왕초보를 위한

MSB 와 HMI 사용설명서

구입하신 제품에 대한 상세설명서는 www.comfile.co.kr 해당 제품 상세페이지에 있습니다. 본 책은 입문자를 위한 해설서입니다.

COMFILE TECHNOLOGY

컴파일 테크놀로지 주식회사

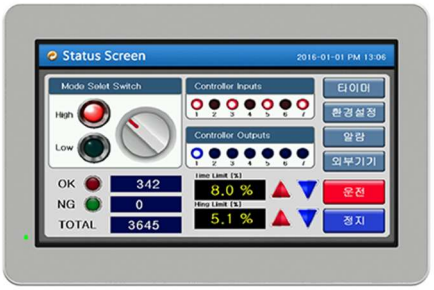
www.comfile.co.kr

머릿말

본 책은 저희 회사 MSB제품과 HMI를 처음 접하시는 사용자 분들을 위해, 복잡한 내용은 빼고 꼭 필요한 부분만 엄선해서 간략하게 만든 초보자용 입문 서적입니다. 다음 두 제품을 가지고 설명을 진행합니다.

MSB610L-DC

산업용 콘트롤러 CUBLOC 일체형 제품입니다. 6개의 입력과 4개의 출력포트를 내장하고 있습니다.



CHA-070WT

유저 인터페이스를 담당하는 HMI입니다. 7인치 칼라 디스플레이를 갖춘 제품입니다.

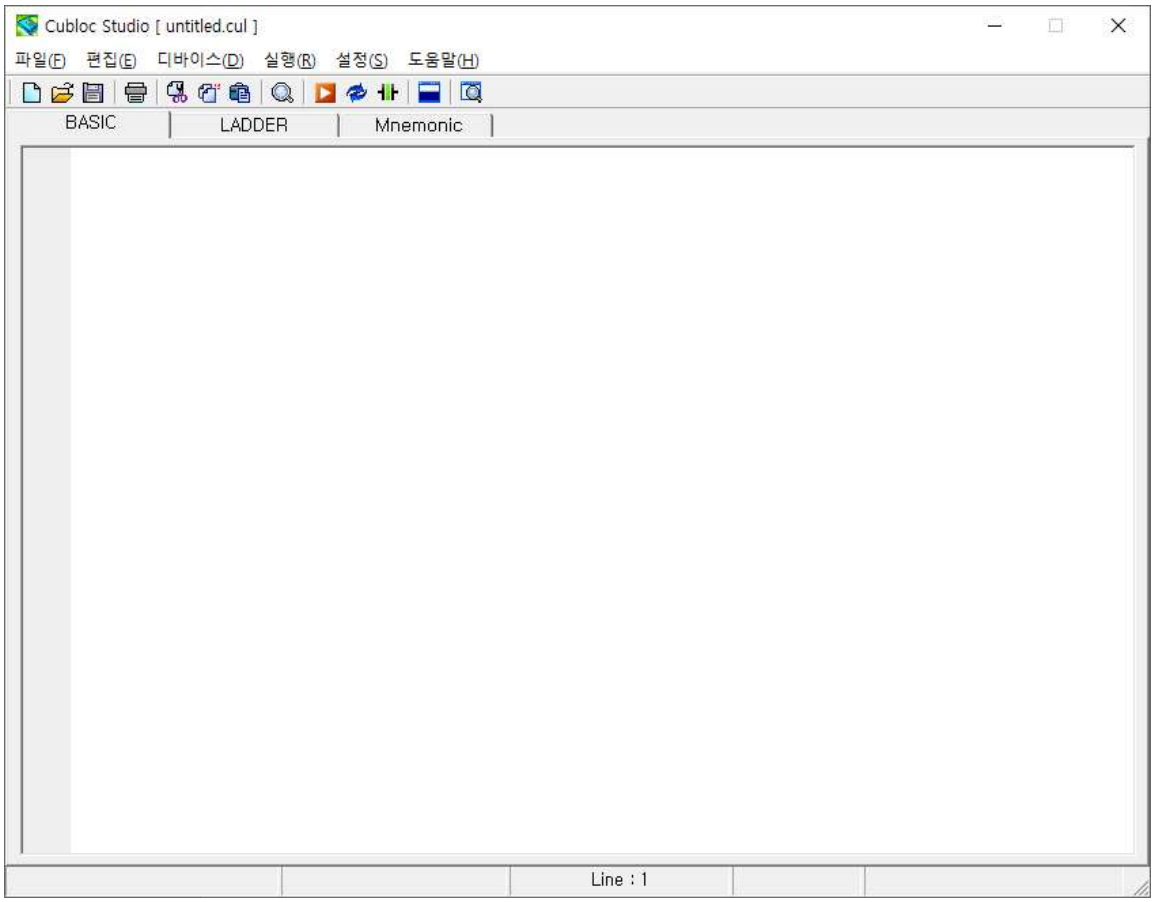
보다 구체적인 내용은 아래 설명서를 참조하세요.

https://www.comfile.co.kr/shop/board/view.php?id=guide&no=4

목차

CUBLOC STUDIO

MSB를 사용하기 위해서는 CUBLOC STUDIO가 필요합니다. www.comfile.co.kr 의 자료실에서 다운로드 받은 후 설치하세요. 실행하면 다음과 같은 화면이 뜹니다.



이 프로그램을 메모장 프로그램이라고 생각하시고, 가운데 빈 공간에 소스코드를 입력하시면 됩니다.

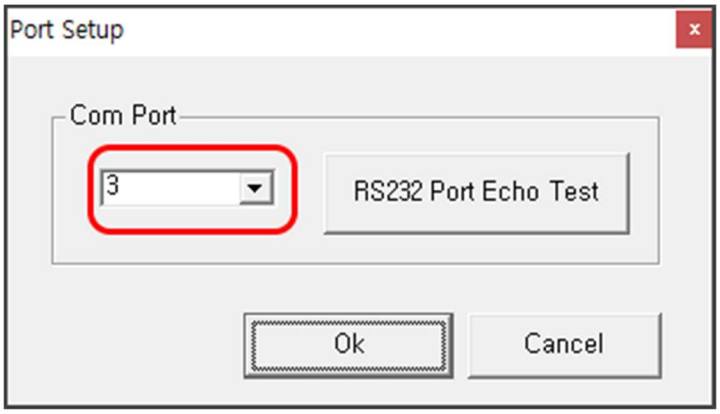
MSB 다운로드 방법

이렇게 생긴 3-PIN 다운로드 케이블이 필요합니다. 이 케이블을 PC의 RS232C 포트에 연결하세요. 만약 PC에 RS232C포트가 없다면 USB-TO-RS232C 컨버터를 통해서 연결하십시오.

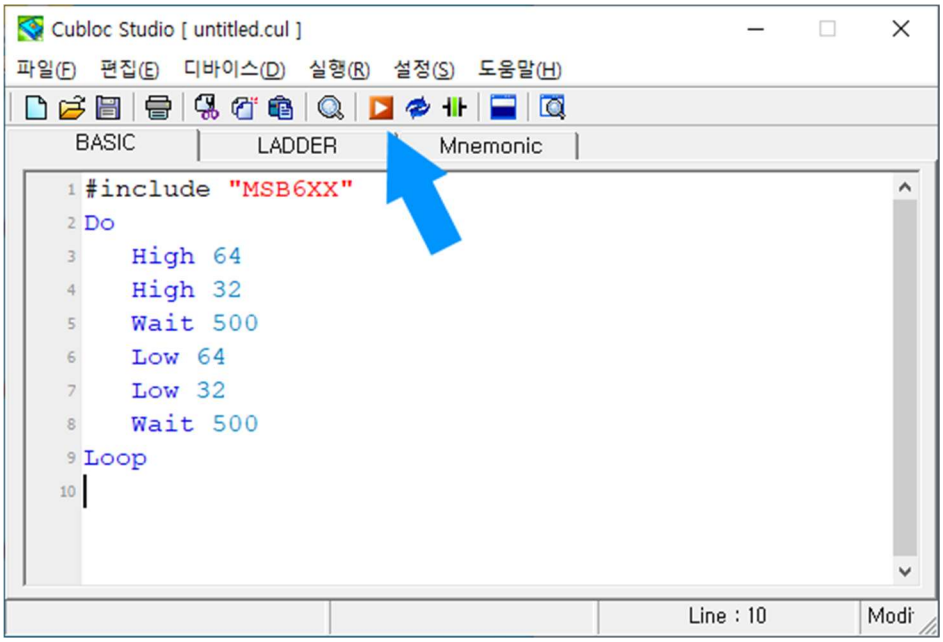
윈도우7이상에서는 드라이버가 자동으로 설치됩니다. 드라이버가 제대로 설치되었다면, 장치관리자에서 COM포트를 확인하실 수 있습니다.

USB드라이버가 자동으로 설치되지 않는다구요? www.comfile.co.kr 자료실에 있는 USB-TO-RS232C 드라이버를 다운받아서 직접 설치해보시기 바랍니다.

설정메뉴의 <PC인터페이스설정>을 누르면 아래와 같은 창이 뜹니다. 여기에 연결된 COM포트를 선택하시면 됩니다.



자 그러면 이제 소스 코드를 입력해 넣고, 빨간색 버튼을 누르면 됩니다.



MSB에 전원을 연결하고 위 소스를 다운로드 해보세요. 0.5초간격으로 Status LED 와 릴레이가 점멸한다면 성공한 것입니다.

BASIC 언어

BASIC 언어

BASIC 언어는 초보자가 쉽게 배울 수 있도록 설계된 언어입니다. 문법자체가 어렵지 않으므로 가볍게 따라오시면 됩니다. 우선 간단한 BASIC 소스를 하나 보겠습니다.

```
#include "MSB6XX"
Do
        High 32
        Wait 500
        Low 32
        Wait 500
Loop
```

가장 첫 줄에는 디바이스 선언문을 적어주면 됩니다. MSB6XX 시리즈는 #include "MSB6XX"를 사용합니다.

이 소스코드는 포트32를 0.5초 간격으로 On/OFF해주는 소스입니다. 그러기 위해서는 먼저 포트32를 On 해주어야 겠지요? High 32가 포트32를 On해주는 명령어입니다.

Wait 500 은 0.5초를 기다려주는 명령어 입니다. 그 뒤 Low 명령으로 포트32를 꺼주면 됩니다.

Do, Loop 는 루프 명령어 입니다. Do 뒤에 특별한 조건이 없기 때문에 루프 사이에 있는 명령어들을 무한 반복 합니다.

가장 기본적인 BASIC 문법

초보자는 아래 5가지만 알아도 BASIC소스코드를 작성할 수 있습니다.

- Dim 변수를 선언합니다.
- Do 와 Loop 반복루프를 만듭니다.
- For 와 Next 지정된 횟수만큼 반복 실행하도록 합니다.
- If 비교 명령어 입니다.
- Goto 와 Gosub 분기명령입니다.

변수선언

CUBLOC BASIC 에서의 변수형은 모두 5 가지가 있습니다.

