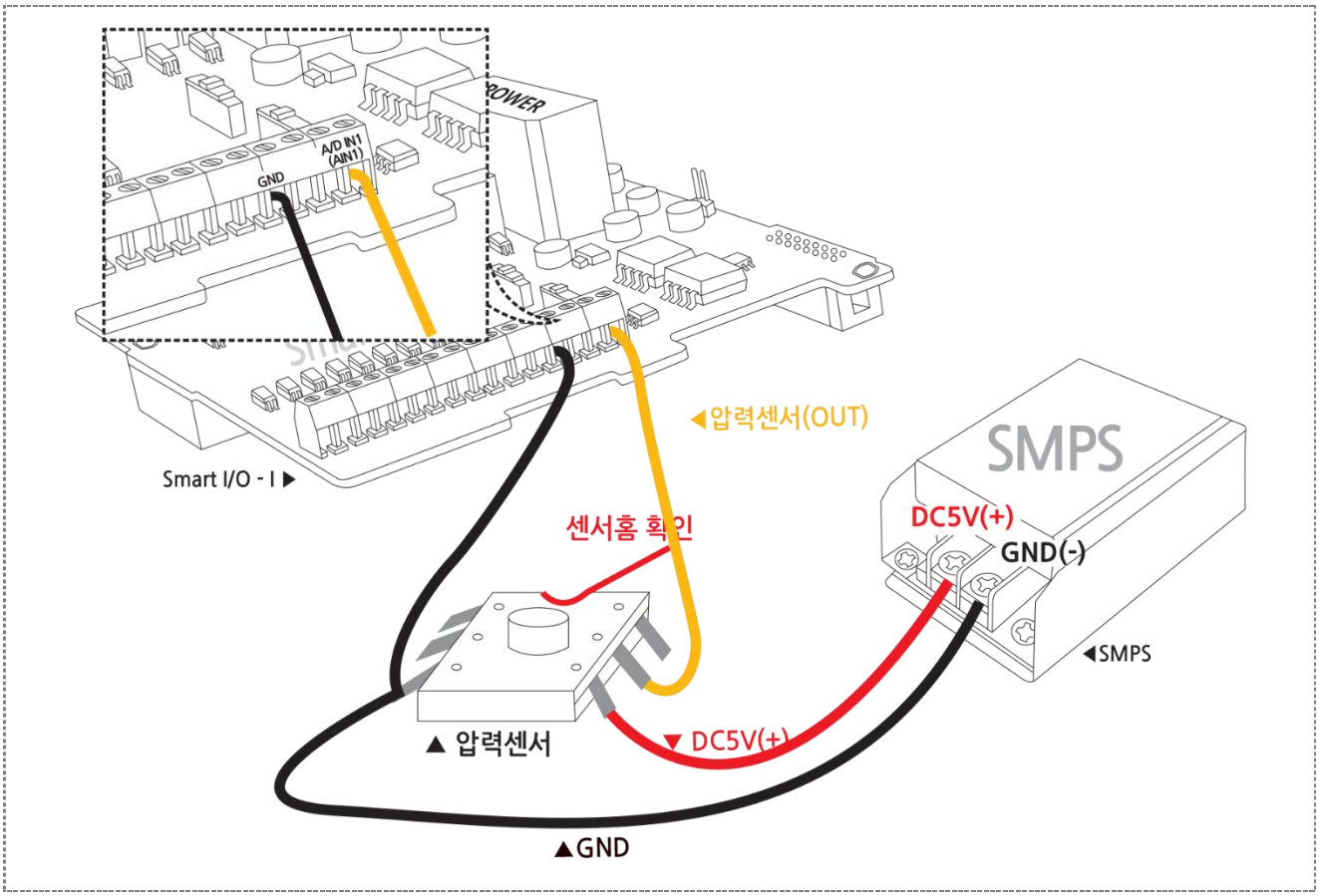
3-2) 압력센서 연결방법

아래와 같이 결선을 하시고, Smart I/O_ADC의 ADC1항목에서 확인가능

① 결선도



② SmartI/O - I Base Board 결선도 예



4) 언어별 주요소스 코드

C#

```
// Adc 값 읽어서 표시하는 Timer
private void smartTimer1_Tick(object sender, EventArgs e)
{
    int iVal = 0, iVal1 = 0, iVal2 = 0, iVal3 = 0;
    double iADCVal = 0, iADCVal1 = 0, iADCVal2 = 0, iADCVal3 = 0;

    if (ChkADC0 == true)
    {
        iVal = smartADC1.ReadData(0);              // ADC0 데이터를 얻어온다.
        Adc0_DataParsing(iVal);                    // 거리측정계산 처리코드로 개발자가 직접 작성
    }
…중략…
}
```

VB.NET

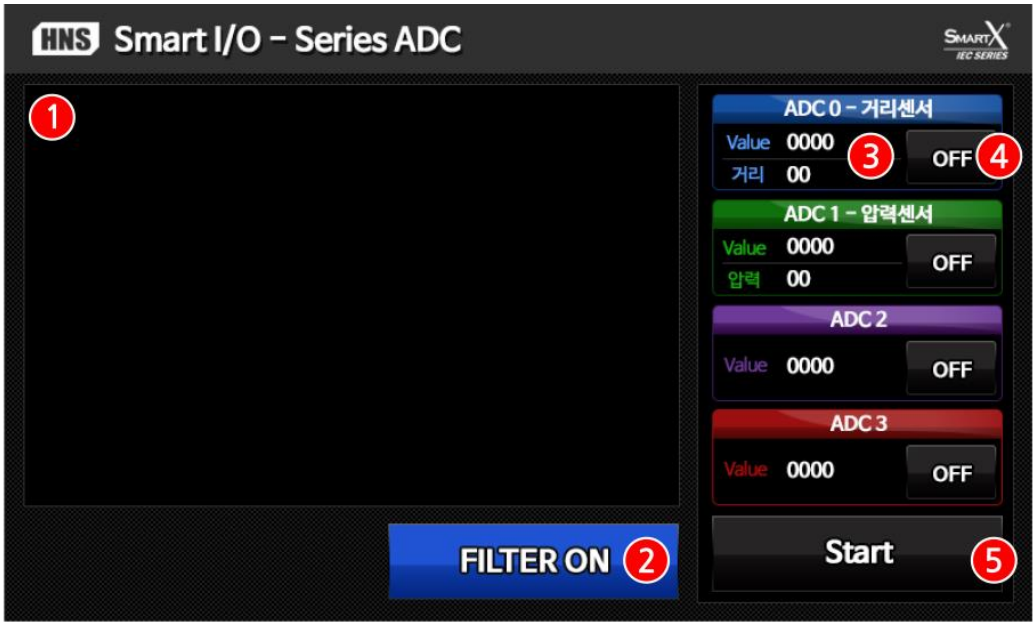
```
' Adc 값 읽어서 표시하는 Timer
Private Sub smartTimer1_Tick(ByVal sender As Object, ByVal e As EventArgs)
    Dim iVal As Integer = 0, iVal1 As Integer = 0, iVal2 As Integer = 0, iVal3 As Integer = 0
    Dim iADCVal As Double = 0, iADCVal1 As Double = 0, iADCVal2 As Double = 0, iADCVal3 As Double = 0

    If ChkADC0 = True Then
        iVal = smartADC1.ReadData(0)     ' ADC0 데이터를 얻어온다.
        Adc0_DataParsing(iVal)           ' 거리측정계산 처리코드로 개발자가 직접 작성
    End If
…중략…
End Sub
```

C++

```
void CSmartADCEVCDlg::OnTimer(UINT nIDEvent)
{    // Timer 식별자가 3인경우. SetTimer(3, 2000, NULL);에서 식별자 지정
    if(nIDEvent == 3)
    {
        // 선택된채널의ADC 값을 얻는다
        m_iNowADC = m_SmartADC.GetADCData(iSelCh);
        Adc0_DataParsing(m_iNowADC)      //거리측정계산 처리코드로 개발자가 직접 작성
        CDialog::OnTimer(nIDEvent);
    }
}
```

5) Application 응용예제

SmartX Framework	SmartADC
소스파일	Smart I/O_ADC
소스경로	홈페이지 자료실 -> Application Note -> Smart I/O 활용
기 능	온습도, 근접, 압축, 가속도, 자이로센서등 Analog센서값 처리
응용분야	센서사용 측정분야
준비사항	▪ 거리센서 : SHARP(GP2Y0A41SK0F) ▪ 압력센서 : Smate(33A-030G-2210)



❶ ADC 값 그래프로 표시 ❷ ADC Filtering ON/OFF ❸ 각 ADC값 표시 부분
❹ 각 ADC ON/OFF ❺ 전체 ADC ON/OFF

[동작설명]

▶ 프로그램이 시작되면서 ②의 FILTER ON버튼을 클릭하면 Filtering속성을 Enable로 활성화하고 일정 범위 이외의 데이터 값을 무시한 데이터의 평균을 구합니다.

▶ ⑤의 Start 버튼을 클릭하면 SmartTimer를 Start합니다. SmartTimer_Tick 이벤트 에서는 ADC0채널~ADC3채널까지를 읽는 코드가 Tick이벤트 실행할 때마다 호출됩니다.

▶ ④ 의 OFF 버튼을 클릭하면 해당 ADC채널의 smartADC1.ReadData(채널넘버); 코드가 활성화(Active) 됩니다. OFF 버튼은 해당 채널 별로 존재하며 각 버튼의 OFF/ON은 해당채널만 활성화 / 비활성화 시킵니다. 즉 ADC0과 ADC1을 ON하는 경우 ADC0과 ADC1채널에서 값을 읽어옵니다. ADC2와 ADC3은 OFF 되어 값을 읽지 않습니다.

▶ ⑤ 클릭 → ④ 클릭하여 ON 후 해당 채널에서 아날로그 데이터가 입력되는 경우 ③의 디지털(Digital) 값으로 표시됩니다.
해당 디지털 값은 SmartDraw1.PutData()를 사용하여 ①에 차트로 그려줍니다.

5-1) C#예제 전체소스 코드

본 예제의 회로도에서 ADC 0은 거리센서, ADC 1은 압력센서로 연결되어 있습니다.
여기서 설명하는 ADC 0은 거리센서이며 거리센서 이외의 압력센서에 관한 설명은 Application Notes의 Smart ADC 예제 소스를 참고 바랍니다.

[STEP-1] 모든 채널(4ch) ADC Read Start / Stop 설정

```
// All Adc Read/Stop 설정
private void BtnAdcStart_Click(object sender, EventArgs e)
{
    // ADC Start
    if (BtnAdcStart.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartTimer1.Start();
    }
    // ADC Stop
    else if (BtnAdcStart.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartTimer1.Stop();
    }
}
```

[STEP-2] ADC 값 읽어서 표시하는 Timer

```
private void smartTimer1_Tick(object sender, EventArgs e)
{
    int iVal = 0, iVal1 = 0, iVal2 = 0, iVal3 = 0;
    double iADCVal = 0, iADCVal1 = 0, iADCVal2 = 0, iADCVal3 = 0;
    if (ChkADC0 == true)
    {
        iVal = smartADC1.ReadData(0);              // ADC0 데이터를 얻어온다.
        Adc0_DataParsing(iVal);                    // 거리측정계산 함수
    }
    if (ChkADC1 == true)
    {
        iVal1 = smartADC1.ReadData(1);            // ADC1 데이터를 얻어온다.
    }
    if (ChkADC2 == true)
    {
        iVal2 = smartADC1.ReadData(2);            // ADC2 데이터를 얻어온다.
    }
    if (ChkADC3 == true)
    {
        iVal3 = smartADC1.ReadData(3);            // ADC3 데이터를 얻어온다.
    }

    // 그래프에 4채널의 데이터를 표시 하기위한 계산
    // IEC266은 10bit(1024)
    if (ChkIEC == 1)
    {
        iADCVal = ((double)iVal / 1024);
        iADCVal1 = ((double)iVal1 / 1024);
        iADCVal2 = ((double)iVal2 / 1024);
        iADCVal3 = ((double)iVal3 / 1024);
    }
```

```
        // IEC667, IEC1000은 12bit(4096)
        else if (ChkIEC == 2)
        {
            iADCVal = ((double)iVal / 4096);
            iADCVal1 = ((double)iVal1 / 4096);
            iADCVal2 = ((double)iVal2 / 4096);
            iADCVal3 = ((double)iVal3 / 4096);
        }

        iVal = (int)((double)270.0 * iADCVal);
        iVal1 = (int)((double)270.0 * iADCVal1);
        iVal2 = (int)((double)270.0 * iADCVal2);
        iVal3 = (int)((double)270.0 * iADCVal3);

        // 그래프의 데이터를 입력 합니다. (인자의 수에 맞추어 그래프의 각각의 계열에 적용됨)
        smartDraw1.PutData(iVal, iVal1, iVal2, iVal3);
    }
```

[STEP-3] ADC0 센서값을 거리로 계산해주는 메서드(센서의 구간이 선형적일 때 적용)

```
// 거리센서의 종류마다 특성이 다르므로 계산공식을 센서에 맞게 적용하시기 바랍니다.
// 센서 특성상 선형적이지 않을경우 구간별로 측정해야 합니다.
// ADC0 센서값을 거리로 계산해주는 공식. 20cm ~ 50cm 구간 사이 변환 공식
// 본 코드는 AD값으로 거리를 계산하는 공식이며 AD값에 따라 거리값의 증/감이 비교적 비례하는 구간을 계산한 공식입니다.
// AD값의 구간별로 정확하게 거리를 계산하기위해서는 AD구간별로 계산공식이 필요합니다.
private void Adc0_DataParsing(int val)
{
    // 거리센서의 ADC값 = 1064는 거리20cm, ADC값 = 491은 거리50cm 입니다.
    // ADC값 1064 - 491 = 573(ADC구간) 이고 20cm ~ 50cm(ADC의 거리구간)
    // 계산공식은 573 : 30 = (1064 - val) : x 의 결과값에 20(출발점)을 더합니다.
    LblAdc0_1.Text = ((((1064 - val) * 30) / 573) + 20).ToString() + "cm";      // 거리값
    LblAdc0.Text = val.ToString();      // ADC값. int형으로 형변환
}
```

5-2) VB.NET 예제 전체소스 코드

본 예제의 회로도에서 ADC 0은 거리센서, ADC 1은 압력센서로 연결되어 있습니다.
여기서 설명하는 ADC 0은 거리센서이며 거리센서 이외의 압력센서에 관한 설명은 Application Notes의 Smart ADC 예제 소스를 참고 바랍니다.

[STEP-1] 모든 채널(4ch) ADC Read Start / Stop 설정

```
' All ADC Read/Stop 설정
Private Sub BtnAdcStart_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAdcStart.Click
    ' ADC Start
    If BtnAdcStart.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        SmartTimer1.Start()
    ' ADC Stop
    ElseIf BtnAdcStart.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        SmartTimer1.Stop()
    End If
End Sub
```

[STEP-2] ADC 값 읽어서 표시하는 Timer

```
' ADC 값 읽어서 표시하는 Timer
Private Sub SmartTimer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)

    Dim iVal As Integer = 0, iVal1 As Integer = 0, iVal2 As Integer = 0, iVal3 As Integer = 0
    Dim iADCVal As Double = 0, iADCVal1 As Double = 0, iADCVal2 As Double = 0, iADCVal3 As Double = 0

    If ChkADC0 = True Then
        iVal = SmartADC1.ReadData(0)          ' ADC0 데이터를 얻어온다.
        Adc0_DataParsing(iVal)                ' 거리측정계산 함수
    End If
    If ChkADC1 = True Then
        iVal1 = SmartADC1.ReadData(1)         ' ADC1 데이터를 얻어온다.
    End If
    If ChkADC2 = True Then
        iVal2 = SmartADC1.ReadData(2)         ' ADC2 데이터를 얻어온다.
    End If
    If ChkADC3 = True Then
        iVal3 = SmartADC1.ReadData(3)         ' ADC3 데이터를 얻어온다.
    End If

    ' ADC 값 표시
    LblAdc0.Text = iVal.ToString()
    LblAdc1.Text = iVal1.ToString()
    LblAdc2.Text = iVal2.ToString()
    LblAdc3.Text = iVal3.ToString()

    ' 그래프에 4채널의 데이터를 표시 하기위한 계산
    ' IEC266은 10bit(1024)
    If ChkIEC = 1 Then
        iADCVal = (CDbl(iVal) / 1024)
        iADCVal1 = (CDbl(iVal1) / 1024)
        iADCVal2 = (CDbl(iVal2) / 1024)
        iADCVal3 = (CDbl(iVal3) / 1024)
        ' IEC667, IEC1000은 12bit(4096)
    ElseIf ChkIEC = 2 Then
```

```
            iADCVal = (CDbl(iVal) / 4096)
            iADCVal1 = (CDbl(iVal1) / 4096)
            iADCVal2 = (CDbl(iVal2) / 4096)
            iADCVal3 = (CDbl(iVal3) / 4096)
        End If

        iVal = CInt(CDbl(270.0) * iADCVal)
        iVal1 = CInt(CDbl(270.0) * iADCVal1)
        iVal2 = CInt(CDbl(270.0) * iADCVal2)
        iVal3 = CInt(CDbl(270.0) * iADCVal3)

        ' 그래프의 데이터를 입력 합니다. (인자의 수에 맞추어 그래프의 각각의 계열에 적용됨)
        smartDraw1.PutData(iVal, iVal1, iVal2, iVal3)
    End Sub
```

[STEP-3] ADC0 센서값을 거리로 계산해주는 메서드(센서의 구간이 선형적일 때 적용)

```
' 거리센서의 종류마다 특성이 다르므로 계산공식을 센서에 맞게 적용하시기 바랍니다.
' 센서 특성상 선형적이지 않을경우 구간별로 측정해야 합니다.
' ADC0 센서값을 거리로 계산해주는 공식. 20cm ~ 50cm 구간 사이 변환 공식
' 본 코드는 AD값으로 거리를 계산하는 공식이며 AD값에 따라 거리값의 증/감이 비교적 비례하는 구간을 계산한 공식입니다.
' AD값의 구간별로 정확하게 거리를 계산하기위해서는 AD구간별로 계산공식이 필요합니다.
Private Sub Adc0_DataParsing(ByVal val As Double)
    ' 거리센서의 ADC값 = 1064는 거리20cm, ADC값 = 491은 거리50cm 입니다.
    ' ADC값 1064 - 491 = 573(ADC구간) 이고 20cm ~ 50cm(ADC의 거리구간)
    ' 계산공식은 573 : 30 = (1064 - val) : x 의 결과값에 20(출발점)을 더합니다.
    LblAdc0_1.Text = Math.Round(((((1064 - val) * 30) / 573) + 20), 2).ToString() + "cm"      ' 거리값
    LblAdc0.Text = CInt(val).ToString() ' ADC값. int형으로 형변환
End Sub
```

※ ADC0번 거리센서

센서 특성상 선형적이지 않을 경우 구간별로 측정해야 합니다. 본 예제 소스에서 사용한 거리센서는 일부 구간만이 선형적인 관계로 해당 부분을 계산공식으로 표현하였으며 거리센서의 전체구간을 표현하려면 데이터시트를 참고하여 구간별로 측정하여야 합니다.

※ ADC1번 압력센서

센서를 불거나 흡입하면 센서 값이 바뀝니다.

5-3) C++ 예제 전체소스 코드

CPP 예제 소스 코드는 별도로 제공하지 않습니다. SmartX Framework 관련 예제를 참고하시기 바랍니다.

자료위치 안내 : 자사홈페이지(www.hnsts.co.kr) → 자료실 → SmartX 관련자료 → SmartX Framework 예제파일 → SmartX_Example_C++ → SmartADCEVC

[참고] ADC관련 자세한 설명은 SmartX Programming Guide의 SmartADC편을 참고 바랍니다.

3. FET 출력단자

1) FET 출력단자 소개

N-Channel FET를 사용하였으며, 회로와 같이 드레인에 연결이 되어있습니다. 구동하고자하는 부하에 따라 별도의 전원이 필요합니다.

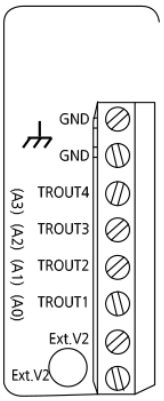
외부입력단자	TROUT 1	TROUT 2	TROUT 3	TROUT 4
내부 Extension Port 연결단자	PORTA 0	PORTA 1	PORTA 2	PORTA 3

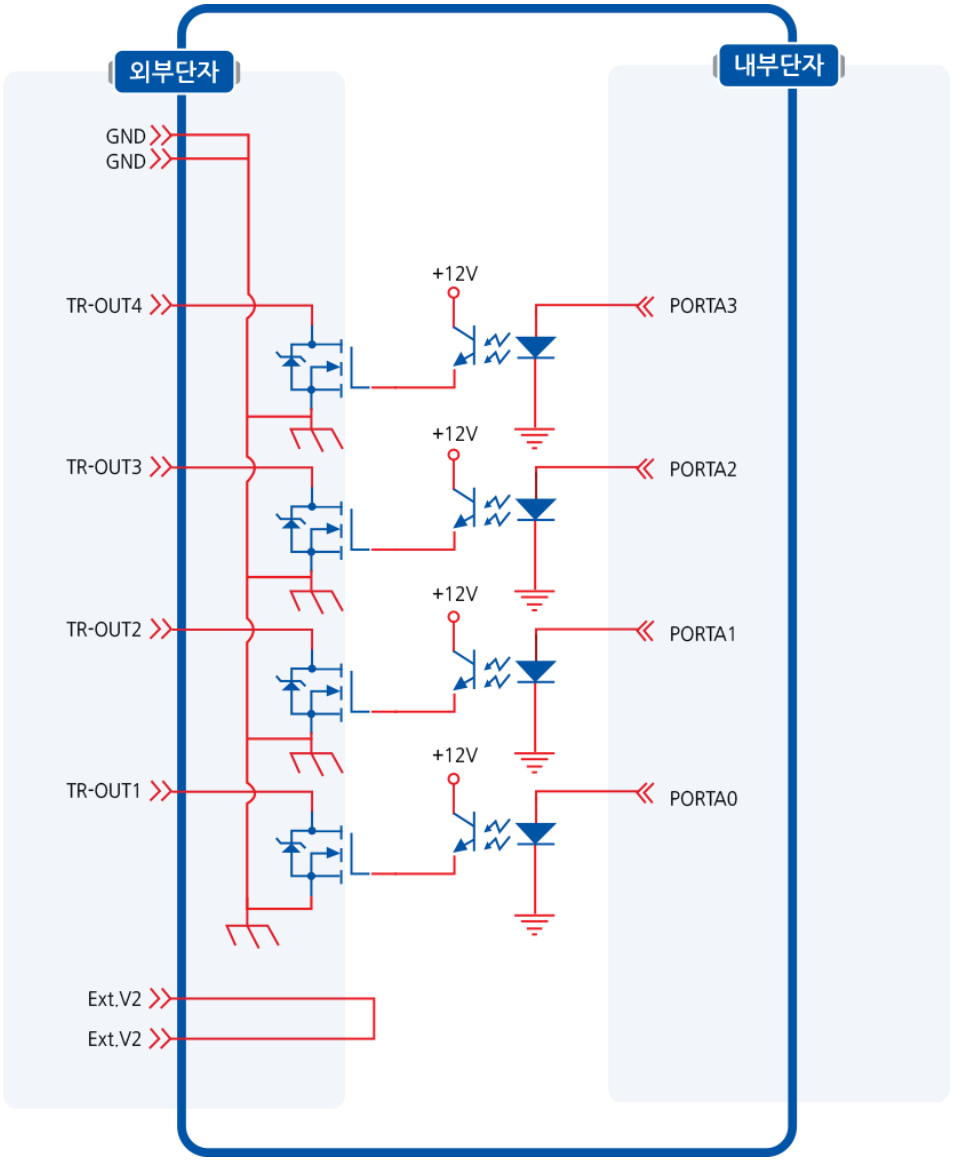
Direction	출력상태(출력)	PortData 값 True/False	LED Status
출력	ON	True	ON
	OFF	False	OFF

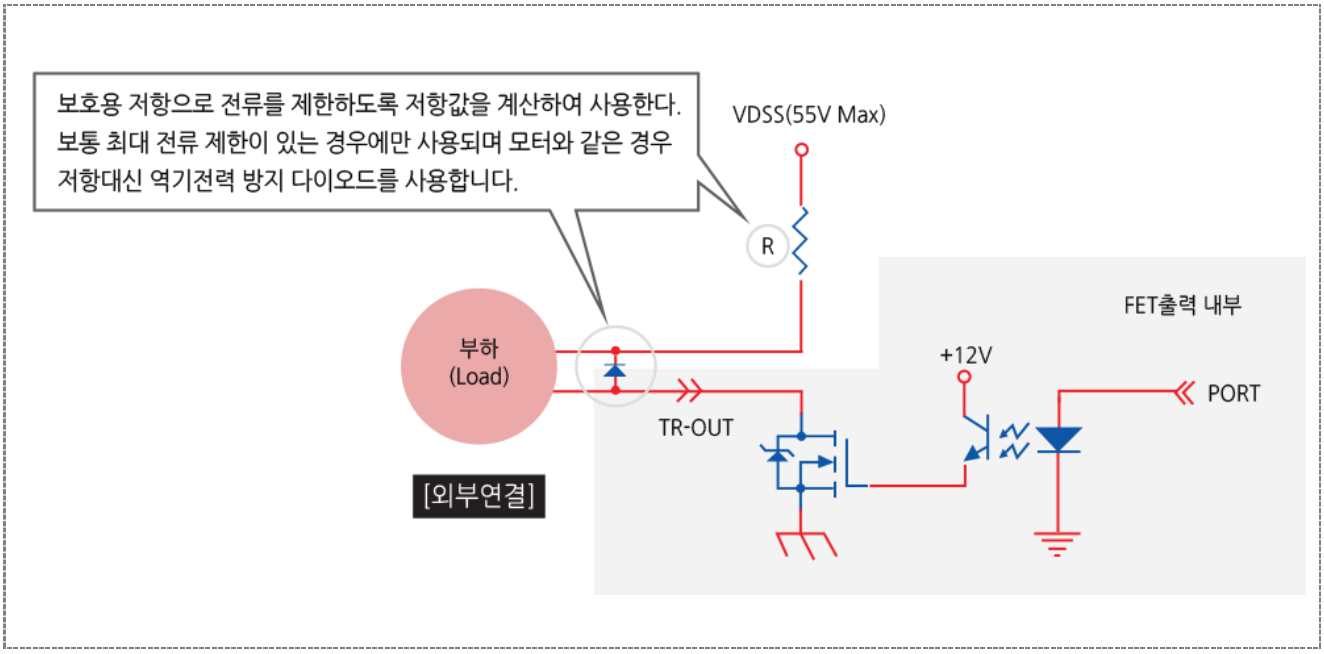
[주의]

[발열에 따른 주의사항]

전력량에 따라 FET 소자에 발열이 발생할 수 있으며, 발열량을 검토하여 전력량을 줄여 발열이 적게 발생하도록 하여 사용하시기 바랍니다.

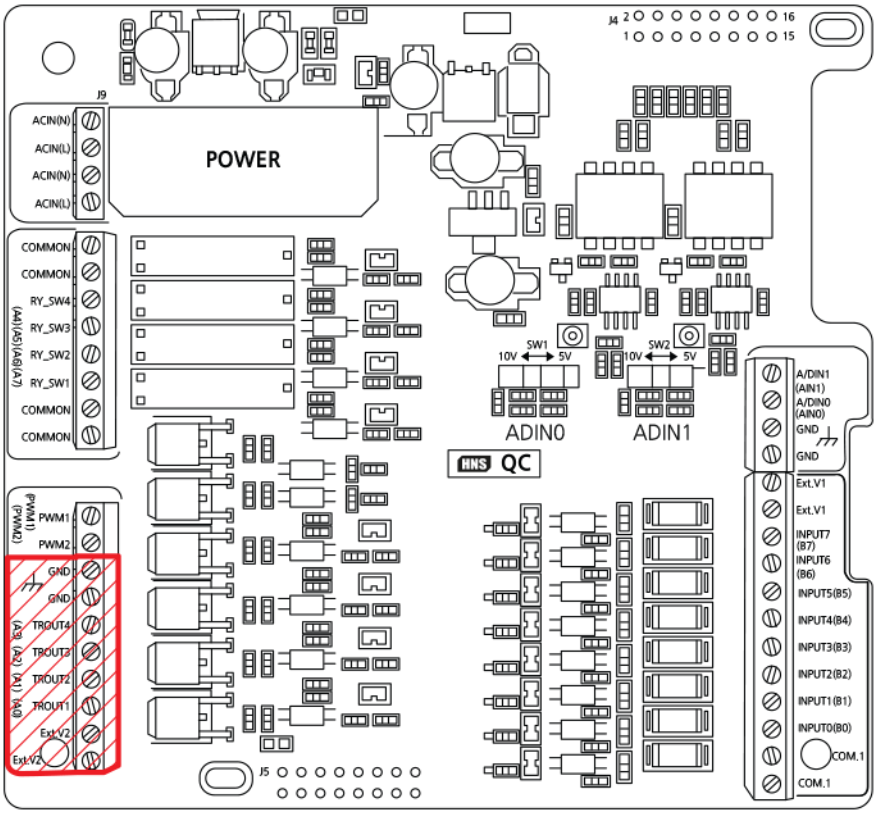






[주의]	IRFR024N은 VDSS = 55V, ID= 17A 정격의 FET입니다.