- BYTE 8 비트 부호 없는 정수, 0~255
- INTEGER 16 비트 부호 없는 정수, 0~65535
- LONG 32 비트 부호 있는 정수, -2147483648~+2147483647
- SINGLE 32 비트 실수, -3.402823E+38~3.402823E+38
- STRING 문자열, 0 TO 127 byte

Dim 명령을 사용해서 변수를 선언합니다.

```
Dim [변수명] As 변수형
```

사용예)

```
Dim Motor1 As Integer
```

여기서 Motor1은 변수명입니다. 변수명은 유저가 직접 정하는 이름입니다. 이 변수에 값을 저장할 수 있습니다. 변수형마다 저장할 수 있는 값의 최대치가 다르므로 처음 변수를 정할 때 적당한 변수형을 고르면 됩니다. 변수명은 변수의 성격을 잘 대변할 수 있는 적절한 이름으로 작명해야 소스를 작성하는 내내 편합니다.

BASIC 흐름 제어 명령

BASIC은 소스코드의 맨 첫 줄부터 아래로 실행됩니다. 그 흐름을 바꿀 수 있는 명령어가 흐름제어 명령어입니다.

Do While 와 Loop

[조건]이 참이라면 안쪽에 있는 명령문들을 계속 실행합니다.

```
Do While [조건]
    명령문
Loop
```

또는

```
Do
    명령문
Loop While [조건]
```

이렇게 사용할 수 있습니다.

[조건]에는 비교연산자를 사용해서 비교연산식을 써주면 됩니다.

While 없이 사용한다면 무한 루프가 됩니다.

```
Do
    명령문
Loop
```

IF

비교 연산식의 참, 거짓 여부에 따라 어떤 명령문을 수행할 것인지가 결정됩니다. 다양한 유형이 있습니다.

유형1

```
If  A<10  Then  B=1
```

유형2

```
If  A<10  Then  B=1  Else  C=1
```

유형3

```
If  A<10  Then   '*복문구조 if문일 경우에는 반드시 Then 뒤에 명령어가 없어야 합니다.
  B=1
End If
```

유형4

```
If  A<10  Then
        B=1
Else
        C=1
End If
```

유형5

```
If  A<10  Then
        B=1
Elseif A<20 Then
        C=1
End If
```

ELSEIF에서 ELSE와 IF는 꼭 붙여서 작성하세요

유형6

```
If  A<10  Then
        B=1
Elseif A<20 Then
        C=1
Elseif A<40 Then
        C=2
Else
        D=1
End If
```

FOR와 NEXT

FOR..NEXT문은 지정된 횟수만큼 명령문을 반복 실행하는 명령어입니다.

```
For 카운터=시작 값 To 끝 값 [Step 증가분]
        명령문
Next
```

다음 예와 같이 STEP명령이 생략된 경우에는 1씩 증가합니다.

```
Dim K As Long
For K=0 To 10
        Debug Dp(K),CR
Next
```

FOR문의 카운터 변수는 카운터 할 값의 범위를 충분히 커버할 수 있는 변수 형으로 선언해야 합니다. 예를 들어 255까지 카운트하는 경우에도 256까지 카운트할 수 있는 INTEGER형을 사용해야 합니다.

```
Dim K As Byte
For K=0 To 255
        Debug Dp(K),CR   'K를 Byte형으로 했을 경우 비교가 불가능하므로 무한루프가 됩니다.
Next
```

Goto

소스코드의 특정라벨로 점프합니다.

```
EndlessLoop:
        명령문
        Goto EndlessLoop
```

Goto를 사용하기 위해서는 라벨을 먼저 선언해야 합니다. 라벨은 여러분이 작명하신 이름 뒤에 콜론(:)을 사용해서 선언하시면 됩니다.

```
Proc_Main:
```

Gosub

서브루틴으로 점프했다가, Return명령어를 만나면 되돌아 옵니다.

```
        명령문1
        Gosub Proc_Sub
        명령문2

Proc_Sub:
        명령문3
        Return
```

명령문1을 실행하다가 Gosub를 만나면, Proc_Sub라벨 위치로 점프합니다. 여기서 명령문3을 수행하고 Return을 만나면 돌아와서 명령문2를 수행합니다.

Goto 와 Gosub는 초보자단계에서만 쓰시고, 나중에는 Sub, Function을 써서 "부 프로그램"을 만드는 방법을 쓰시는 것이 좋습니다.

연산식과 연산자

BASIC에서는 다양한 연산식이 쓰이는데 이때 필요한 것이 바로 연산자입니다.

- 산술연산자 : 더하기 빼기와 같은 수학연산을 하는데 사용합니다.(+,-,*등)
- 비교연산자: IF, DO..WHILE 문 등에서 비교연산을 하는데 사용합니다. (<,>,=등)
- 논리연산자: 논리연산을 하는데 사용합니다. (AND, OR 등)

연산자	설명	종류	우선순위
^	거듭제곱	산술연산	높음
*, /, MOD	곱하기, 나누기, 나머지	산술연산	
+,-	더하기, 빼기	산술연산	
<<, >>	좌쉬프트, 우쉬프트	비교연산	
<, >, <=, >=	작다, 크다, 같거나작다, 같거나크다	비교연산	
=, <>	같다, 다르다	비교연산	
AND, XOR, OR	AND, XOR, OR연산	논리연산	낮음

연산기호

BASIC에서는 수학에서 사용하는 연산기호와 다른 연산기호를 사용합니다.

Operator	수학식	Basic언어	BASIC에서 사용 예
덧셈	+	+	3+4+5, 6+A
뺄셈	-	-	10-3, 63-B
곱셈	X	*	2 * 4, A * 5
나누기	÷	/	1234/3, 3843/A
제곱	5^3	^	5^3, A^2
나머지연산	없음	mod	102 mod 3

BASIC 문법에 대해서는 이 정도만 알아도 충분히 소스코드를 짤 수 있습니다. 나중에 추가적으로 필요한 부분을 그때 그때 아래 책을 보시면서 공부하시면 됩니다.

https://www.comfile.co.kr/download/cubloc/cubloc_basic_manual.pdf

CUBLOC BASIC언어 중심 사용설명서

입출력 라이브러리

라이브러리에는 "명령문"형식과 "함수"형식이 있습니다.

명령문	INPUT, OUTPUT
함수	IN( ), ADIN( )

뒷부분에 괄호가 있으면, 수식의 일부로 사용할 수 있는 "함수"입니다. 뒷부분에 괄호가 없으면 수식에서 사용할 수 없는 "명령문"입니다.

```
High 32          ' 명령문 형식. 32번 포트를 On시킵니다.
A = ADIN(0)      ' 함수 형식. 0번 채널에서 AD입력을 수행합니다.
```

In( )

Variable = IN(Port)

Variable : 결과가 저장될 변수

Port : 사용 가능한 I/O를 가리키는 변수 또는 상수

MSB제품은 24V입력회로가 내장되어 있습니다. 해당 포트로 24V를 입력하면 In( )함수에서 1을 읽어옵니다.

Out

OUT Port, Value

Port : 사용 가능한 I/O를 가리키는 변수 또는 상수

Value : I/O에 출력할 값을 가지고 있는 변수 또는 상수

MSB 제품의 출력포트는 NPN TR 또는 릴레이로 되어 있습니다. Out명령어로 1을 출력하면 NPN TR 또는 릴레이가 ON상태가 됩니다. 0을 출력하면 NPN TR또는 릴레이가 OFF상태가 됩니다.

High

HIGH Port

Port : 사용 가능한 I/O를 가리키는 정수형 변수 또는 상수

출력포트를 On상태로 만듭니다.

Low

LOW Port

Port : 사용 가능한 I/O를 가리키는 변수 또는 상수

출력포트를 Off상태로 만듭니다.

여기서 잠깐

MSB6XX시리즈는 8~31까지는 입력으로, 32~63까지는 출력으로 설정되어 있습니다. 유저가 별도로 입/출력 방향설정을 해주지 않아도 됩니다.

Adin( )

Variable = ADIN (Channel)

Variable : 결과가 저장될 변수

Channel : AD입력 채널

입력중인 아날로그 값을 읽어서 지정한 변수에 저장합니다. 아날로그 입력 기능이 있는 MSB제품에서만 사용할 수 있습니다.

A/D입력 채널	종류	결과값
0~3	0~20mA	0~1023
4~7	0~10VDC	0~1023

```
#Include "MSB6XX"
Dim A As Integer
Do
        A=Adin(0)
        Debug Goxy,5,3
        Debug dec5 A
        Delay 200
Loop
```

위 예제 프로그램은 채널0으로부터 값을 읽어 Debug창에 표시하는 프로그램입니다.

시스템 관련 명령

Debug

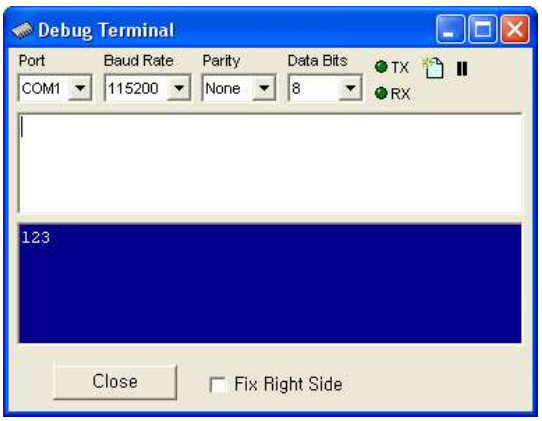
DEBUG data

data : PC로 전송할 데이터

"디버그 터미널"로 데이터를 전송하는 명령입니다. MSB에서는 디버그 명령을 사용해서, 동작 중 보고 싶은 변수 값을 PC상에 표시합니다.

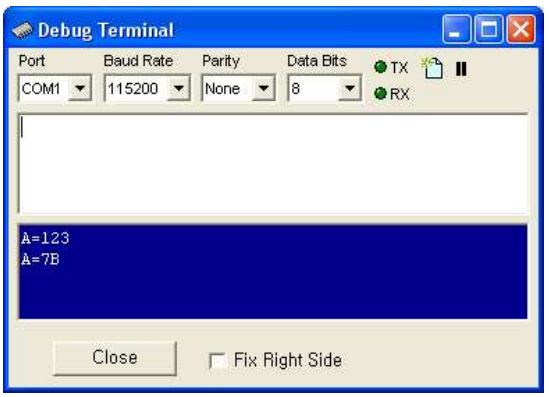
Debug명령이 동작하기 위해서는, MSB와 PC가 연결된 상태이어야 합니다. 이 상태에서 Debug명령이 수행되면, "디버그 터미널"에 그 값이 표시됩니다.

```
#include "MSB6XX"
Wait 1000 '잠시대기
Dim A As Integer
A = 123
Debug Dec A
```



DEC와 HEX를 사용하여 표시할 변수의 진법을 변환합니다. 아래의 예처럼 DEC나 HEX뒤에 ?를 기호를 적어준다면 변수명과 함께 표시됩니다.

```
#include "MSB6XX"
Wait 1000 '잠시대기
Dim A As Integer
A = 123
Debug Dec? A,Cr
Debug Hex? A,Cr
```



Wait

Wait value

밀리 초단위로 시간지연 시키는 명령어입니다. 단, 10밀리 초간격으로 인식하고 동작합니다. 즉 Wait 15라고 하면 15밀리초를 딜레이 하지 않고, 10밀리초를 딜레이합니다.

```
Wait 10          ‘ 10밀리 초를 딜레이 합니다.
Wait 200         ‘ 200밀리 초를 딜레이 합니다.
Wait 2000        ‘ 2 초를 딜레이 합니다.
```

Ramclear

베이직과 레더의 데이터 메모리를 전부 0으로 만듭니다.