2) FET 출력단자 위치

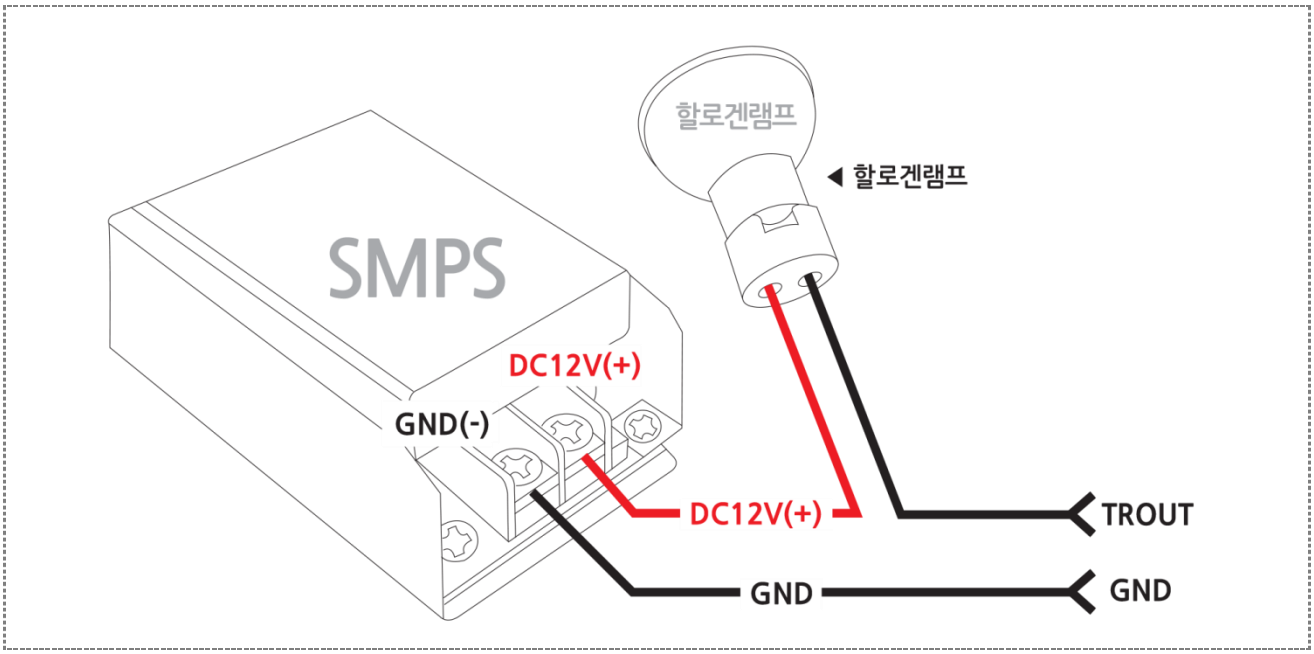

POWER
ACIN(N)
ACIN(L)
ACIN(N)
ACIN(L)
COMMON
COMMON
RY_SW4
RY_SW3
RY_SW2
RY_SW1
COMMON
COMMON
PWM1
PWM2
GND
GND
TROUT4
TROUT3
TROUT2
TROUT1
Ext.V2
Ext.V2
SW1
SW2
10V
5V
ADIN0
ADIN1
QC
A/DIN1
(AIN1)
A/DIN0
(AIN0)
GND
GND
Ext.V1
Ext.V1
INPUT7
(B7)
INPUT6
(B6)
INPUT5(B5)
INPUT4(B4)
INPUT3(B3)
INPUT2(B2)
INPUT1(B1)
INPUT0(B0)
COM.1
COM.1

3) FET 출력단자 응용방법

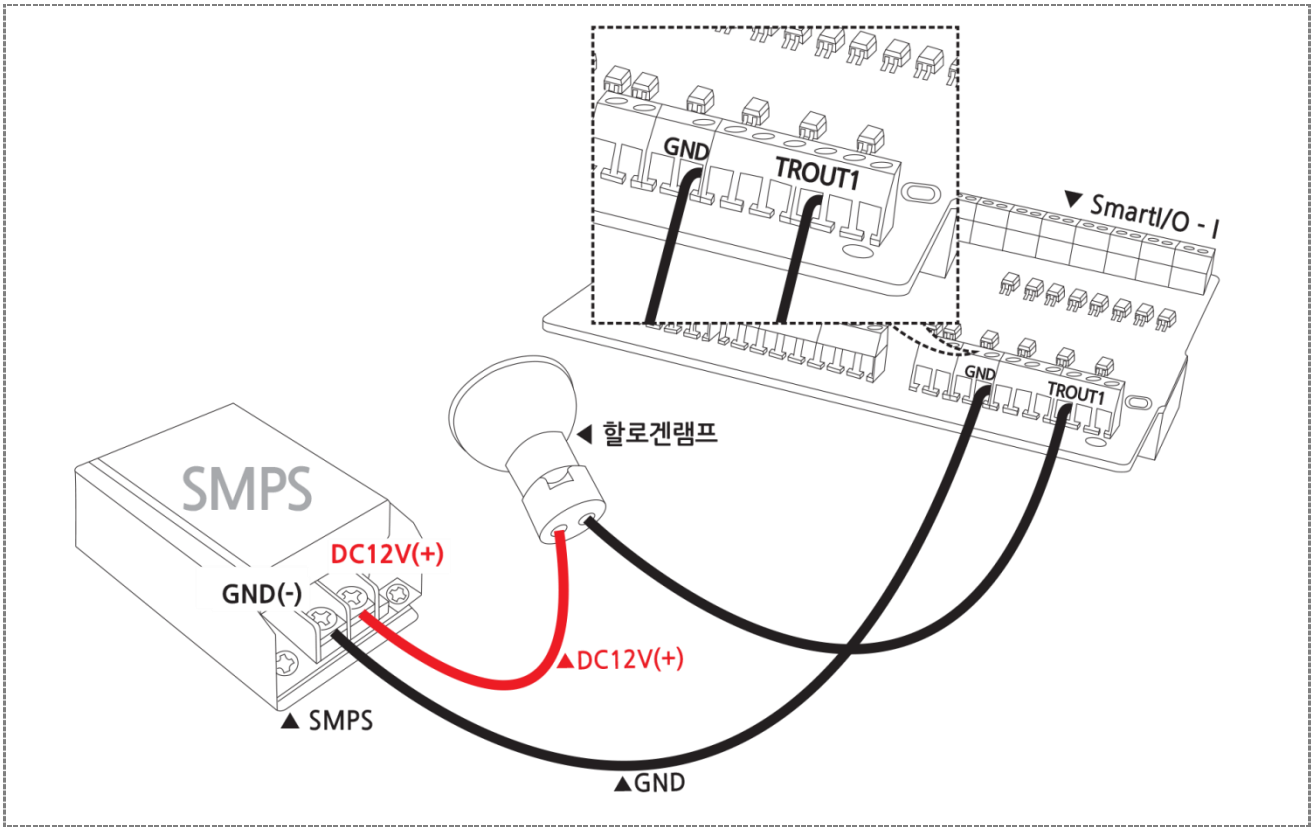
3-1) 할로겐램프 연결방법

아래와 같이 결선을 하시고, Smart I/O_Output의 FET제어(OUT0/TROUT1- DC할로겐) 항목에서 ON/OFF 테스트 가능

① 결선도



② SmartI/O - I Base Board 결선도 예

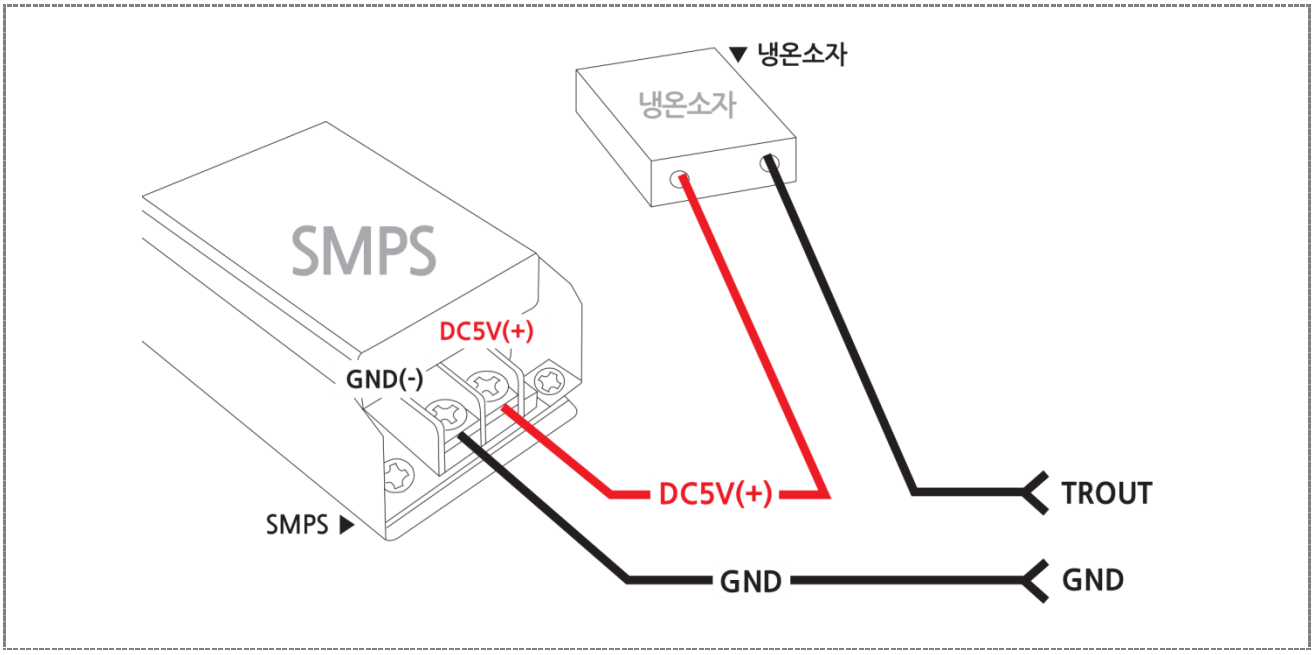


[주의]	할로겐램프는 +/- 구분이 없습니다.

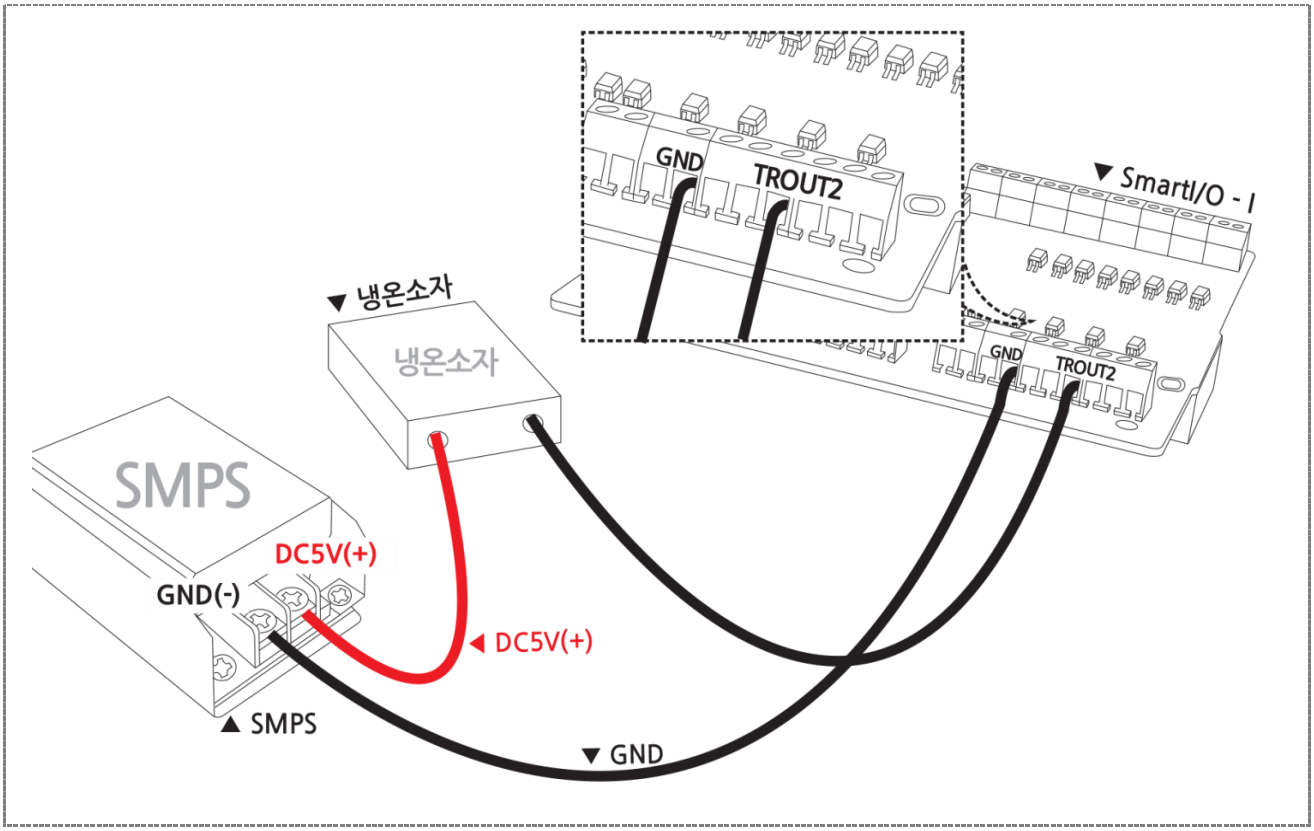
3-2) 냉온소자 연결방법

아래와 같이 결선을 하시고, Smart I/O_Output의 FET제어(OUT1/TROUT2-냉온소자) 항목에서 ON/OFF 테스트 가능

① 결선도



② SmartI/O - I Base Board 결선도 예

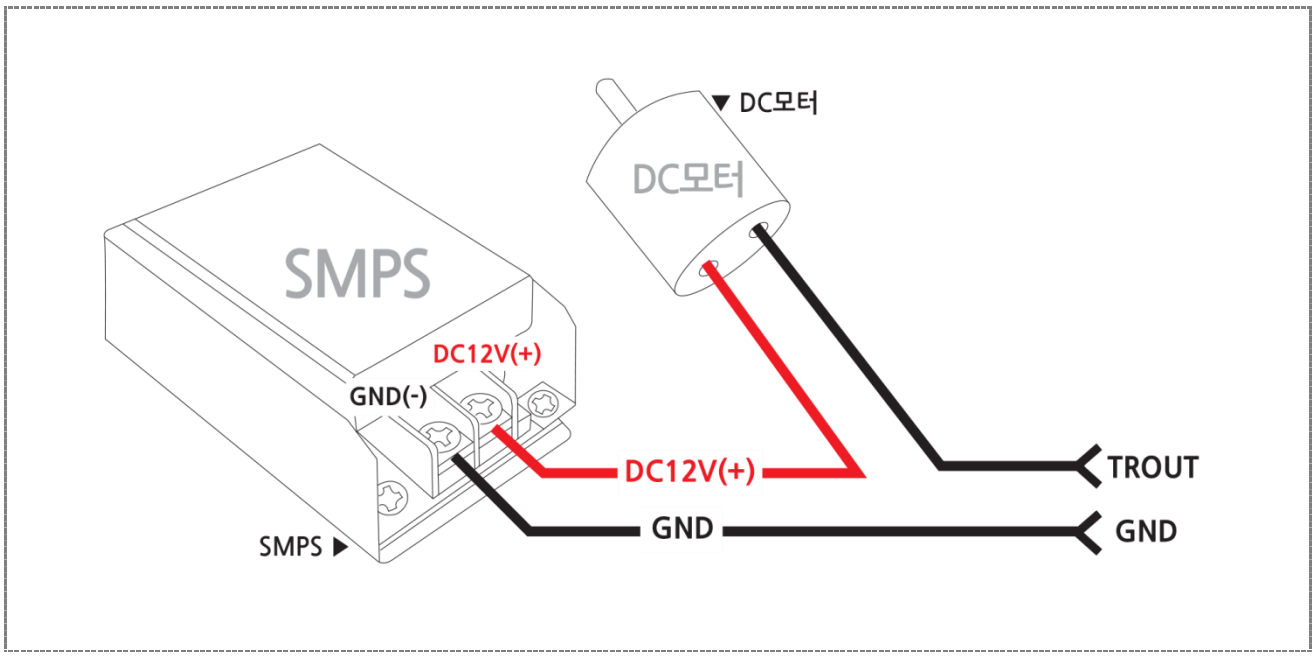


[주의]	전원을 반대로 연결 시 차가운 면과 뜨거운 면으로 바뀝니다.
	뜨거운 면의 열을 방열 판으로 식혀줘야 뜨거운 면이 차가운 면으로 열전도 되는 현상을 막을 수 있습니다.