LADDER 관련명령

Set ladder On

LADDER를 실행 개시 하는 명령입니다.

Alias

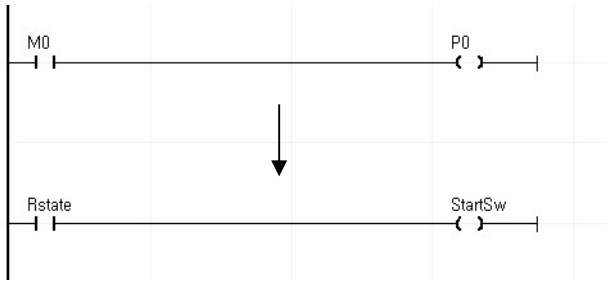
ALIAS Relayname = AliasName

Relayname : P0, M0, T0등의 릴레이 이름

AliasName : 릴레이 이름 대신 사용할 별명이름

P0, M0, C0등과 같은 릴레이 이름대신 사용할 "별명"을 선언하는 명령입니다. 선언된 별명을 사용하면 보다 쉽게 LADDER를 작성하고 이해할 수 있습니다.

```
Alias M0 = Rstate
Alias M1 = Kstate
Alias P0 = StartSw
```



Usepin

USEPIN I/O, In/Out, AliasName

I/O : I/O포트 번호

In/Out : 입출력상태, In또는 Out

AliasName : 릴레이 이름 대신 사용할 별명이름 (생략가능)

파워 On 시 모든 I/O포트는 BASIC쪽에서 사용하도록 되어 있습니다. 이중 LADDER에서 사용할 I/O포트를 정의하는 명령입니다.

```
USEPIN 8,IN      ' 8 번포트는 레더쪽에서 사용합니다.
```

USEPIN명령에서는 동시에 입출력 상태와 "별명"도 같이 정의할 수 있습니다.

```
USEPIN 8, IN, START
USEPIN 9, IN, BKEY
USEPIN 32, OUT, MOTOR
```

통신관련 명령

Opencom

OPENCOM channel, baudrate, protocol, recvsize, sendsize

channel : 사용채널 (1 부터 3)

Baudrate : 보레이트 (변수 또는 상수)

protocol : 프로토콜의 종류

recvsize : 수신용 버퍼의 크기 (최대 1024까지)

sendsize : 송신용 버퍼의 크기 (최대 1024까지)

RS232를 사용하기 위해서 반드시 소스 프로그램 초기에서 선언해야 되는 명령입니다. 보레이트에는 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200 중 하나를 적어주세요. protocol에는 데이터의 형식을 결정하는 코드를 적어줍니다.

비트 수	패리티	스톱비트	입력 값
8	NONE	1	3
8	EVEN	1	19
8	ODD	1	27
7	NONE	1	2
7	EVEN	1	18
7	ODD	1	26

```
OPENCOM 1, 19200, 3, 30, 20   '8비트,NONE패리티,1스톱비트로 설정
```

OPENCOM명령에서 송신용 버퍼와 수신용 버퍼의 크기를 결정할 수 있습니다.. 수신버퍼는 30~100이하, 송신버퍼는 30~50이하로 하면 무난히 사용할 수 있습니다.

송/수신버퍼를 너무 많이 할당해 놓으면, 그만큼 BASIC에서 변수로 쓸 데이터 영역이 줄어듭니다.

Set Modbus

SET MODBUS mode, slaveaddress, returninterval

mode : 0=ASCII, 1=RTU

slaveaddress : 슬레이브 어드레스 (1~254사이의 값)

returninterval : 응답 지연 간격 (1~255사이의 값, 생략시 1) 255는 25.5mS임

MODBUS 슬레이브 동작을 개시하기 위한 명령입니다. 반드시 OPENCOM이후에 작성해 주어야 하며, 채널1에서만 동작합니다. OPENCOM에서 지정한 보레이트와 비트 수, 패리티 등의 조건으로 외부기기와 통신합니다.

```
Opencom 1,115200,3,80,80 '수신버퍼는 50이상으로 설정하십시오.
Set Modbus 0,1,100       'ASCII모드, SLAVE ADDRESS=1로 설정, 응답지연 100
```

응답 지연이란, CUBLOC에서 MODBUS프레임을 수신한후, 일정시간을 대기하는 것을 말합니다. 100정도가 적당합니다. 100으로 지정하면 약 10mS 지연후 응답합니다.

HMI를 연결하려면 Opencom과 Set Modbus를 통신포트를 준비해 놓아야 합니다. 다음은 HMI 와 통신하기 위한 가장 기본적인 소스입니다.

```
#include "MSB6XX"          ' MSB6XX 시리즈를 위한 디바이스 선언.
Opencom 1,115200,3,200,200 ' 채널1을 115200,8,none,1stopbit로  오픈
Set Modbus 1,1,100         ' 모드버스 RTU 시작, 슬레이브 어드레스는 1,  수신응답은 약 10mS
Set Ladder On              ' 레더시작

Do
Loop
```

레더 로직

레더로직

레더로직은 "동시 다발적인 입력에 즉각 반응한다"라는 장점을 가지고 있습니다. 이것을 BASIC으로 구현하기는 힘듭니다. 예를 들어보겠습니다.

P8 이 입력되면 P32가 1초동안 ON을 유지하다가 꺼지는 동작을 BASIC으로 작성해보면 다음과 같습니다.

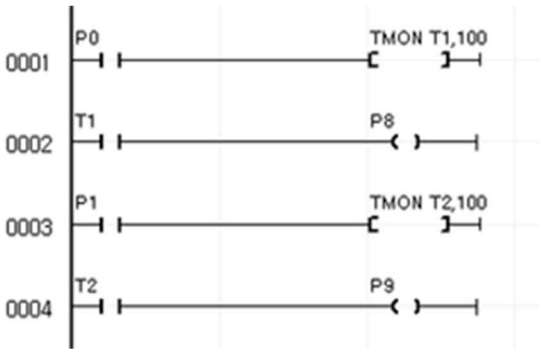
```
Do
 If In(8) = 1 Then
  High 32
  Wait 1000
  Low 32
 Endif
Loop
```

동작도 잘 됩니다. 여기에 P9번 입력이 들어오면 P33를 1초동안 유지시켜주는 동작을 추가해보세요.

```
Const Device = CB210
Do
 If In(8) = 1 Then
  High 32
  Wait 1000
  Low 32
 Endif

 If In(9) = 1 Then
  High 33
  Wait 1000
  Low 33
 Endif
Loop
```

이렇게 하면 동작될 거 같지만, 이건 반쪽짜리 프로그램입니다. P32이 켜져 있는 동안 P9 입력을 받을 수 없으니까요. 제대로 동작하려면, P8, P9 입력이 들어오는 즉시 P32, P33이 각각 1초씩 유지되어 야 합니다. 서로 간섭받지 말고요.

이러한 입력처리 상황은 자동화 현장에서 흔히 생기는 상황입니다. P8, P9에 센서가 연결되었다고 가정해 봤을때, 센서가 언제 동작할지는 아무도 모릅니다. 센서가 입력되는 즉시, 그에 상응한 출력이 바로 나와주어야, 기계가 제대로 동작할 것입니다. 이러한 처리는 레더에서는 쉽게 구현할 수 있습니다.



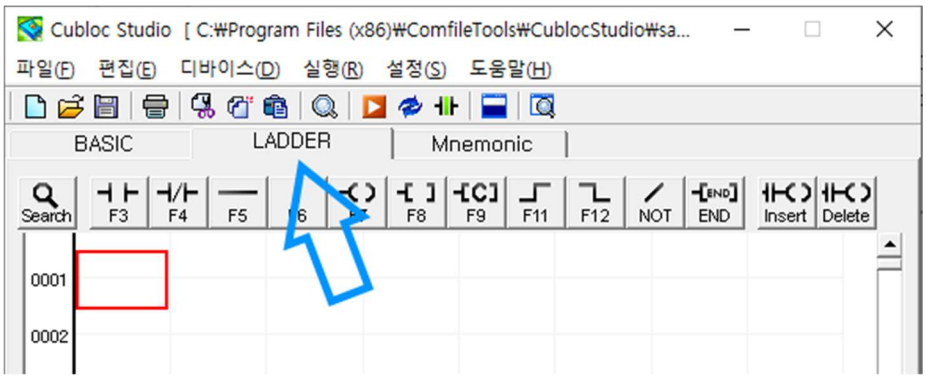
BASIC쪽 소스는 다음과 같습니다.

```
Const Device = CB210
Usepin 8,In
Usepin 9,In
Usepin 32,Out
Usepin 33,Out

Set Ladder On
Do
Loop
```

입출력 제어는 LADDER LOGIC쪽이 우수하므로, 주된 입출력 처리는 레더에서 수행하고, BASIC에서는 보조 역활을 수행하도록 코딩을 하시기 바랍니다.

CUBLOC STUDIO에서 레더로직 편집

CUBLOC STUDIO 의 LADDER 탭을 누르면 레더편집 상태로 전환됩니다.



기본 레더 작성

화면 상단에 있는 아이콘을 클릭하거나 펑션키를 누르면 현재 커서가 있는곳에 심볼이 표시됩니다. 이를 이용해서 레더의 골격을 먼저 그린후에 텍스트를 입력하세요.

레더의 끝

LADDER 의 가장 끝에는 반드시 END명령이 있어야 합니다. 상단에 있는 END아이콘을 눌러서 END명령을 입력하세요. CUBLOC STUDIO는 END 명령이 위치한 곳까지 번역하고, 저장합니다..

텍스트 입력/수정

심볼위에 커서를 놓고 ENTER를 누르면 그 위치에 TEXT를 작성할 수 있습니다. 이미 작성되어 있는 TEXT부분을 수정할때에도 해당위치에 커서를 놓은 뒤, ENTER키를 누르면 수정할 수 있습니다.

종 접속선의 추가 삭제

종 접속이 가능한곳에 위치하면 커서가 아래 모양처럼 변합니다. 이때 F6키를 누르면 해당위치에 종접속선이 표시됩니다. 이미 종접속선이 있는 곳에서 한번 더 F6을 누르면 종접속선이 지워집니다.

F6 을 누르면 이곳에 종접속선이 표시됩니다.

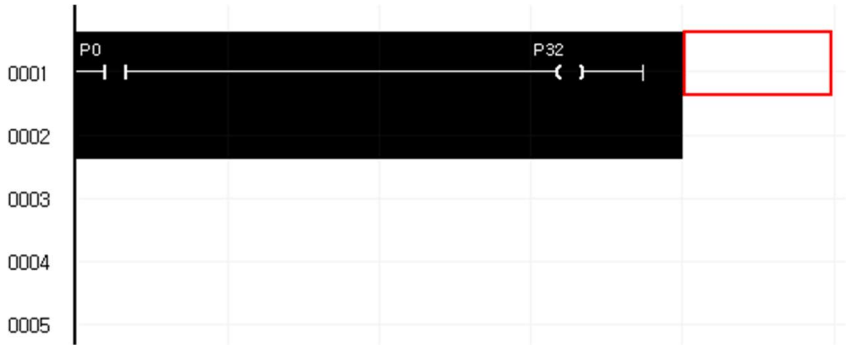
종 접속 선을 지우려면 해당위치에서 한 번 더 F6 키를 누르면 지워집니다.

셀 단위 삽입과 삭제

현재 커서의 위치에서 DEL키를 누르면 셀이 삭제되고, 뒤에 있던 부분이 당겨지게 됩니다. INS키를 누르면 빈칸이 삽입되고, 뒷부분은 밀려납니다.

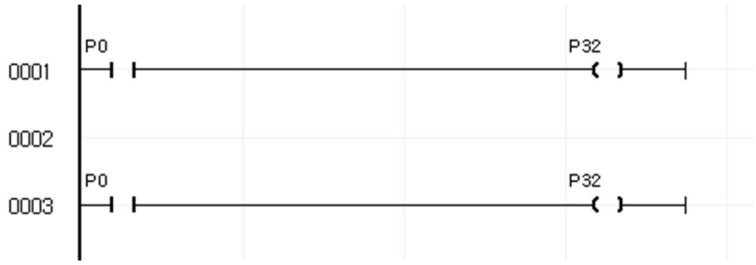
레더의 블록 이동, 복사

레더의 일부분을 선택하여 따로 보관하거나, 이동 복사 등을 할 수 있습니다.



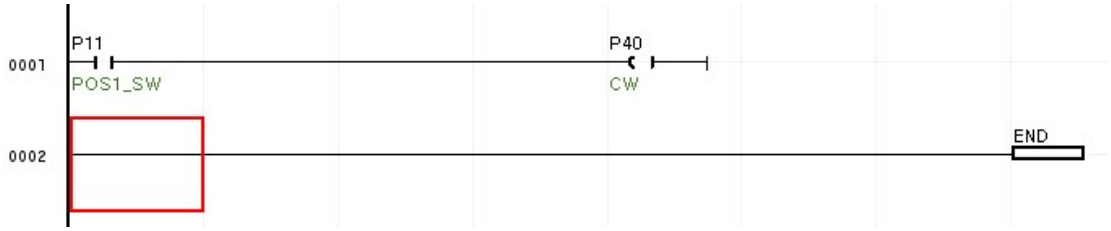
마우스를 이용하여 드레그-드롭한다면 원하는 영역을 선택할 수 있습니다. 선택된 영역은 반전되어 표시됩니다. 이 상황에서 CTRL-C를 누르면 선택된 영역이 버퍼로 복사됩니다.

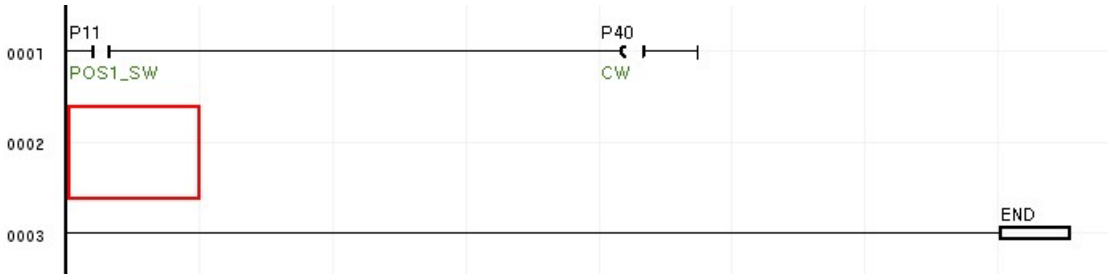
복사하고자 하는 지점으로 커서를 이동시킨 뒤 CTRL-V를 누르면 버퍼에 있던 내용이 표시됩니다.



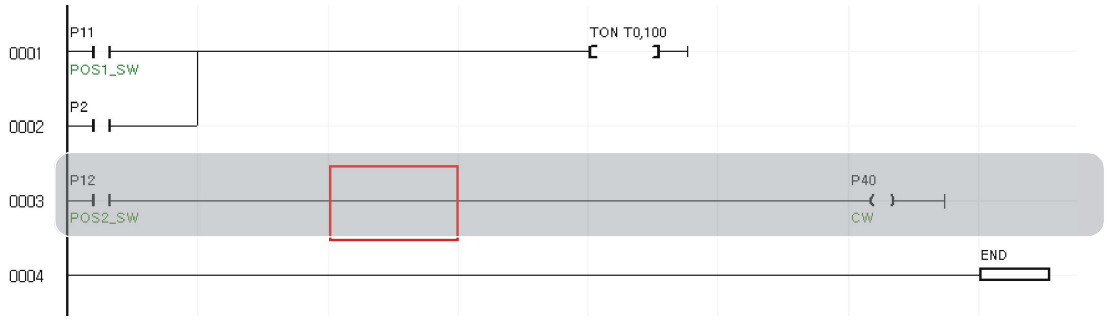
빈라인 삽입

CTRL+I를 누르면 커서위치 바로 위에 빈 라인이 삽입됩니다.

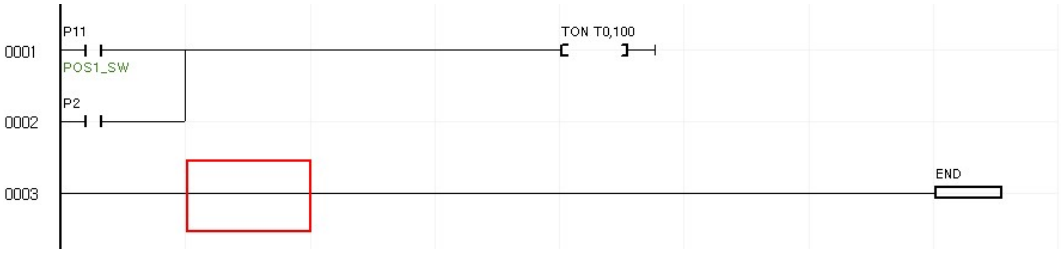




라인의 삭제



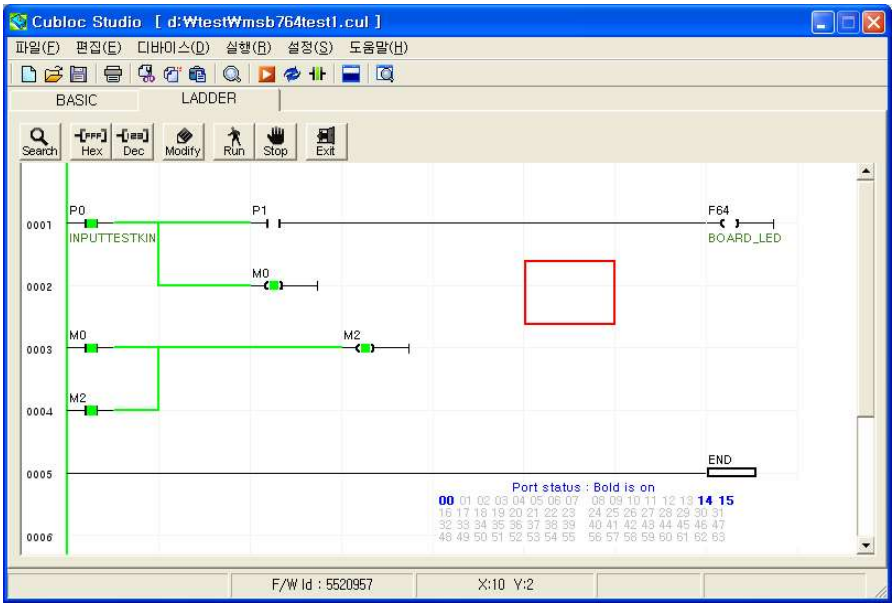
지우고 싶은 라인에 커서를 위치시킨 뒤 CTRL-D 를 누르면 해당 라인이 삭제됩니다.



모니터링

CUBLOC STUDIO에서는 LADDER LOGIC의 동작상황을 실시간으로 보여주는 "모니터링"기능을 지원합니다. 화면상단에 있는 툴 바에서 모니터링 ON/OFF 스위치를 누르면 모니터링 상태로 들어가거나, 빠져 나올 수 있습니다. 또는 F2 키를 누르면 모니터링 상태로 들어갑니다.

다음은 모니터링중인 화면입니다. ON상태인 접점은 "녹색블록"이 표시되며, 타이머, 카운터 등은 현재 값을 표시합니다.



화면하단에는 I/O포트의 상태가 표시됩니다. ON된 접점은 파란색의 볼드체로 표시됩니다. 0부터 63까지만 표시됩니다.

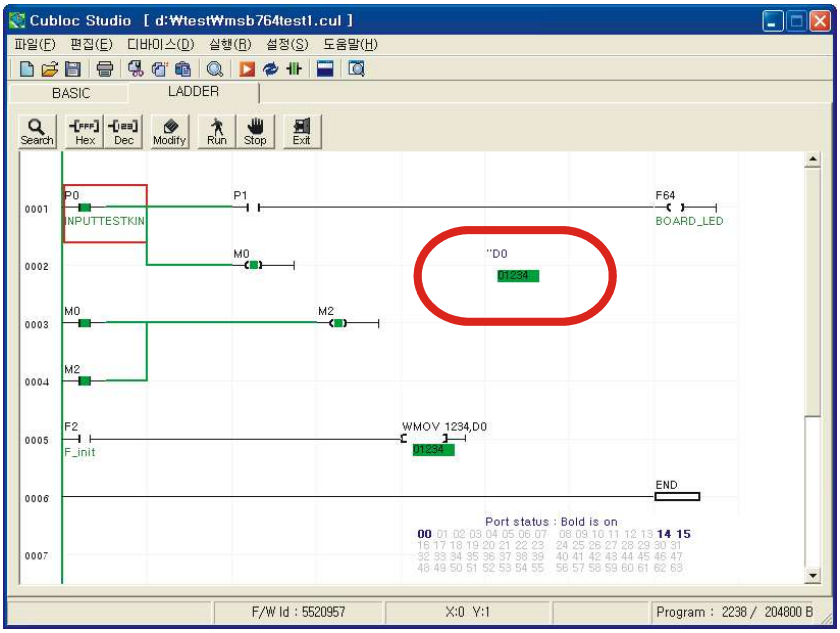
LADDER모니터링 상태에서는 편집 및 저장이 불가능합니다. 반드시 LADDER모니터링 상태를 끈 후에 다른 작업을 하기 바랍니다.