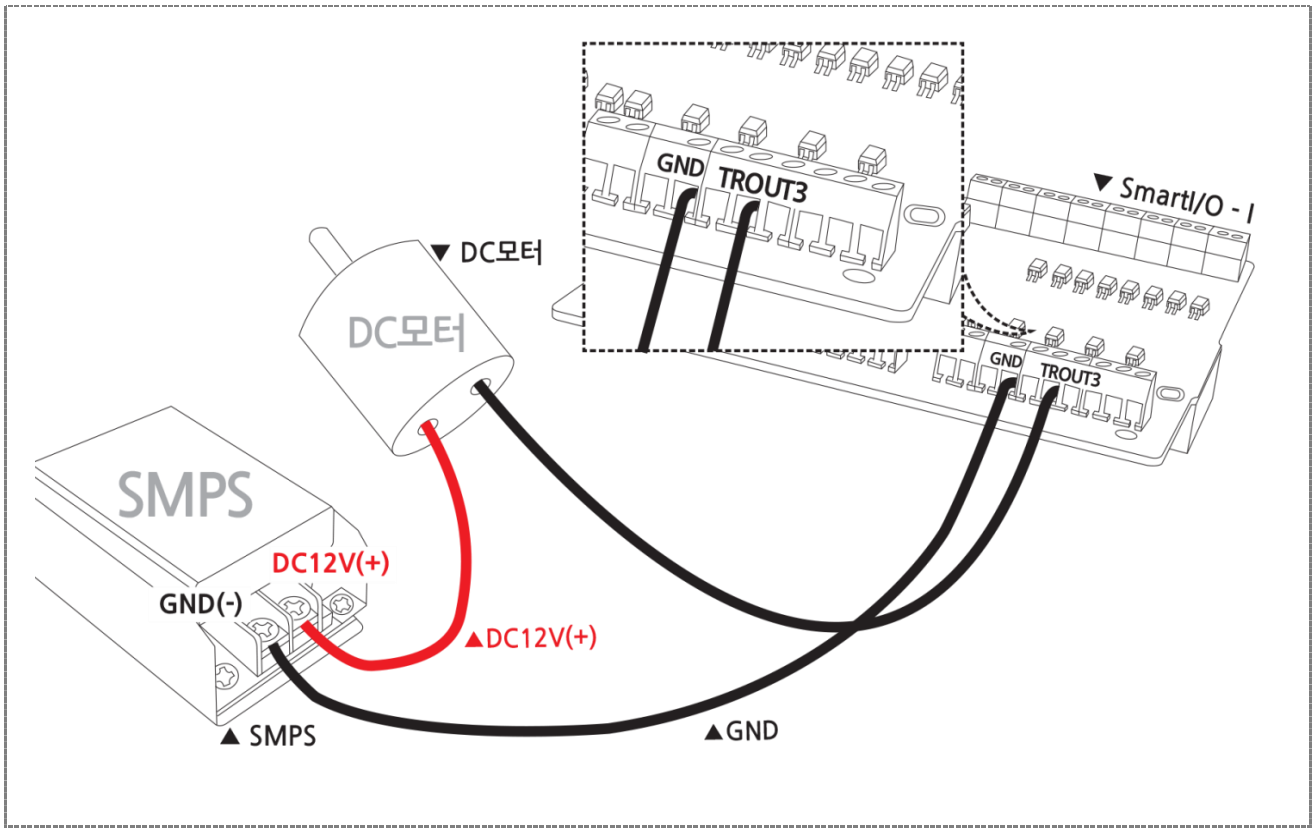
3-3) DC모터 연결방법

아래와 같이 결선을 하시고, Smart I/O_Output의 FET제어(OUT2/TROUT3-DC모터) 항목에서 ON/OFF 테스트가능

① 결선도



② SmartI/O - I Base Board 결선도 예



[주의]	DC모터는 (+/-)를 반대로 연결할 시에 역방향으로 회전합니다.
	Ext.V2 단자 2EA는 공핀입니다. 내부로 Ext.V2끼리 서로 묶여있습니다.

4) 언어별 주요소스 코드

C#

```
// FET(TROUT1) DC할로겐램프 ON/OFF 제어
private void BtnDcLampCtrl_Click(object sender, EventArgs e)
{
    // DC 할로겐램프 ON
    if (BtnDcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartGPIO1.PORTADATA0 = true;
        SDDcLamp.SetBackimage = DcLamp_1;
    }
    // DC 할로겐램프 OFF
    else if (BtnDcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartGPIO1.PORTADATA0 = false;
        SDDcLamp.SetBackimage = DcLamp_0;
    }
}
```

VB.NET

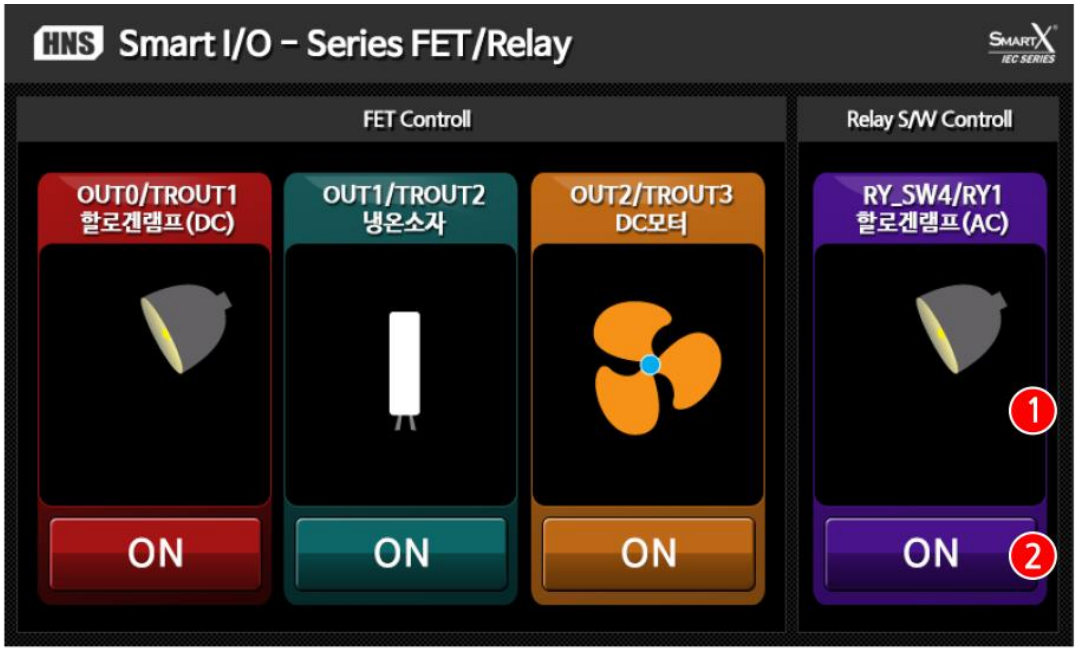
```
Private Sub BtnDcLampCtrl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnDcLampCtrl.Click
    ' DC 할로겐램프ON
    If BtnDcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        smartGPIO1.PORTADATA0 = True
        SDDcLamp.SetBackimage = DcLamp_1
    ' DC 할로겐램프OFF
    ElseIf BtnDcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartGPIO1.PORTADATA0 = False
        SDDcLamp.SetBackimage = DcLamp_0
    End If
End Sub
```

C++

```
// PORT CONTROL STATUS 의 PORT0 버튼을 클릭 시
void CSmartGPIOEVCDlg::OnBnClickedButport0()
{
    Int iOutData;
    if (nID==IDC_BUTTON0)    // PORT0 버튼이 눌러지면
    {
        iOutData = 1;            // DC 할로겐램프ON
    }else {
        iOutData = 0;           // DC 할로겐램프OFF
    }
    m_SmartGPIO.SetPortABit(0, iOutData);
}
```

5) Application 응용예제

SmartX Framework	SmartGPIO
소스파일	Smart I/O_Output
소스경로	홈페이지 [자료실] → [Application Note] → [Smart I/O 활용]
기 능	AC/DC전원으로 구동하는 장비 연동 시 온/오프조작
응용분야	DC할로겐 램프, 냉온소자, DC 모터 등 제어
준비사항	▪ 냉온소자 : (FALC1-00705T150) ▪ 할로겐램프 : (AC/DC 12V, 10W) ※할로겐램프는 AC/DC상관없이 전압만 맞춰주면 됨. ▪ DC모터 : ㈜금일모터(KDG37-3429A-050)



❶ 이미지 표시 창 ❷ FET ON/OFF

[동작설명]

▶ 프로그램이 시작되면서 ② 각 포트의 ON 버튼을 클릭하면 해당 포트 별로 출력(LOW→HIGH)가 발생하면서 ①의 이미지가 변경(이미지 변경 그림 추가)됩니다. 4번째 RY_SW4/RY1 할로겐램프(AC)의 경우에는 Relay 기능을 테스트 가능합니다.

5-1) C#예제 전체소스 코드

본 예제의 회로도에서 Output 0은 할로겐램프, Output 1은 냉온소자, Output 2은 DC모터로 연결되어 있습니다.
여기서 설명하는 Output 0은 할로겐램프이며 할로겐램프 이외의 냉온소자, DC모터에 관한 설명은 Application Notes의 Smart I/O_Output 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
// 폼 로드시 실행됨
private void Form1_Load(object sender, EventArgs e)
{
    smartGPIO1.PORTADIRS = 255;              // PORTA GPIO 방향설정 (ALL OUTPUT)
    smartGPIO1.PORTADATAS = 0;               // PORTA DATA 설정      (ALL LOW)
    …중략…
}
```

[STEP-2] BtnDcLampCtrl 클릭되면 smartGPIO1.PORTADATA0의 상태 값(true/false)을 변경

```
// FET(TROUT1) DC할로겐램프 ON/OFF 제어
private void BtnDcLampCtrl_Click(object sender, EventArgs e)
{
    // DC 할로겐램프 ON
    if (BtnDcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartGPIO1.PORTADATA0 = true;
        SDDcLamp.SetBackimage = DcLamp_1;
     }
    // DC 할로겐램프 OFF
    else if (BtnDcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartGPIO1.PORTADATA0 = false;
        SDDcLamp.SetBackimage = DcLamp_0;
     }
}
```

5-2) VB.NET 예제 전체소스 코드

본 예제의 회로도에서 Output 0은 할로겐램프, Output 1은 냉온소자, Output 2은 DC모터로 연결되어 있습니다.
여기서 설명하는 Output 0은 할로겐램프이며 할로겐램프 이외의 냉온소자, DC모터에 관한 설명은 Application Notes의 Smart I/O_Output 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
' 폼로드시 실행 됨
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    smartGPIO1.PORTADIRS = 255        ' PORTA GPIO 방향설정(ALL OUTPUT)
    smartGPIO1.PORTADATAS = 0         ' PORTA DATA 설정      (ALL LOW)
    …중략…
End Sub
```

[STEP-2] BtnDcLampCtrl 클릭되면 smartGPIO1.PORTADATA0의 상태 값(true/false)을 변경

```
'=======================================
'= FET(TROUT1) DC할로겐램프ON/OFF 제어=
'=======================================
Private Sub BtnDcLampCtrl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnDcLampCtrl.Click
    ' DC 할로겐램프 ON
    If BtnDcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        smartGPIO1.PORTADATA0 = True
        SDDcLamp.SetBackimage = DcLamp_1
    ' DC 할로겐램프 OFF
    ElseIf BtnDcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartGPIO1.PORTADATA0 = False
        SDDcLamp.SetBackimage = DcLamp_0
    End If
End Sub
```

5-3) C++ 예제 전체소스 코드

CPP 예제 소스 코드는 별도로 제공하지 않습니다. SmartX Framework 관련 예제를 참고하시기 바랍니다.

자료위치 안내 : 자사홈페이지(www.hnsts.co.kr) → 자료실 → SmartX 관련자료 → SmartX Framework 예제파일 → SmartX_Example_C++ → SmartGPIOEVC

[표1] Port 핀과 모드, iPortDatas 값 설명 (0번 비트가 1인 경우)

Port 핀	7	6	5	4	3	2	1	0
모드	0	0	0	0	0	0	0	1
iPortDatas 값	0x01(16진수)							

[참고] GPIO관련 자세한 설명은 SmartX Programming Guide의 SmartGPIO편을 참고 바랍니다.

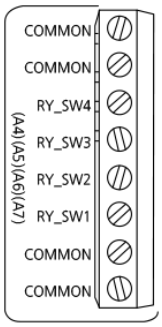
4. Relay 접점 출력단자

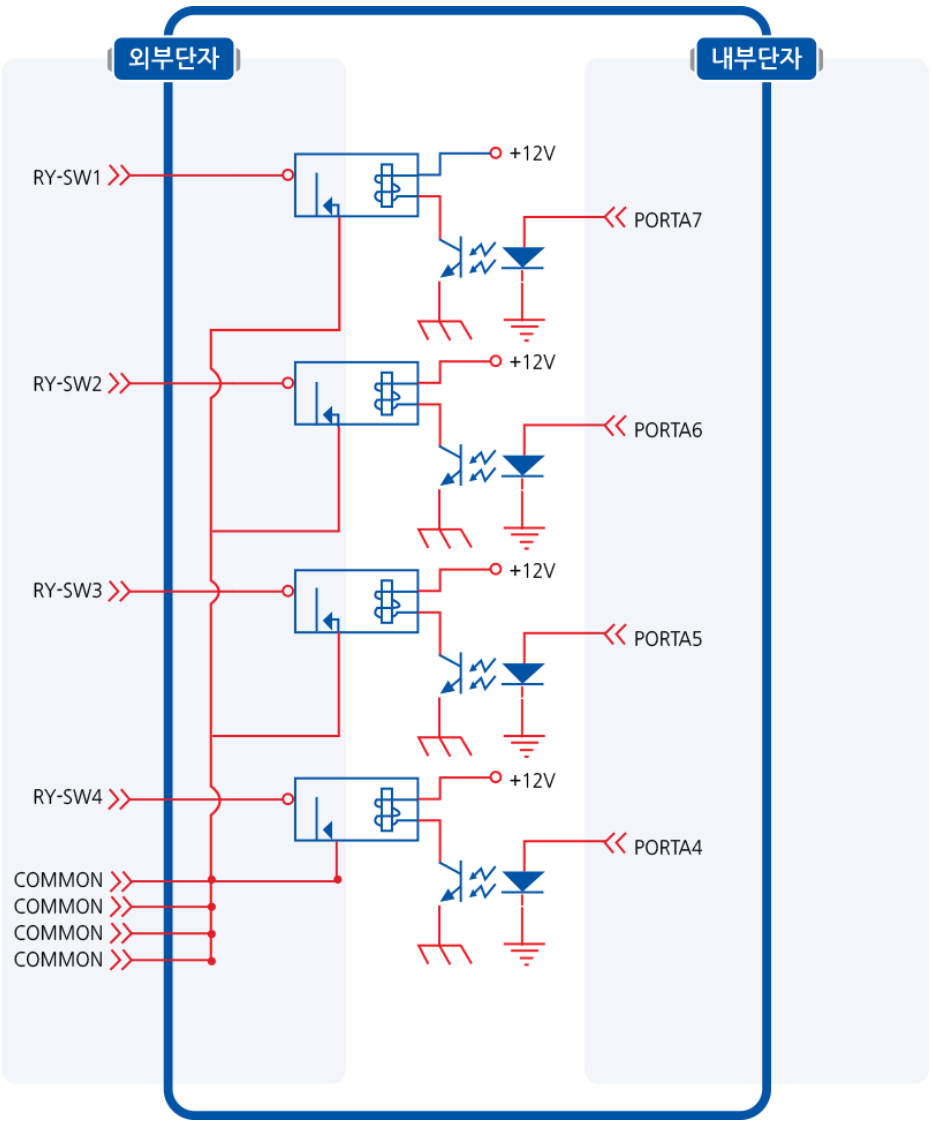
1) Relay 접점 출력단자 소개

RELAY 접점 OUTPUT는 RELAY 접점을 이용하여 간단한 접점제어를 할 수 있습니다. (5A 250VAC/5A 30VDC)

외부입력단자	RY_접점4	RY_ 접점3	RY_ 접점2	RY_ 접점1
내부 Extension Port 연결단자	PORTA4	PORTA5	PORTA6	PORTA7

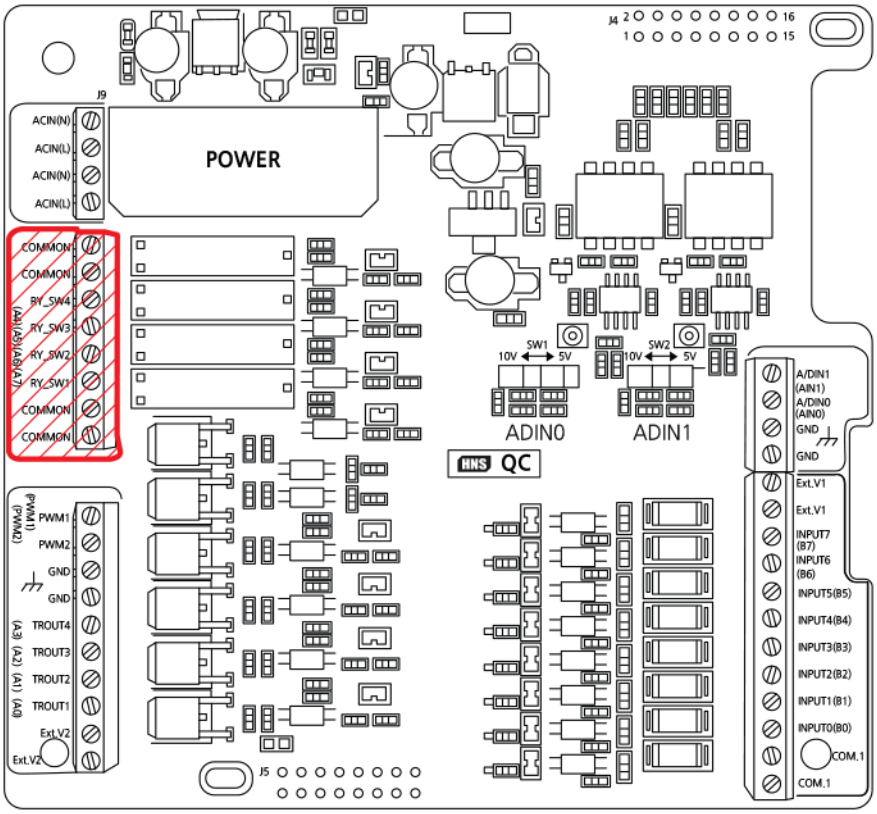
Direction	출력상태(출력)	PortData 값 True/False	LED Status
출력	ON	True	ON
	OFF	False	OFF





[주의] 릴레이 COM단자는 그림과 같이 COMMON단자에 공통으로 묶여 있습니다.