WATCH POINT

특정 릴레이나 메모리의 내용을 보고 싶을 때에는 WATCH POINT를 이용합니다. LADDER중 빈 부분에 "어포스트로피"를 두 개 연달아 작성한 뒤 보고 싶은 릴레이번호를 적어줍니다. (**따옴표가 아닌, 어포스트로피 두 개입니다.**) 다운로드 한 뒤, 모니터링 상태에서 와치포인트에 릴레이의 상태가 표시됩니다.

사용 예)

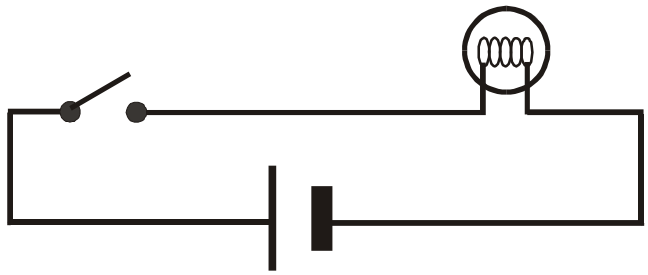
''D0 ''D12 ''T1



C, T, D는 16비트 값으로 표시 되며, 나머지 P, M, F는 접점형태로 표시됩니다.

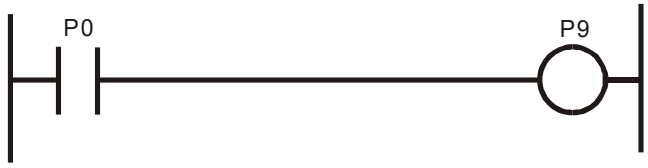
LADDER의 기초

하나의 스위치와 하나의 램프가 있는 회로를 동작시키기 위해서는 다음과 같은 회로가 필요합니다.



이 회로에서 전원을 생략하고 그리면 아래 그림처럼 표현할 수 있습니다. 양쪽의 두 선이 전원선입니다.

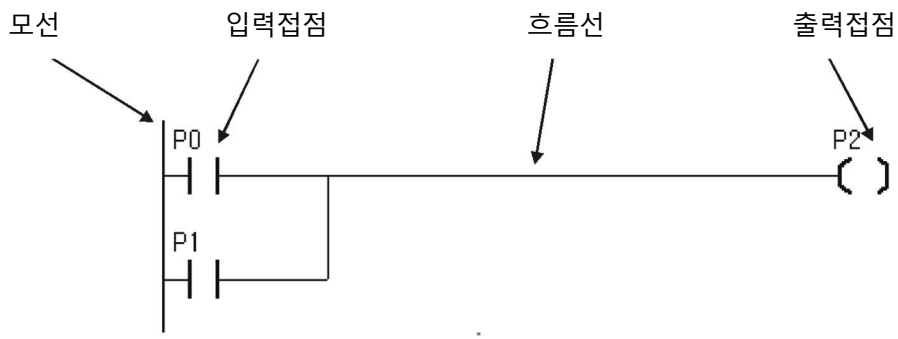
이 회로를 LADDER LOGIC으로 표현하면 다음과 같습니다.



이처럼, LADDER 로직은 전기 회로를 알기 쉬운 기호로 표시한 것입니다. 회로를 구성하는 스위치, 램프 등의 "기초소자"가 있고, 이 기초소자는 특정 포트나 메모리의 일부분에 연결되어 있습니다. 위의 그림에서 스위치는 P0포트에 연결하고 램프를 P1포트에 연결하는 상황을 가정한 것입니다.

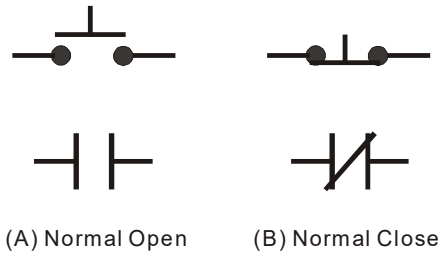
레더 로직 다이어그램의 기본 구성

레더로직은 다음과 같은 레더 다이어 그램으로 표현합니다. 레더 다이어그램은 "모선, 입력접점, 출력접점, 신호선"등으로 구성됩니다. (원래는 오른쪽에도 모선이 있지만, CUBLOC에서는 오른쪽 모선을 생략하고 있습니다.)



A 접점, B 접점

A접점은 평상시 열려있다가 신호가 들어오면 닫히는 "Normal Open" 입력입니다. 이와 반대로 B 접점은 평상시 닫혀있다가 신호가 들어오면 열리는 "Normal Close"입력을 말합니다.



기능 릴레이 심볼

기능 릴레이 (Function Relay)는 타이머, 카운터, 연산명령 등의 기능을 가지고 있는 릴레이 표현을 말합니다.

릴레이 및 레지스터 표현

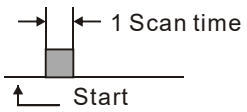
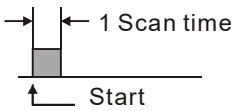
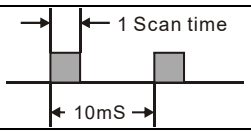
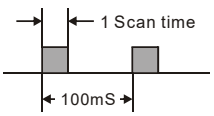
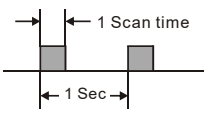
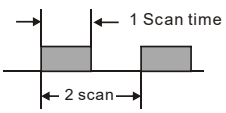
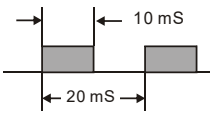
레더 로직에서 사용할 수 있는 레더전용 메모리입니다.

명칭	범위	단위	기능
입출력 릴레이 P	P0~P127	1 비트	외부 기기와의 인터페이스
내부릴레이 M	M0~M511	1 비트	내부 상태의 보존
특수기능 릴레이 F	F0~F127	1 비트	시스템 상태
타이머 T	T0~T99	16 비트 (1워드)	타이머용
카운터 C	C0~C49	16비트 (1워드)	카운터용
데이터 영역 D	D0~99	16비트 (1워드)	데이터보관

P, M, F는 비트단위로 되어 있고 T, C, D는 워드단위로 되어 있습니다. 비트 영역은 릴레이라고 부르고, 워드영역은 레지스터라고 부릅니다.

특수릴레이

특수릴레이를 통하여 MSB의 상태나 타이밍 정보, 시스템정보 등을 알 수 있습니다.

특수릴레이	설명	동작	별명 (Alias)
F0	상시 OFF		F_OFF
F1	상시 ON		F_ON
F2	**최초에 1 SCAN 만 ON**		F_INIT
F3	**파워온시 1 SCAN 만 ON** F3 이 먼저 ON 된후, 바로 뒤에 F2 가 ON 됩니다.		F_POR
F8	10mS 마다 1 SCAN 만 On		
F9	100mS 마다 1SCAN 만 On		F_100mS
F10	1 초 마다 1 SCAN 만 On		F_1Sec
F16	1 스캔타임간격으로 ON/OFF 를 반복		F_SCAN
F24	10mS 마다 ON/OFF 를 반복		F_10MS
F25	20mS 마다 ON/OFF 를 반복		F_20MS
F26	40mS 마다 ON/OFF 를 반복		F_40MS
F27	80mS 마다 ON/OFF 를 반복		F_80MS
F28	160mS 마다 ON/OFF 를 반복		F_160MS
F29	320mS 마다 ON/OFF 를 반복		F_320MS
F30	640mS 마다 ON/OFF 를 반복		F_640MS

https://www.comfile.co.kr/download/cubloc/cubloc_ladder_manual.pdf

레더로직 중심 사용설명서입니다. 레더에 대한 구체적인 설명을 보실 수 있습니다.

Ladder실행을 위한 최소한의 BASIC프로그램

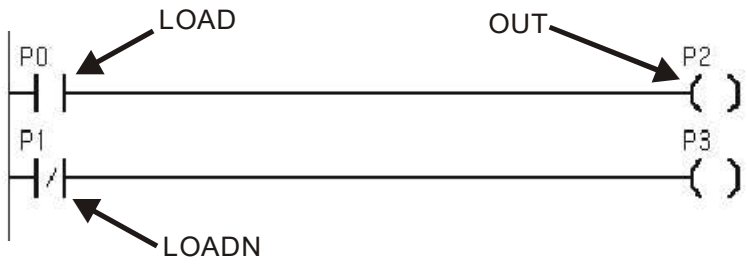
LADDER를 실행하기 위해서는 다음과 같이 "디바이스 선언", "포트선언", "LADDER실행 개시"등을 위한 최소한의 BASIC프로그램이 필요합니다.

```
#include "MSB6XX"          '디바이스 모델을 선언부

Usepin 0,In,START          '레더에서 사용할 입출력 포트
Usepin 1,In,RESETKEY
Usepin 2,In,BKEY
Usepin 3,Out,MOTOR

Alias M0=RELAYSTATE        '별명 선언
Alias M1=MAINSTATE

Set Ladder On              '레더의 시작 지시

Do
Loop                       '무한루프
```

레더로직 기본 명령

LOAD, LOADN, OUT

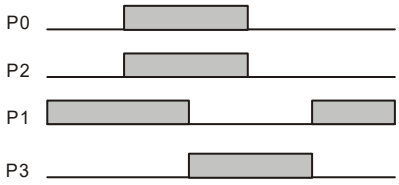
LOAD는 A접점의 시작, LOADN은 B접점의 시작입니다. OUT은 출력접점입니다.



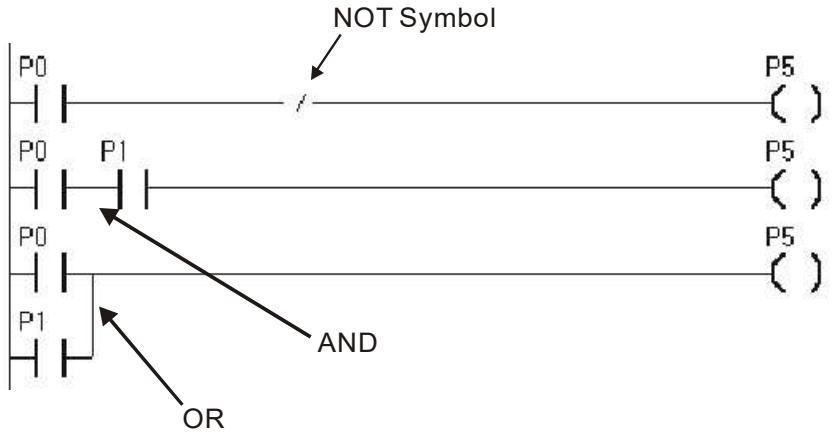
LOAD, LOADN, OUT명령에 사용할 수 있는 릴레이를 정리한 표입니다.

가능영역	P	M	F	C	T	D	상수
LOAD LOADN	O	O	O	O	O		
OUT	O	O					

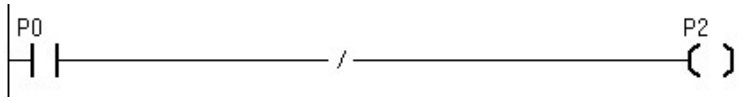
P릴레이는 I/O포트에 연결되어 있으므로, 포트로부터 입출력 되는 상황이 그대로 전달됩니다. M.F등 다른 릴레이는 메모리 내부에 저장되는 것이므로 외부 I/O포트에 전달되지 않습니다.



NOT, AND, OR



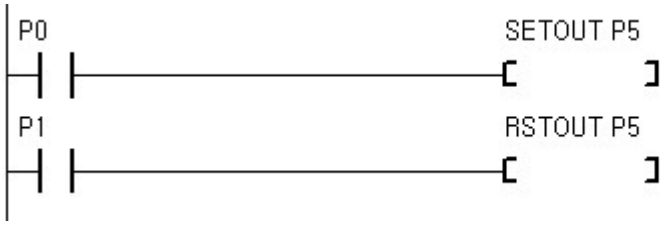
NOT심볼을 만나면 직전까지의 연산결과를 반전시킵니다. 화면 상단의 NOT심볼을 누르거나 슬레쉬 "/" 키를 누르면 커서위치에 NOT이 표시됩니다.



이 레더는 다음과 같이 써도 동일한 동작을 수행합니다. 즉 A접점과 NOT이 만나면 B접점과 같은 동작을 수행합니다.

SETOUT, RSTOUT

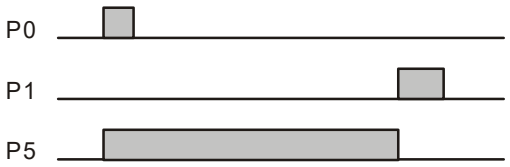
SETOUT은 입력이 있을 때 출력을 ON하고, ON상태를 계속 유지합니다. 반대로 RSTOUT은 입력이 있을 때 출력을 OFF한 뒤 OFF상태를 계속 유지합니다.



SETOUT, RSTOUT명령에서 사용할 수 있는 릴레이는 아래와 같습니다.

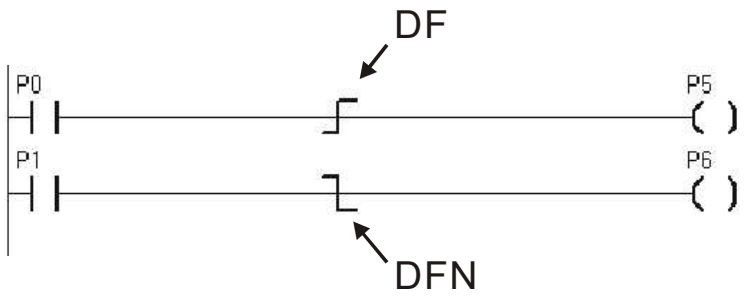
가능영역	P	M	F	C	T	D	상수
SETOUT	O	O	O				
RSTOUT	O	O	O				

위의 레더 실행결과를 타임차트로 표시한 것입니다.

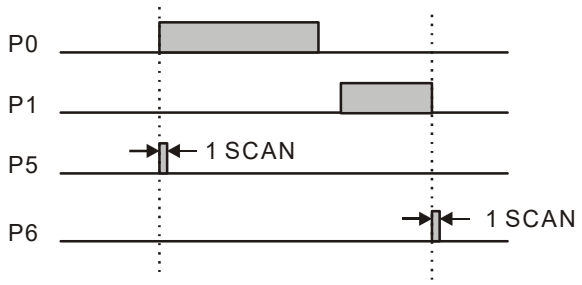


DF, DFN

DF는 입력조건에서 상승에지 (OFF -> ON) 상황이 발생하면 출력접점을 1스캔시간 동안 ON 하는 명령입니다. 반대로, DFN은 입력조건에서 하강에지 (ON -> OFF)상황이 발생하면 1스캔시간 동안 ON하는 명령입니다.



DF, DFN은 심볼만으로 동작됩니다. 별도로 릴레이를 할당할 필요가 없습니다. DF, DFN의 동작 파형은 다음과 같습니다.



TON, TAON

TON [릴레이], 설정값

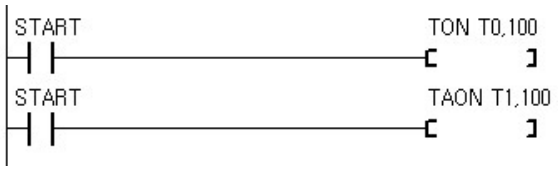
TAON [릴레이], 설정값

릴레이 : T 릴레이만 사용가능합니다.

설정값 : 1에서 65535까지의 상수값 또는 D레지스터 사용가능

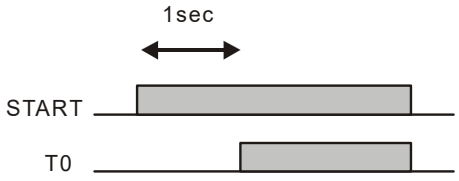
입력 조건이 ON이 되면 타이머 값이 증가되어, 설정 치에 도달하면 출력접점이 ON됩니다. 시간 단위가 틀린 두 종류의 ON TIMER가 있습니다.

타이머종류	분해능	최대치
TON	0.01 초	655.35 초
TAON	0.1 초	6553.5 초



TON, TAON명령에는 2개의 인수가 있습니다. 타이머번호는 T레지스터중 하나의 값을 지정하고, 설정 치로는 상수 및 데이터 레지스터등을 사용할 수 있습니다.

위의 LADDER에서 START입력이 들어오면 T0타이머가 증가됩니다. 0.01초마다 1씩 증가되어 증가치가 100 (1초경과) 이 되면 T0접점이 ON 됩니다. TAON을 사용한 T1타이머의 경우에는 0.1초마다 1씩 증가되어 10초가 경과되면 T1접점이 ON됩니다. 입력신호가 OFF되면 타이머는 초기값으로 설정되고, 타이머 접점도 OFF됩니다.

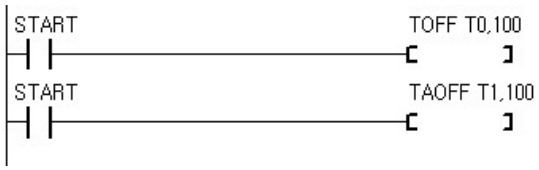


TOFF, TAOFF

TOFF [릴레이], 설정값
TAOFF [릴레이], 설정값
릴레이 : T 릴레이만 사용가능합니다.
설정값 : 1에서 65535까지의 상수값 또는 D레지스터 사용가능

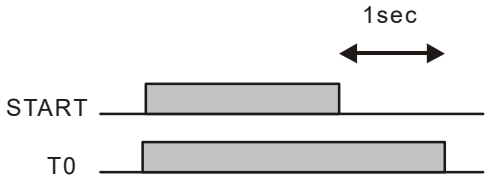
입력 조건이 ON이 되면 타이머접점은 바로 ON됩니다. 이후 입력이 OFF되면 타이머접점은 OFF 되지 않고, 설정치 만큼의 시간이 경과된 뒤 OFF됩니다. 시간 단위가 틀린 두 종류의 OFF TIMER 가 있습니다.

타이머종류	분해능	최대치
TOFF	0.01 초	655.35 초
TAOFF	0.1 초	6553.5 초



TOFF, TAOFF명령에는 2개의 인수가 있습니다. 타이머번호는 T레지스터중 하나의 값을 지정하고, 설정치로는 상수 및 데이터 레지스터등을 사용할 수 있습니다.

위의 LADDER에서 START입력이 들어오면 T0접점은 바로 ON됩니다. 이후 START접점이 OFF된 뒤 타이머가 증가됩니다. 0.01초마다 1씩 증가되어 증가치가 100 (1초경과) 이 되면 T0접점이 OFF 됩니다. TAON을 사용한 T1타이머의 경우에는 0.1초마다 1씩 증가되어 10초가 경과되면 T1접점이 OFF됩니다.



TMON, TAMON

TMON [릴레이], 설정값

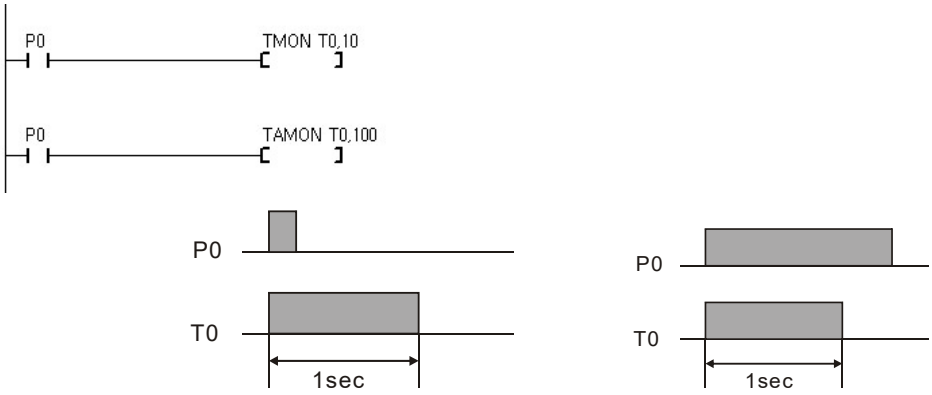
TAMON [릴레이], 설정값

릴레이 : T 릴레이만 사용가능합니다.

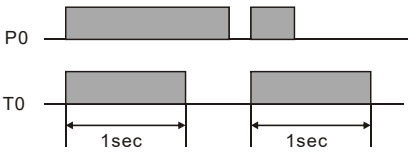
설정값 : 1에서 65535까지의 상수값 또는 D레지스터 사용가능

입력 조건이 잠깐 ON 되도, 바로 타이머 출력 접점이 ON, 시간 경과후 OFF 되는 트리거 타이머 입니다. 짧은 입력신호에도 일정시간 출력이 유지되야 하는 경우에 사용하십시오.

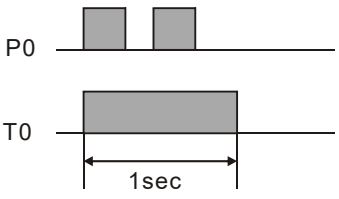
타이머종류	분해능	최대치
TMON	0.01 초	655.35 초
TAMON	0.1 초	6553.5 초



입력 신호가 계속 들어오는 경우라도, 타이머는 정해진 시간만큼 경과된 뒤 OFF됩니다.



이후 입력이 다시 들어온다면, 타이머는 동작됩니다.



타이머 동작구간에 중복입력이 되어도, 무시합니다.

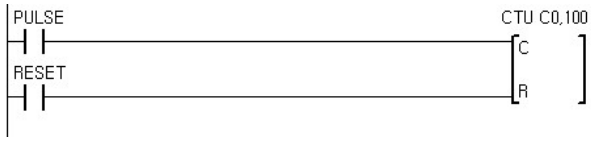
CTU

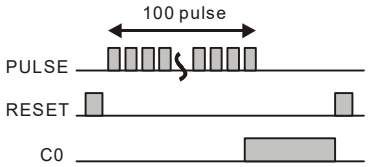
CTU [릴레이], 설정값

릴레이 : C 릴레이만 사용가능합니다.