2) Relay 접점 출력단자 위치

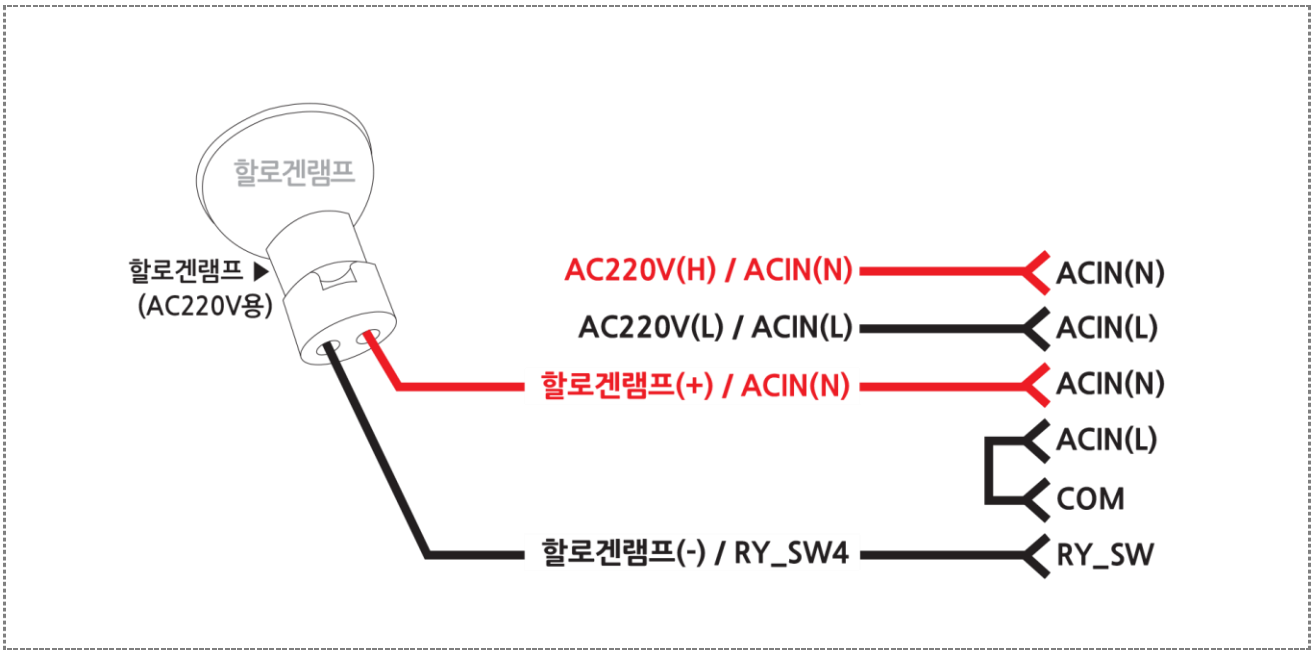


3) Relay 접점 출력단자 응용방법

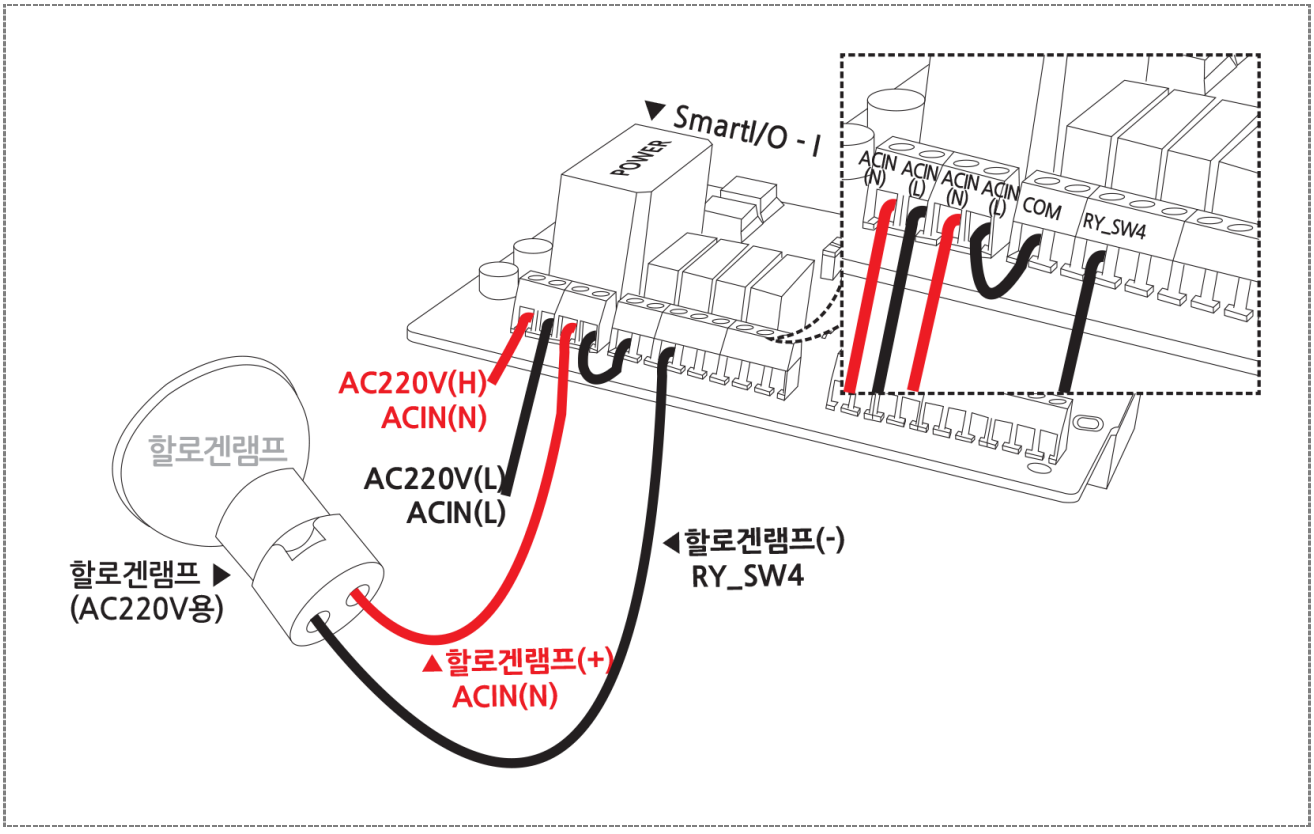
3-1) 할로겐램프 연결방법

아래와 같이 결선을 하시고, Smart I/O_Output의 Relay 접점제어(RY_접점4/RY1-AC할로겐) 항목에서 ON/OFF 테스트 가능

① 결선도



② SmartI/O - I Base Board 결선도 예



[주의]	할로겐램프는 +/- 구분이 없습니다.

4) 언어별 주요소스 코드

C#

```
// Relay S/W AC할로겐램프 ON/OFF 제어
private void BtnAcLampCtrl_Click(object sender, EventArgs e)
{
    // AC 할로겐램프 ON
    if (BtnAcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartGPIO1.PORTADATA4 = true;
        SDAcLamp.SetBackimage = AcLamp_1;
    }
    // AC할로겐램프 OFF
    else if (BtnAcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartGPIO1.PORTADATA4 = false;
        SDAcLamp.SetBackimage = AcLamp_0;
    }
}
```

VB.NET

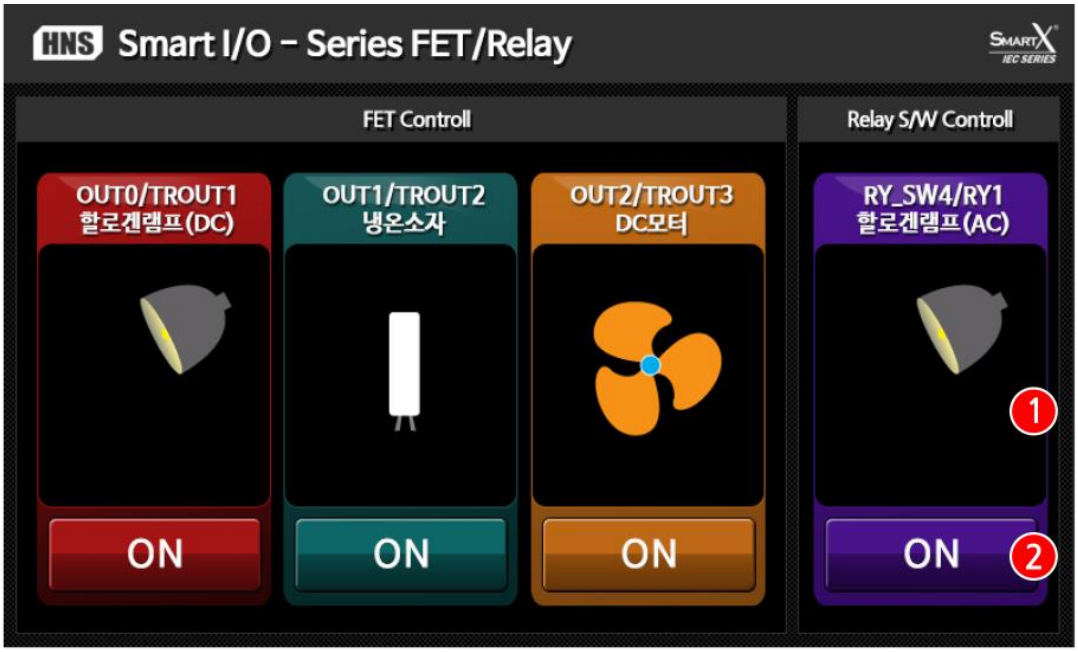
```
'=====================================
'= Relay S/W AC할로겐램프ON/OFF 제어=
'=====================================
Private Sub BtnAcLampCtrl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnAcLampCtrl.Click
    ' AC 할로겐램프ON
    If BtnAcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        smartGPIO1.PORTADATA4 = True
        SDAcLamp.SetBackimage = AcLamp_1
    ' AC할로겐램프OFF
    ElseIf BtnAcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartGPIO1.PORTADATA4 = False
        SDAcLamp.SetBackimage = AcLamp_0
    End If
End Sub
```

C++

```
// PORT CONTROL STATUS의 PORT4 버튼을 클릭 시
void CSmartGPIOEVCDlg::OnBnClickedButport4()
{
    Int iOutData;
    if (nID==IDC_BUTTON4)    // PORT4 버튼이 눌러지면
    {
        iOutData = 1;            // AC 할로겐램프ON
    }else {
        iOutData = 0;           // AC 할로겐램프OFF
    }
    m_SmartGPIO.SetPortABit(4, iOutData);
}
```

5) Application 응용예제

SmartX Framework	SmartGPIO
소스파일	Smart I/O_Output
소스경로	홈페이지 자료실 -> Application Note -> Smart I/O 활용
기 능	AC/DC전원으로 구동하는 장비연동시 온/오프조작
응용분야	AC/DC 스위치기능
준비사항	▪ 할로겐램프 : (AC 220V, 50W)



❶ 이미지 표시 창 ❷ FET ON/OFF

[동작설명]

▶ 프로그램이 시작되면서 ② 각 포트의 ON 버튼을 클릭하면 해당 포트 별로 출력(LOW→HIGH)가 발생하면서 ①의 이미지가 변경 (이미지 변경 그림 추가)됩니다. 4번째 RY_SW4/RY1 할로겐램프(AC)의 경우에는 Relay 기능을 테스트 가능합니다.

5-1) C#예제 전체소스 코드

본 예제의 회로도에서 Output 0은 할로겐램프, Output 1은 냉온소자, Output 2는 DC모터로 연결되어 있습니다.
여기서 설명하는 Output 0은 할로겐램프이며 할로겐램프 이외의 냉온소자, DC모터에 관한 설명은 Application Notes의 Smart I/O_Output 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
// 폼 로드시 실행 됨
private void Form1_Load(object sender, EventArgs e)
{
    smartGPIO1.PORTADIRS = 255;          // PORTA GPIO 방향설정 (ALL OUTPUT)
    smartGPIO1.PORTADATAS = 0;           // PORTA DATA 설정      (ALL LOW)
    …중략…
}
```

[STEP-2] BtnAcLampCtrl 클릭되면 smartGPIO1.PORTADATA4의 상태 값(true/false)을 변경

```
// Relay S/W AC할로겐램프 ON/OFF 제어
private void BtnAcLampCtrl_Click(object sender, EventArgs e)
{
    // AC 할로겐램프 ON
    if (BtnAcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartGPIO1.PORTADATA4 = true;
        SDAcLamp.SetBackimage = AcLamp_1;
    }
    // AC할로겐램프 OFF
    else if (BtnAcLampCtrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartGPIO1.PORTADATA4 = false;
        SDAcLamp.SetBackimage = AcLamp_0;
    }
}
```

5-2) VB.NET 예제 전체소스 코드

본 예제의 회로도에서 Output 0은 할로겐램프, Output 1은 냉온소자, Output 2는 DC모터로 연결되어 있습니다.
여기서 설명하는 Output 0은 할로겐램프이며 할로겐램프 이외의 냉온소자, DC모터에 관한 설명은 Application Notes의 Smart I/O_Output 예제소스를 참고 바랍니다.

[STEP-1] Form1_Load함수에서 SmartGPIO 초기화

```
' 폼로드시 실행 됨
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    smartGPIO1.PORTADIRS = 255      ' PORTA GPIO 방향설정(ALL OUTPUT)
    smartGPIO1.PORTADATAS = 0       ' PORTA DATA 설정      (ALL LOW)
    …중략…
End Sub
```

[STEP-2] BtnAcLampCtrl 클릭되면 smartGPIO1.PORTADATA4의 상태 값(true/false)을 변경