설정값 : 1에서 65535까지의 상수값 또는 D레지스터 사용가능

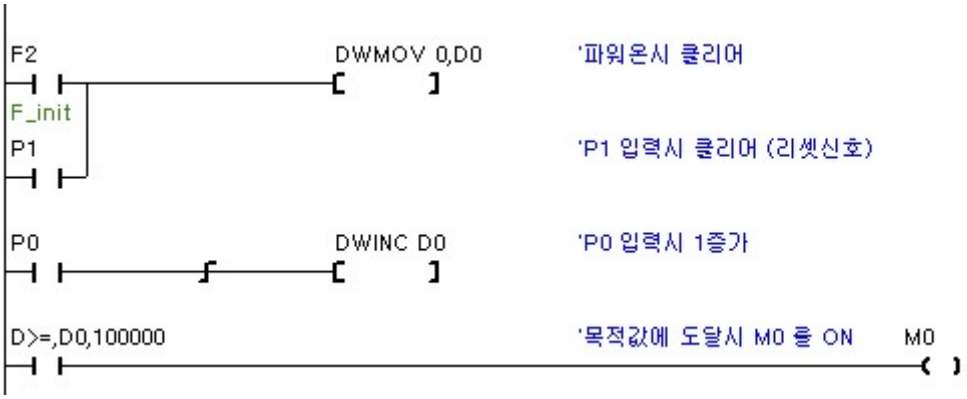
업 카운터 명령입니다. 입력이 들어오면 카운터 값을 1 증가 시킵니다. 카운터 값이 설정치와 일치하면 카운터 접점을 ON합니다. 리셋 입력이 들어오면 카운터 값은 0이 됩니다. 최대 65535까지만 사용가능합니다. 설정치에 도달하고, 접점이 ON된 이후에는 입력시그널이 들어와도 무시합니다.





더블워드 카운터

CTU명령은 최대 65535까지만 카운트가 가능합니다. 만약 그 이상의 값을 카운트하고 싶다면, 다음처럼 DWINC명령을 사용하시면 됩니다.



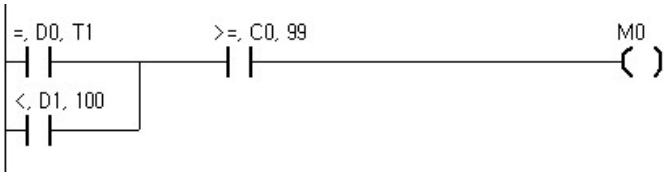
P0가 입력되면 D0,D1에 있는 더블워드값이 증가됩니다. 이 값이 100000 에 도달하면 M0가 ON 됩니다. 리셋시 또는 P1이 입력되면 D0,D1이 클리어됩니다.

비교명령

두 개의 워드(16비트), 또는 더블워드(32비트)값을 비교해서 조건을 만족하면 접점을 ON합니다. 총 12개의 비교명령이 있습니다.

비교명령	비교대상	동작설명
=, s1, s2	워드(16 비트)	S1 과 s2 가 서로 같을 때 접점이 On 됩니다.
<>, s1, s2	워드(16 비트)	S1 과 s2 가 서로 다를 때 접점이 On 됩니다.
>, s1, s2	워드(16 비트)	S1 > s2 일 때 접점이 On 됩니다.
<, s1, s2	워드(16 비트)	S1 < s2 일 때 접점이 On 됩니다.
>=, s1, s2	워드(16 비트)	S1 >= s2 일 때 접점이 On 됩니다.
<=, s1, s2	워드(16 비트)	S1 <= s2 일 때 접점이 On 됩니다.
D=, s1, s2	더블워드(32 비트)	S1 과 s2 가 서로 같을 때 접점이 On 됩니다.
D<>, s1, s2	더블워드(32 비트)	S1 과 s2 가 서로 다를 때 접점이 On 됩니다.
D>, s1, s2	더블워드(32 비트)	S1 > s2 일 때 접점이 On 됩니다.
D<, s1, s2	더블워드(32 비트)	S1 < s2 일 때 접점이 On 됩니다.
D>=, s1, s2	더블워드(32 비트)	S1 >= s2 일 때 접점이 On 됩니다.
D<=, s1, s2	더블워드(32 비트)	S1 <= s2 일 때 접점이 On 됩니다.

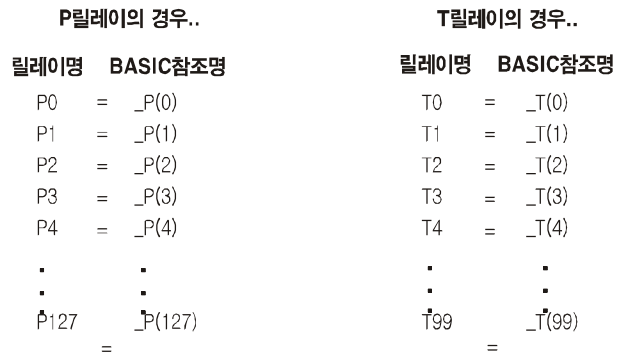
아래와 같이 여러 개의 조건을 AND, OR접속으로 사용할 수 있습니다.



D0 = T1이고 C0 >= 99 이면 M0이 ON됩니다. 또는 D1 < 100이고 C0 >= 99 이면 M0이 ON됩니다. 마치 베이직에서 IF D0 = T1 AND C0 >= 99 ....와 같은 식으로 AND, OR로 여러 개의 조건을 나열하는 것과 같습니다.

베이직과의 데이터 공유

BASIC에서 특정 배열변수를 사용하면 레더의 릴레이 영역을 억세스 할 수 있도록 되어 있습니다.



레더 억세스용 시스템 배열	억세스 단위	LADDER 의 해당영역
_P	비트 단위 _P(0) ~ P(127)	P 릴레이 영역
_M	비트 단위 _P(0) ~ P(511)	M 릴레이 영역
_WP	워드단위 _WP(0) ~ _WP(7)	P 영역을 워드단위로 억세스
_WM	워드단위 _WM(0) ~ _WM(31)	M 영역을 워드단위로 억세스
_T	워드단위 _T(0) ~ _T(99)	T 영역 (타이머)
_C	워드단위 _C(0) ~ _C(49)	C 영역 (카운터)
_D	워드단위 _D(0) ~ _D(99)	D 영역 (데이터)

P와 M릴레이는 비트 단위로 억세스 되고, 나머지 C, T, D영역은 워드 단위로 억세스 됩니다. 다음은 BASIC에서 LADDER 데이터 영역을 억세스 하는 샘플 프로그램입니다.

```
_D(0) = 1234
_D(1) = 3456
_D(2) = 100
```

D영역에 어떤 값을 넣을 수 있습니다. 반대로, 레더로직에서 베이직 변수의 값을 고치거나, 참조할 수는 없습니다.

HMI

HMI

어떠한 기계를 만들더라도 유저 인터페이스 부분은 반드시 필요합니다. 아주 옛날에는 버튼이나 램프등을 기계 전면에 배치하는 식으로 해서 유저 인터페이스를 해왔습니다.

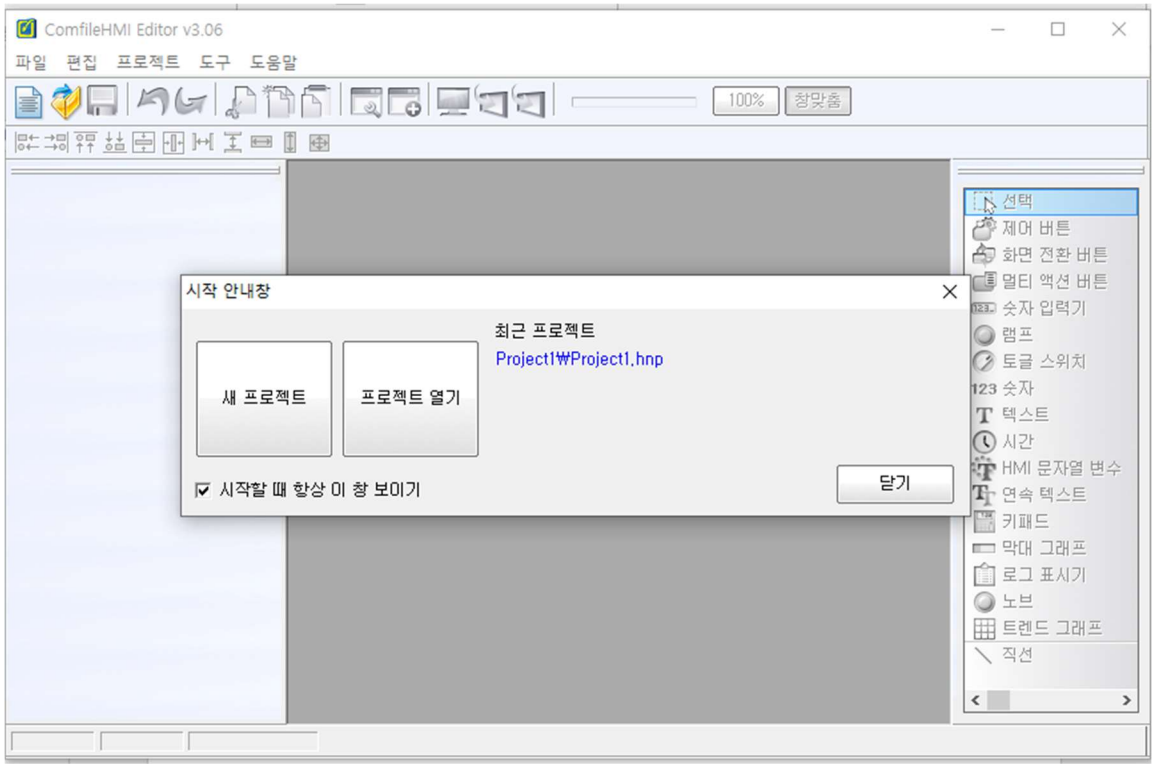
이 방법은 추후 기능 추가나 수정이 힘들고, 복잡한 유저 인터페이스를 하려면 수많은 버튼과 램프를 설치해주어야 한다는 단점이 있습니다.

HMI 를 사용하면 이보다는 세련되게 유저인터페이스를 해결할 수 있습니다. 뿐만 아니라 추후 보완이나 기능수정이 쉽고, 유지보수가 간편하다는 장점이 있습니다.