```
'=====================================
'= Relay S/W AC할로겐램프ON/OFF 제어=
'=====================================
Private Sub BtnAcLampCtrl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnAcLampCtrl.Click
    ' AC 할로겐램프 ON
    If BtnAcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        smartGPIO1.PORTADATA4 = True
        SDAcLamp.SetBackimage = AcLamp_1
    ' AC 할로겐램프 OFF
    ElseIf BtnAcLampCtrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartGPIO1.PORTADATA4 = False
        SDAcLamp.SetBackimage = AcLamp_0
    End If
End Sub
```

5-3) C++ 예제 전체소스 코드

CPP 예제 소스 코드는 별도로 제공하지 않습니다. SmartX Framework 관련 예제를 참고하시기 바랍니다.

자료위치 안내 : 자사홈페이지(www.hnsts.co.kr) → 자료실 → SmartX 관련자료 → SmartX Framework 예제파일 → SmartX_Example_C++ → SmartGPIOEVC

[표1] Port 핀과 모드, iPortDatas 값 설명 (0번 비트가 1인 경우)

Port 핀	7	6	5	4	3	2	1	0
모드	0	0	0	0	0	0	0	1
iPortDatas 값	0x01(16진수)							

[참고] GPIO관련 자세한 설명은 SmartX Programming Guide의 SmartGPIO편을 참고 바랍니다.

5. PWM(Pulse Width Modulation) 출력단자

1) PWM(Pulse Width Modulation) 출력단자 소개

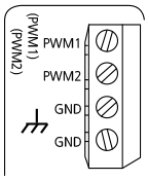
PWM OUTPUT은 FET OUTPUT과 같이 N-Channel FET를 사용하였습니다. 구동하고자 하는 부하에 따라 별도의 전원이 필요합니다. (최대 DC 55V)이므로 부하에 따라 폭넓게 사용할 수 있습니다.

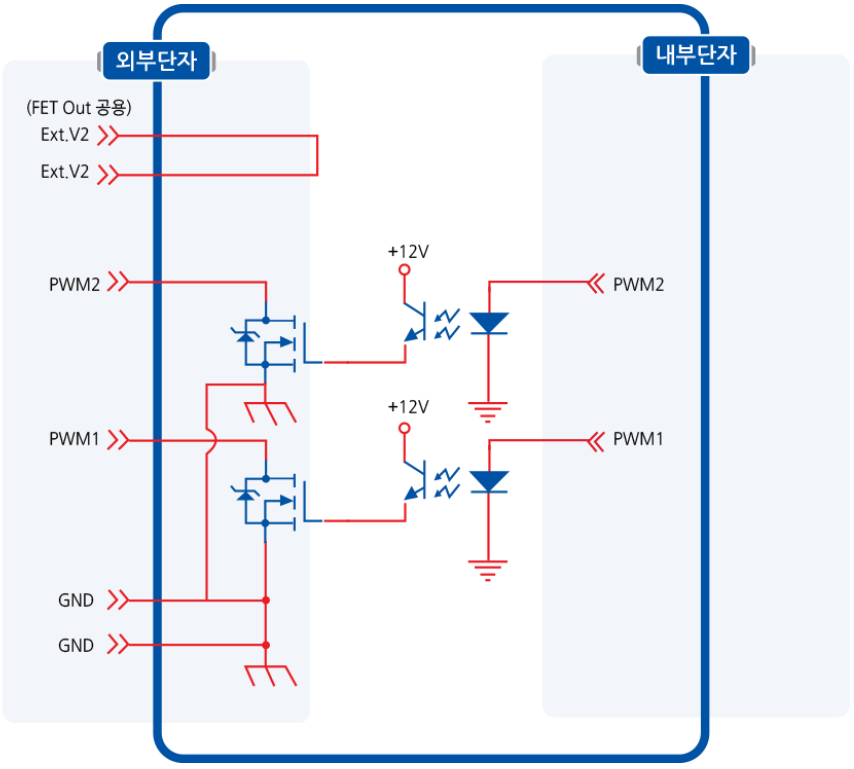
외부입력단자	PWM 1	PWM 2
내부 Extension Port 연결단자	PWM 1	PWM 2

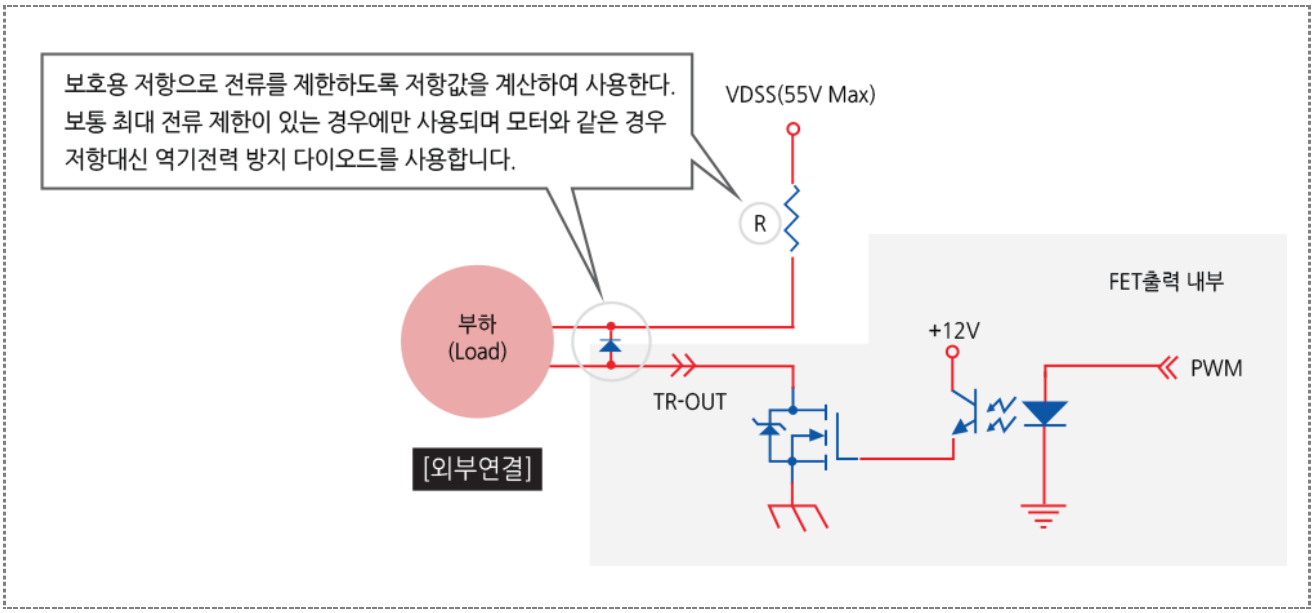
[주의]

[발열에 따른 주의사항]

전력량에 따라 FET 소자에 발열이 발생할 수 있으며, 발열량을 검토하여 전력량을 줄여 발열이 적게 발생하도록 하여 사용하시기 바랍니다.

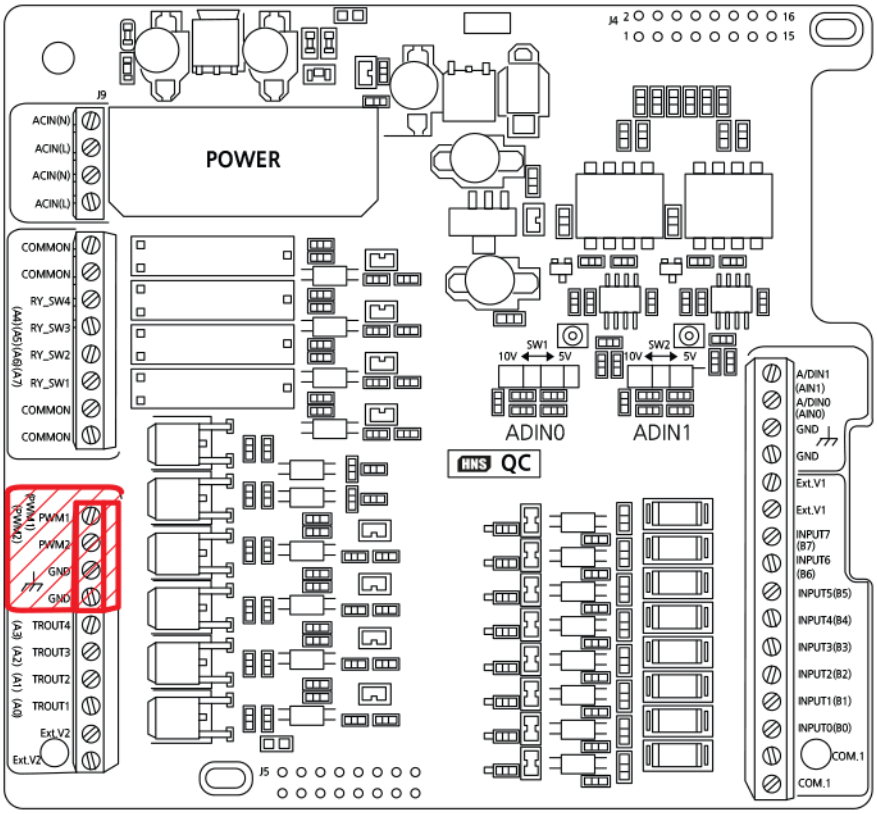






[주의]	IRFR024N은 VDSS = 55V, ID= 17A 정격의 FET입니다.
	Smart I/O – I, II, III 에서는 PWM파형이 반전되어서 출력됩니다.

2) PWM(Pulse Width Modulation) 출력단자 위치

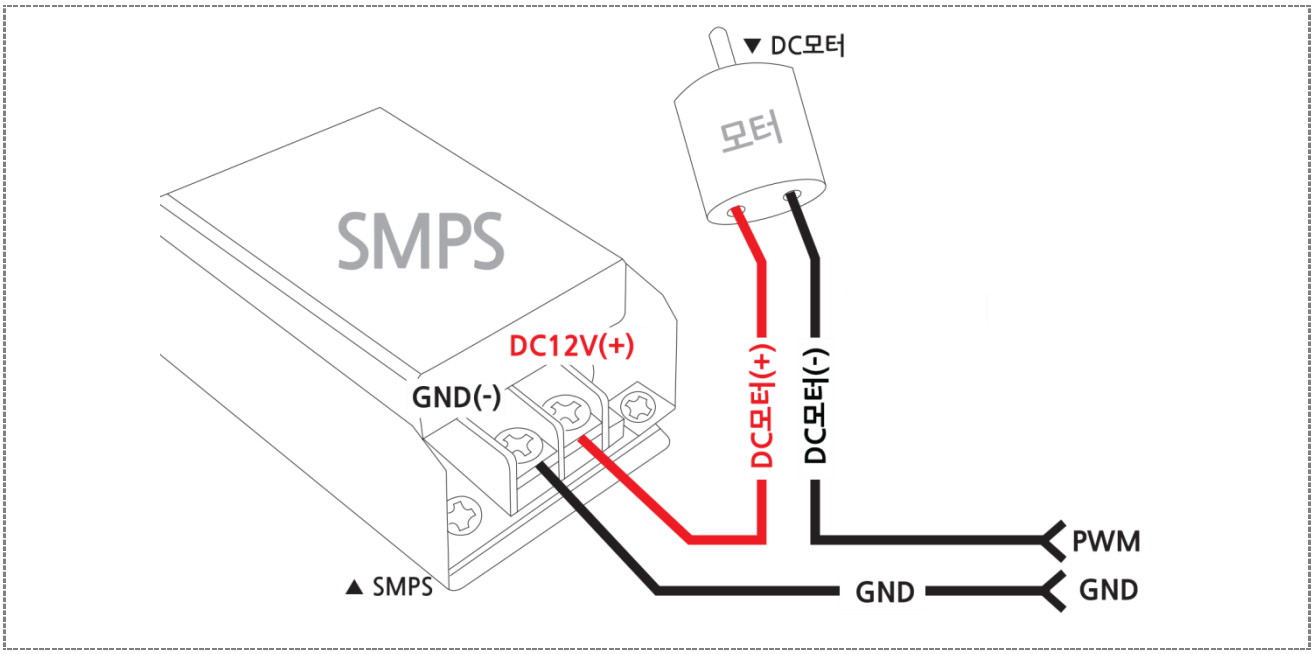


3) PWM(Pulse Width Modulation) 출력단자 응용방법

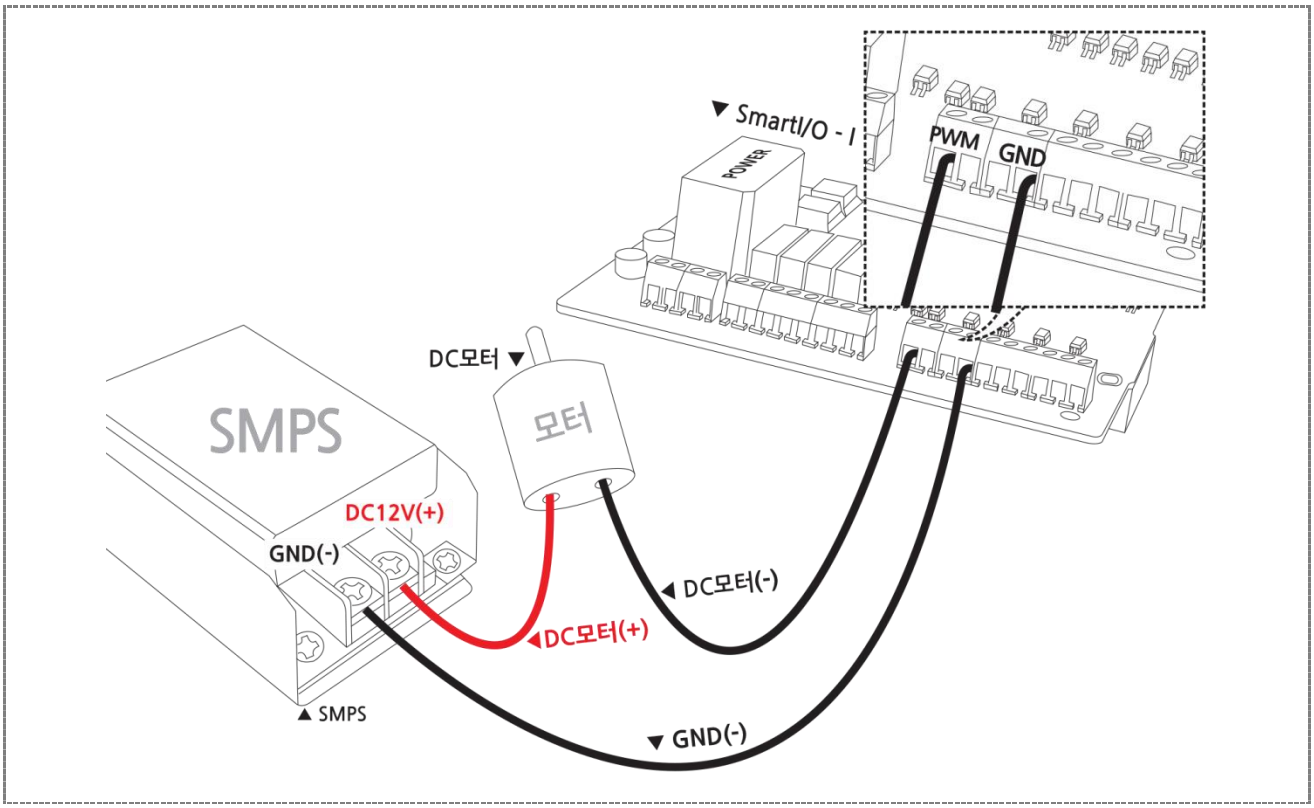
3-1) DC모터 연결방법

아래와 같이 결선을 하시고, Smart I/O_PWM의 DC모터/냉온소자(PWM1)항목에서 Duty Rate를 조절하여 속도테스트 가능하며, PWM의 DutyRate를 이용한 전압조절로 DC모터의 속도를 조절할 수 있다.

① 결선도



② Smart I/O - I Base Board 결선도 예

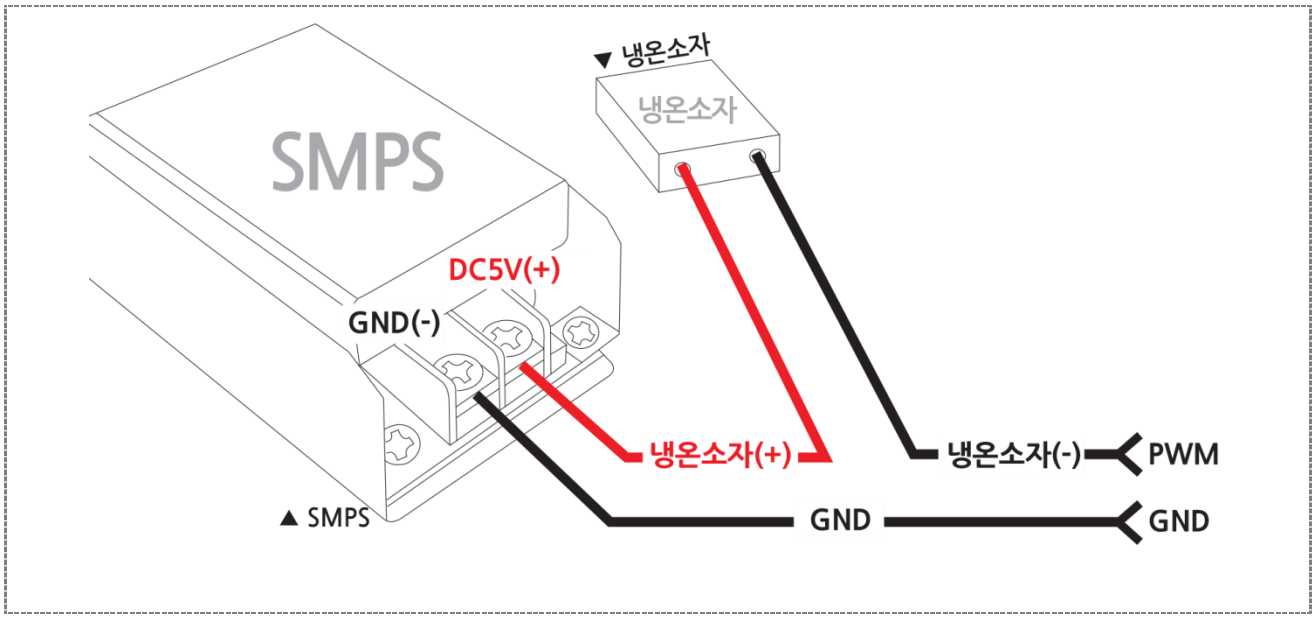


[주의] DC모터는 (+/-)를 반대로 연결할 시에 역방향으로 회전합니다.

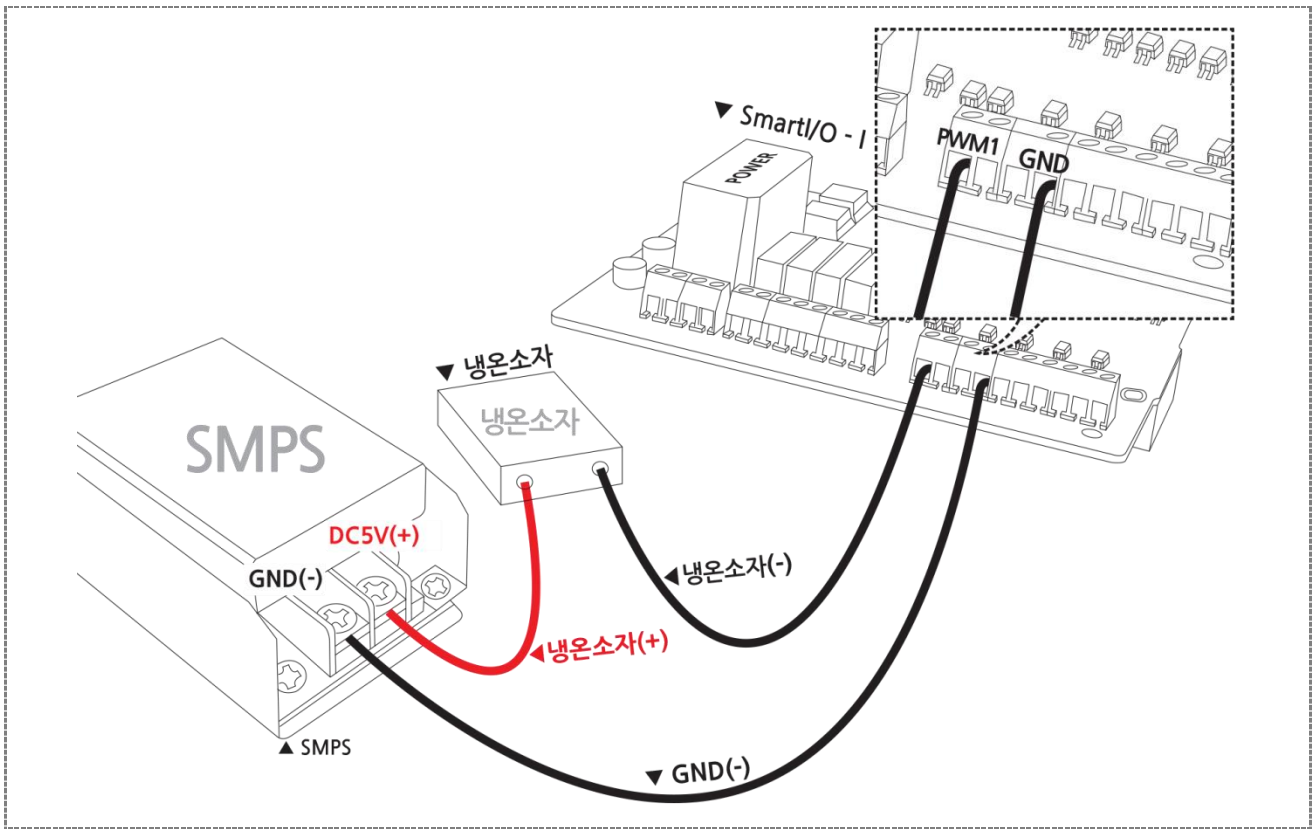
3-2) 냉온소자 연결방법

아래와 같이 결선을 하시고, Smart I/O_PWM의 DC모터/냉온소자(PWM1)항목에서 DutyRate를 조절하여 온도조절테스트 가능하며, PWM의 DutyRate를 이용한 전압조절로 냉온소자의 온도를 조절할 수 있다.

① 결선도



② Smart I/O - I Base Board 결선도 예

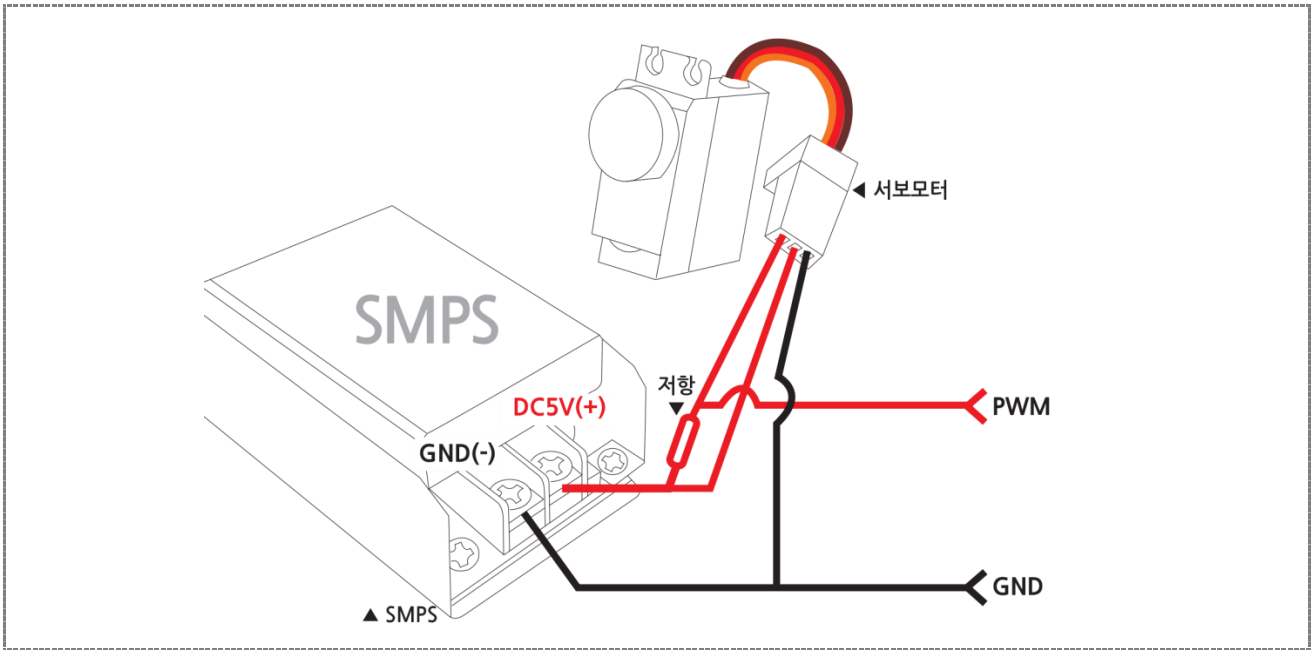


[주의]	전원을 반대로 연결 시 차가운 면과 뜨거운 면이 바뀝니다.
	뜨거운 면의 열을 방열 판으로 식혀줘야 뜨거운 면이 차가운 면으로 열전도 되는 현상을 막을 수 있습니다.
	Ext.V2 단자 2EA는 공핀입니다. 내부로 Ext.V2끼리 서로 묶여있습니다.

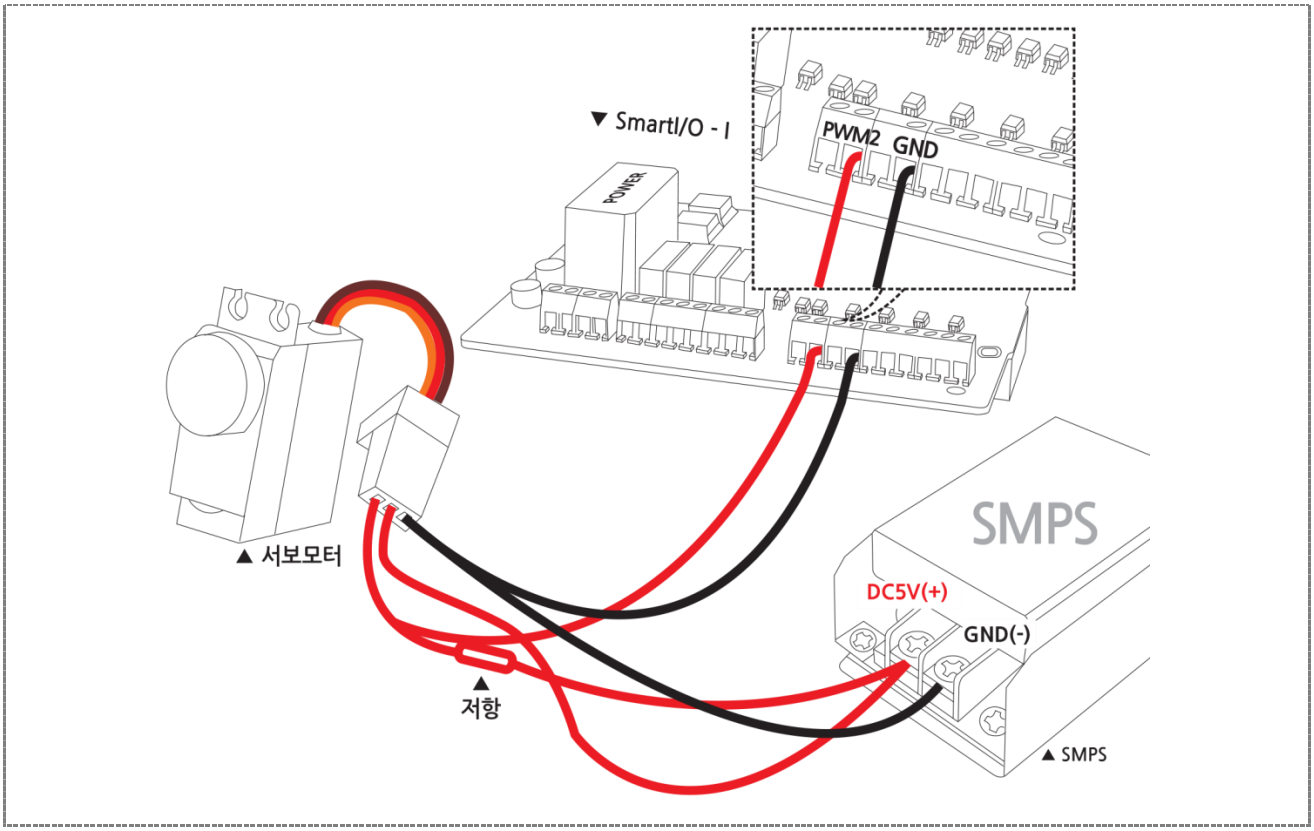
3-3) 서보모터 연결방법

아래와 같이 결선을 하시고, Smart I/O_PWM의 Servo모터(PWM2)항목에서 Servo모터의 각도조절 테스트 가능

① 결선도



② Smart I/O - I Base Board 결선도 예



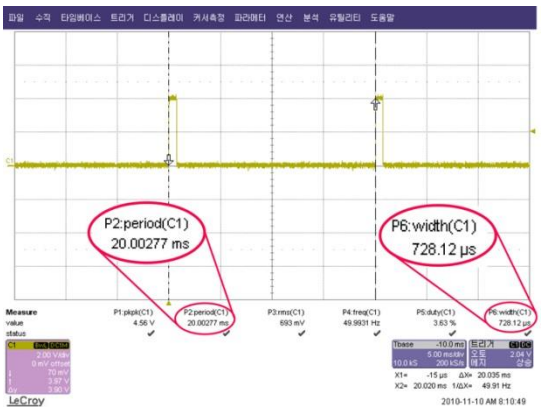
[주의] 저항값은 모터 구동전류에 따라 선택해서 사용합니다. (테스트중에는 4.7KΩ 사용)

※ Smart I/O_PWM 소스상에 Pwm2 부분이 PreScale = 240, PwmCounter = 1381, ClockDivider = 1/4, DutyRate = 96.3 으로 설정되어 있습니다.

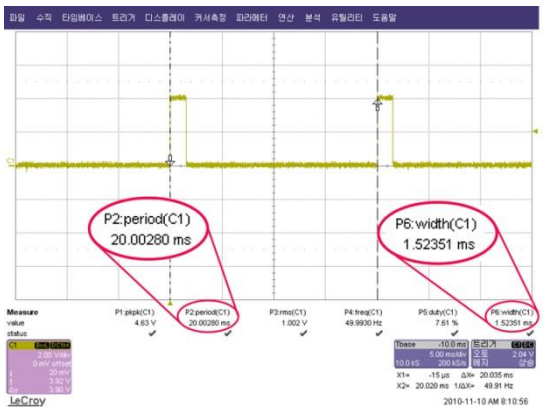
※ ServoMotor는 한주기를 20ms(20.00277ms)로 만들고, 0°(0.7 ms = Duty 96.3), 90°(1.5 ms = Duty 92.3), 180°(2.3 ms = Duty 92.3) 각도 설정버튼(0도, 90도, 180도) 이나 듀티설정으로 각도를 제어할 수 있습니다.

아래는 서보모터 각도별 오실로스코프 화면입니다.

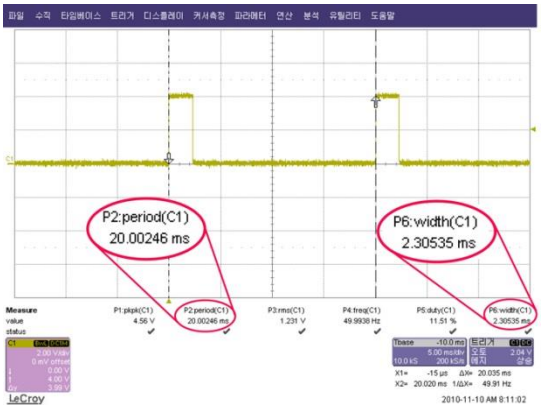
※ 0° (0.7 ms(728.12us) = Duty 96.3)



※ 90° (1.5 ms (1.52351ms) = Duty 92.3)



※ 180° (2.3 ms(2.30535ms)= Duty 92.3)



[주의]	Smart I/O – I, II, III 에서는 PWM파형이 반전되어서 출력됩니다.

4) 언어별 주요소스 코드

C#

```
// PWM1의 DutyRate를 +1 씩 증가
private void BtnPwm1DutyUp_Click(object sender, EventArgs e)
{
    // 1000이상 증가 안 됨
    if (PWM1DutyRate < 100)
    {
        // m_dPWM1DutyRate 값 1씩 증가
        PWM1DutyRate++;
        //레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = PWM1DutyRate.ToString();
    }
    // PWM1의 듀티비 적용
    smartPWM1.DutyRate1 = PWM1DutyRate;
    Pwm1DutyCheck();     // DutyRate1에 따른 처리 코드로 개발자가 직접 작성
}
```

VB.NET

```
' PWM1의DutyRate를+1 씩증가
Private Sub BtnPwm1DutyUp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnPwm1DutyUp.Click
' 1000이상증가안됨
    If PWM1DutyRate < 100 Then
        ' m_dPWM1DutyRate 값1씩 증가
        PWM1DutyRate += 1
        ' 레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = PWM1DutyRate.ToString()
    End If
    ' PWM1의 듀티비 적용
    smartPWM1.DutyRate1 = PWM1DutyRate
    Pwm1DutyCheck()     ' DutyRate1에 따른 처리 코드로 개발자가 직접 작성
End Sub
```

C++

```
// PWM1 Duty값 1%씩 증가
void CSmartPWMEVCDlg::OnDutyInc1()
{
    CString strTemp;
    m_Duty1 = (m_Duty1 < 100) ? ++m_Duty1 : m_Duty1;
    // PWM1의 듀티비 적용
    m_SmartPWM.SetPWMDutyRate1(m_Duty1);
    //레이블에 현재 듀티비 값 표시
    strTemp.Format(L"%d",  m_Duty1);
}
```

5) Application 응용예제

SmartX Framework	SmartPWM
소스파일	Smart I/O_PWM
소스경로	홈페이지 자료실 -> Application Note -> Smart I/O 활용
기 능	DC모터 속도제어, 냉온소자 온도제어, Servo모터 각도제어
응용분야	DutyRate를 이용한 전압제어로 속도 및 강약조절
준비사항	▪ DC모터 : ㈜ 금일모터(KDG37-3429A-050) ▪ 냉온소자 : FALC1-00705T150 ▪ Servo모터 : ㈜ M.I.Tech(MTS-A410SE)

❶ 주파수 설정 표시 ❷ Duty조절(1단위/1~100) ❸ Duty조절(20단위/20~100)
❹ PWM 1 On/Off ❺ Duty / Image 표시 ❻ 파형 반전 / 비 반전 설정
❼ Duty조절(0.1단위/96.3~88.4) ❽ 서보 각도조절(0,90,180도) ❾ PWM 2 On/Off

[참고] 서보 모터 제조사 마다 모델 별 주파수 및 제어 각이 다를 수 있습니다.

[동작설명]

▶ 프로그램이 시작되면서 ④ PWM1 START를 클릭하면 PWM1을 시작합니다. ⑨PWM2 START를 클릭하면 PWM2를 시작합니다.

▶ ①의 Up/Down 버튼을 클릭하면 PWM1의 카운터 값을 조절하여 주파수를 설정할 수 있습니다.

▶ ②의 Down / Up 버튼을 클릭하면 PWM1의 DutyRate를 조절가능 합니다. ⑦의 Down / Up 버튼을 클릭하면 PWM2의 DutyRate를 조절가능 합니다.

▶ ③의 Lv.1 ~ Lv.5버튼을 클릭하면 각 Level에 맞는 DutyRate값과 ⑤에 해당 이미지가 표시됩니다.
예를 들면 Lv.1의 DutyRate는 20, Lv.2의 DutyRate는 40 ...입니다.

▶ ⑨PWM2 START를 클릭하여 PWM2를 시작한 다음 ⑧의 Servo 0', Servo 90', Servo 180'을 클릭하면 해당 각도만큼 서보 모터가 동작합니다.

▶ ⑥을 클릭하여 PWM 출력을 반전 / 비 반전으로 전환 가능합니다.

5-1) C#예제 전체소스 코드

본 예제의 회로도에서 PWM 1은 DC모터 / 냉온소자, PWM 2는 Servo 모터로 연결되어 있습니다.
여기서 설명하는 PWM 1은 DC모터이며 냉온소자, Servo 모터에 관한 설명은 Application Notes의 Smart I/O_PWM 예제 소스를 참고 바랍니다.

[STEP-1] 변수선언 및 초기화

```
public partial class Form1 : Form
{
    // IEC667  설정가능한 주파수 배열
    private string[] m_Frequence = { "1K", "2K", "5K", "10K", "20K", "50K", "100K", "200K", "500K", "1M", "2M", "8M",
    "11M", "16M"};

    // PWM1 & 2의 프리스케일러의 값을 처리하는 변수선언 및 초기화
    private byte Prescaler = 240;
    // PWM1 & 2의 DeadTime 값을 처리하는 변수선언 및 초기화
    private byte DeadTime = 0;

    // PWM1의 듀티비, PWMCounter, ClockDiviert, Polarity 설정 변수선언 및 초기화
    private double PWM1DutyRate = 0;
    private uint PWM1Counter = 200;
    private SmartX.SmartPWM.CLOCKDIVIDER PWM1ClockDivide = SmartX.SmartPWM.CLOCKDIVIDER.DIVIDE2;
    private SmartX.SmartPWM.POLARITY PWM1Polarity = SmartX.SmartPWM.POLARITY.HIGHACTIVE;
    …중략…
}
```

[STEP-2] Form1_Load함수에서 SmartPWM 초기값 설정

```
// 폼 로드시 실행 됨
private void Form1_Load(object sender, EventArgs e)
{
    // PWM1, PWM2 초기 프리스케일러 설정
    smartPWM1.PreScaler = Prescaler;

    // PWM1, PWM2 초기 디바이더 설정
    smartPWM1.ClockDivider1 = PWM1ClockDivide;
    smartPWM1.ClockDivider2 = PWM2ClockDivide;

    // PWM출력 비반전 처리
    smartPWM1.Polarity1 = PWM1Polarity;
    smartPWM1.Polarity2 = PWM2Polarity;

    // PWM1, PWM2 Counter 초기 값 설정
    smartPWM1.PWMCounter1 = PWM1Counter;
    smartPWM1.PWMCounter2 = PWM2Counter;

    // PWM1, PWM2 듀디비 초기 값 표시 및 설정
    LblPwm1Duty.Text = PWM1DutyRate.ToString();
    smartPWM1.DutyRate1 = PWM1DutyRate;
    smartPWM1.DutyRate2 = PWM2DutyRate;

    // PWM1 & PWM2 DeadTime 초기 값 표시 및 설정
    smartPWM1.DeadTime = DeadTime;
    …중략…
}
```

[STEP-3] PWM1의 PWM출력을 시작 / 정지

```
private void BtnPwm1Ctrl_Click(object sender, EventArgs e)
{
    // PWM1 Start
    if (BtnPwm1Ctrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartPWM1.StartPWM1();
    }
    // PWM1 Stop
    else if (BtnPwm1Ctrl.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartPWM1.StopPWM1();
        // Pwm1 DutyRate Value 0
        PWM1DutyRate = 0;
        // PWM1의 듀티비 적용
        smartPWM1.DutyRate1 = PWM1DutyRate;
        //레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = "0";
        Pwm1DutyCheck();
        BtnPwm1Lv1.ButtonUp();
        BtnPwm1Lv2.ButtonUp();
        BtnPwm1Lv3.ButtonUp();
        BtnPwm1Lv4.ButtonUp();
        BtnPwm1Lv5.ButtonUp();
    }
}
```

[STEP-4] PWM의 DutyRate 조정(증가 경우)

```
// PWM1의 DutyRate를 +1 씩 증가
private void BtnPwm1DutyUp_Click(object sender, EventArgs e)
{
    // 1000이상 증가 안됨
    if (PWM1DutyRate < 100)
    {
        // m_dPWM1DutyRate 값 1씩 증가
        PWM1DutyRate++;
        //레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = PWM1DutyRate.ToString();
    }
    // PWM1의 듀티비 적용
    smartPWM1.DutyRate1 = PWM1DutyRate;
    Pwm1DutyCheck();     // DutyRate1에 따른 처리 코드로 개발자가 직접 작성
}
```

[STEP-5] PWM출력 반전 / 비 반전 설정버튼

```
private void BtnPolarity_Click(object sender, EventArgs e)
{
    if (BtnPolarity.ButtonStatus == SmartX.SmartButton.BUTSTATUS.DOWN)
    {
        smartPWM1.Polarity2 = SmartX.SmartPWM.POLARITY.HIGHACTIVE;          // PWM출력 비반전처리
    }
    else if (BtnPolarity.ButtonStatus == SmartX.SmartButton.BUTSTATUS.UP)
    {
        smartPWM1.Polarity2 = SmartX.SmartPWM.POLARITY.LOWACTIVE;          // PWM출력 반전처리
    }
}
```

5-2) VB.NET 예제 전체소스 코드

본 예제의 회로도에서 PWM 1은 DC모터 / 냉온소자, PWM 2는 Servo 모터로 연결되어 있습니다.
여기서 설명하는 PWM 1은 DC모터이며 냉온소자, Servo 모터에 관한 설명은 Application Notes의 Smart I/O_PWM 예제 소스를 참고 바랍니다.

[STEP-1] 변수선언 및 초기화

```
Public Class Form1
    ' IEC667 설정 가능한 주파수 배열
    Dim m_Frequence() As String = {"1K", "2K", "5K", "10K", "20K", "50K", "100K", "200K", "500K", "1M", "2M", "8M",
    "11M", "16M"}
    ' PWM1 & 2의 프리스케일러의 값을 처리하는 변수선언 및 초기화
    Dim Prescaler As Byte = 240
    ' PWM1 & 2의 DeadTime 값을 처리하는 변수선언  및 초기화
    Dim DeadTime As Byte = 0
    ' PWM1의듀티비, PWMCounter, ClockDiviert, Polarity 설정 변수선언 및 초기화
    Dim PWM1DutyRate As Double = 0
    Dim PWM1Counter As Integer = 200
    Dim PWM1ClockDivide As SmartX.SmartPWM.CLOCKDIVIDER = SmartX.SmartPWM.CLOCKDIVIDER.DIVIDE2
    Dim PWM1Polarity As SmartX.SmartPWM.POLARITY = SmartX.SmartPWM.POLARITY.HIGHACTIVE
    …중략…
End Class
```

[STEP-2] Form1_Load함수에서 SmartPWM 초기값 설정

```
' 폼로드시 실행 됨
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    ' PWM1, PWM2 초기 프리스케일러 설정
    smartPWM1.PreScaler = Prescaler

    ' PWM1, PWM2 초기 디바이더 설정
    smartPWM1.ClockDivider1 = PWM1ClockDivide
    smartPWM1.ClockDivider2 = PWM2ClockDivide

    ' PWM출력 비반전처리
    smartPWM1.Polarity1 = PWM1Polarity
    smartPWM1.Polarity2 = PWM2Polarity

    ' PWM1, PWM2 Counter 초기 값 설정
    smartPWM1.PWMCounter1 = PWM1Counter
    smartPWM1.PWMCounter2 = PWM2Counter

    ' PWM1, PWM2 듀디비초기 값 표시 및 설정
    LblPwm1Duty.Text = PWM1DutyRate.ToString()
    smartPWM1.DutyRate1 = PWM1DutyRate
    smartPWM1.DutyRate2 = PWM2DutyRate

    ' PWM1 & PWM2 DeadTime 초기 값 표시 및 설정
    smartPWM1.DeadTime = DeadTim
    …중략…
End Sub
```

[STEP-3] PWM1의 PWM출력을 시작 / 정지

```
' PWM1의PWM 출력을 시작/정지한다.
Private Sub BtnPwm1Ctrl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnPwm1Ctrl.Click
    ' PWM1 Start
    If BtnPwm1Ctrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
    smartPWM1.StartPWM1()
    ' PWM1 Stop
    ElseIf BtnPwm1Ctrl.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartPWM1.StopPWM1()
        ' Pwm1 DutyRate Value 0
        PWM1DutyRate = 0
        ' PWM1의 듀티비 적용
        smartPWM1.DutyRate1 = PWM1DutyRate
        '레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = "0"
        Pwm1DutyCheck()

        BtnPwm1Lv1.ButtonUp()
        BtnPwm1Lv2.ButtonUp()
        BtnPwm1Lv3.ButtonUp()
        BtnPwm1Lv4.ButtonUp()
        BtnPwm1Lv5.ButtonUp()
    End If
End Sub
```

[STEP-4] PWM의 DutyRate 조정(증가 경우)

```
' PWM1의 DutyRate를 +1 씩 증가
Private Sub BtnPwm1DutyUp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnPwm1DutyUp.Click
' 100이상 증가 안 됨
    If PWM1DutyRate < 100 Then
        ' m_dPWM1DutyRate 값1씩 증가
    PWM1DutyRate += 1
        ' 레이블에 현재 듀티비 값 표시
        LblPwm1Duty.Text = PWM1DutyRate.ToString()
    End If
    ' PWM1의 듀티비적용
    smartPWM1.DutyRate1 = PWM1DutyRate
    Pwm1DutyCheck()     ' DutyRate1에 따른 처리 코드로 개발자가 직접 작성
End Sub
```

[STEP-5] PWM출력 반전 / 비 반전 설정버튼

```
' PWM 출력 반전/비반전 설정 버튼
Private Sub BtnPolarity_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnPolarity.Click
    If BtnPolarity.ButtonStatus = SmartX.SmartButton.BUTSTATUS.DOWN Then
        smartPWM1.Polarity2 = SmartX.SmartPWM.POLARITY.HIGHACTIVE                    ' PWM출력비반전처리
    ElseIf BtnPolarity.ButtonStatus = SmartX.SmartButton.BUTSTATUS.UP Then
        smartPWM1.Polarity2 = SmartX.SmartPWM.POLARITY.LOWACTIVE                     ' PWM출력반전처리
    End If
End Sub
```

5-3) C++ 예제 전체소스 코드

CPP 예제 소스 코드는 별도로 제공하지 않습니다. SmartX Framework 관련 예제를 참고하시기 바랍니다.

자료위치 안내 : 자사홈페이지(www.hnsts.co.kr) → 자료실 → SmartX 관련자료 → SmartX Framework 예제파일 → SmartX_Example_C++ → SmartPWMEVC

[주의] Pwm1의 캐리어 주파수를 지정해서 사용하실 경우에는 PreScaler는 Pwm1/2 공통으로 적용되는 부분이라 Pwm 1/2를 동시에 사용 불가합니다.

[참고]

Duty Up / Down버튼과 Lv1~Lv5버튼을 통해서 PWM1의 Dutyrate를 조절하여 DC모터 속도 제어/냉온소자 온도조절 테스트가능

Pwm2의 서보모터 제어부분도 위와 동일한 방식으로 설정하고 Servo모터 연결방법부분에 설명기재.
Pwm1의 캐리지주파수는 "1K", "2K", "5K", "10K", "20K", "50K","100K", "200K", "500K", "1M", "2M", "8M", "11M", "16M" 별로 설정해서 사용 가능합니다. (IEC266은 8Mhz까지만 지원됨)

PWM관련 자세한 설명은 SmartX Programming Guide의 SmartPWM편을 참고 바랍니다.

[MEMO]

Smart I/O - I

제품 매뉴얼

제품 및 자세한 문의사항은 아래의 연락처로 연락 및 메일 문의 주시기 바랍니다. 감사합니다.

㈜에이치앤에스

서울특별시 금천구 가산디지털1로 181, 1505호(가산 W CENTER)

대표전화 : 02-6402-8001 / 팩스 : 02-6442-9775

부서안내	연락처	직통 전화	이메일
제품 구매 및 견적문의	02-6402-8001(내선 1번)	070-7094-5770	sales@hnsts.co.kr
하드웨어 기술문의	02-6402-8001(내선 2번)	070-7094-5001	hns@hnsts.co.kr
소프트웨어 기술문의	02-6402-8001(내선 3번)	070-7094-5002	app@smartx.co.kr
제품서비스 기술문의	02-6402-8001(내선 4번)	070-7094-5003	tech@smartx.co.kr

홈페이지 : www.hnsts.co.kr / 쇼핑몰(제품구매) : www.hnsstore.co.kr