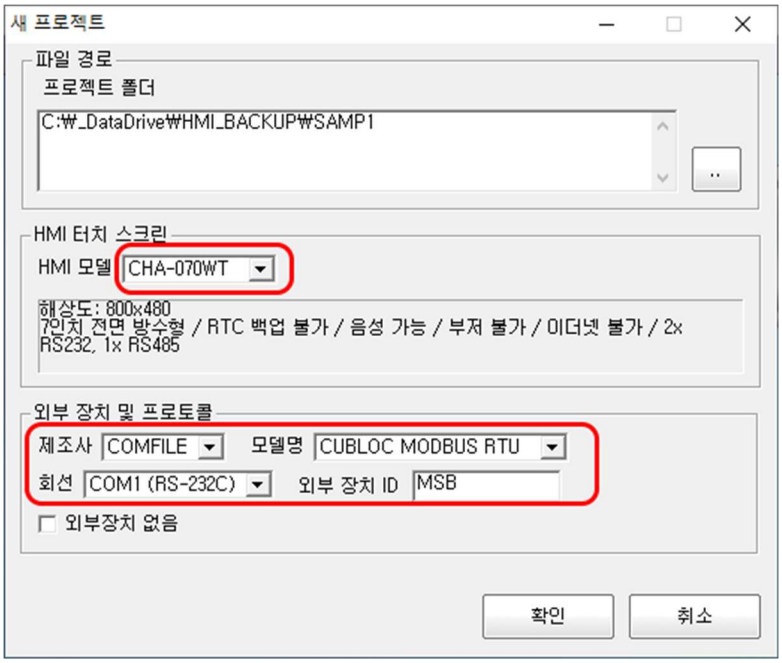
최종 제품이 한결 세련되어 보인다는 장점도 있네요.

ComfileHMI Editor

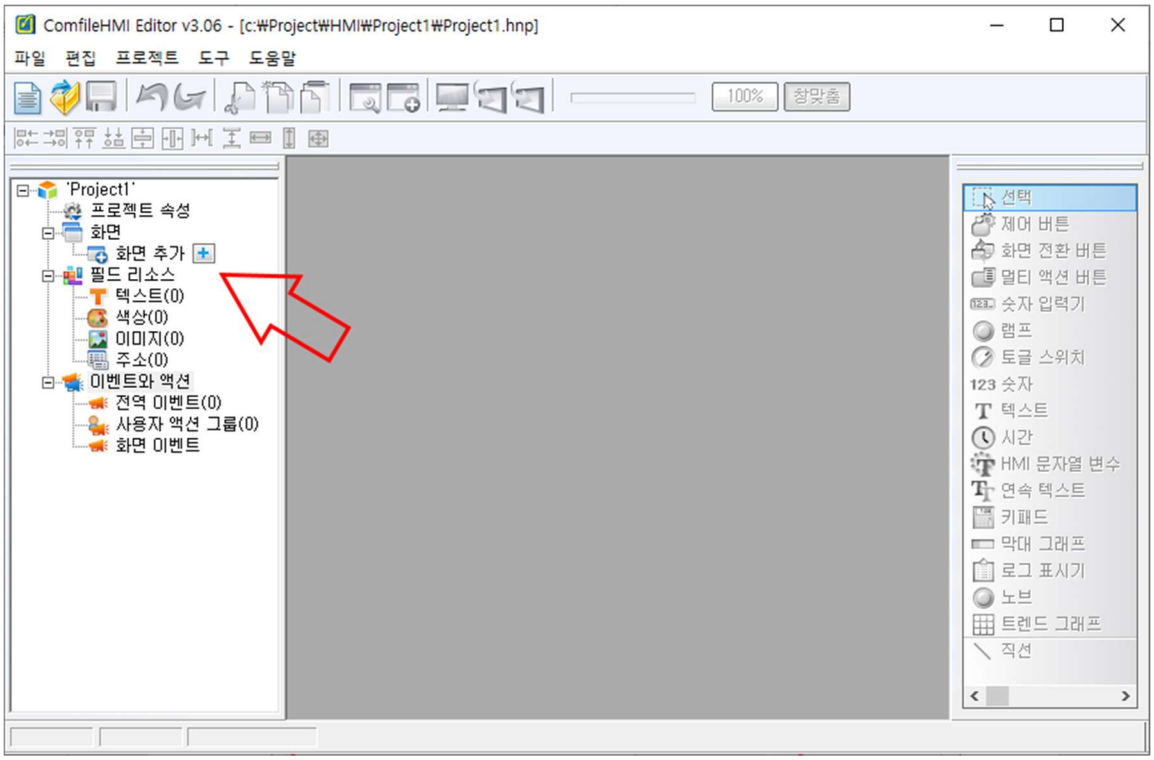
ComfileHMI 는 저희 회사에서 만든 HMI 제품입니다. 이를 사용하기 위해서 먼저 www.comfile.co.kr 자료실에서 ComfileHMI Editor 를 다운로드 받으세요. 실행하면 다음과 같은 화면이 표시됩니다.



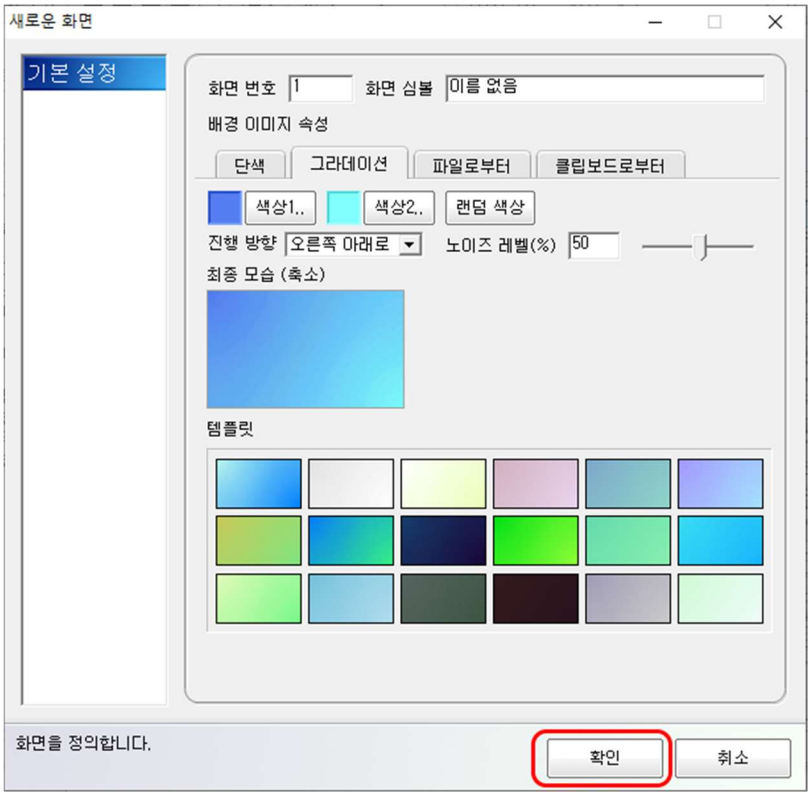
여기에서 새 프로젝트를 누르세요.



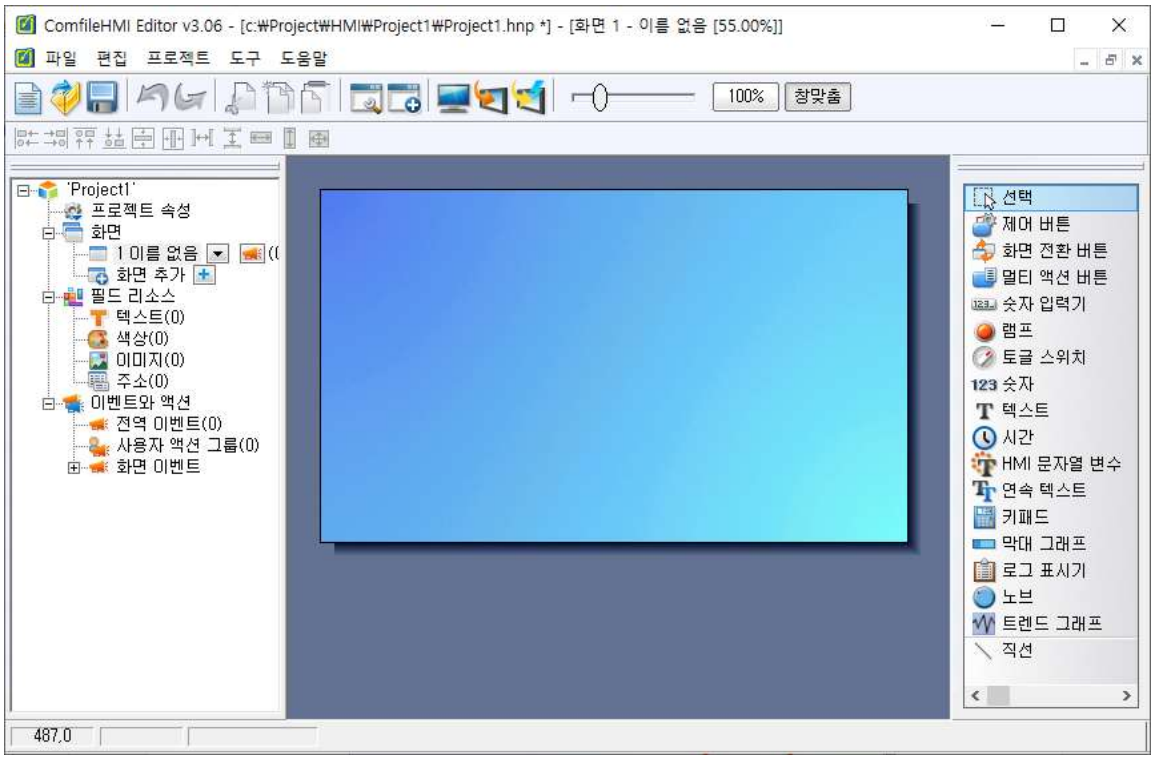
내용을 위와 같이 바꾼뒤 <확인>버튼을 누르세요. 다음 화면이 표시됩니다. 이제 화면 추가옆에 있는 + 버튼을 누르세요.



아래 화면이 표시되는데, 여기서는 그냥 <확인>을 누르세요.

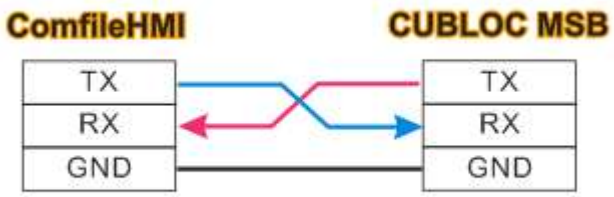


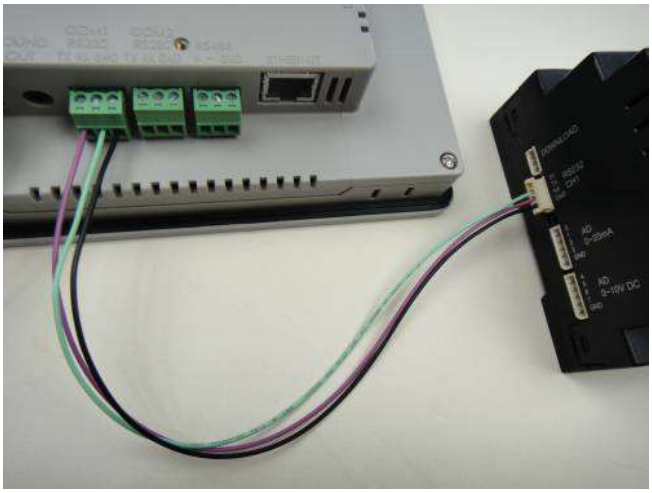
그러면 다음과 같은 상태가 됩니다. 이것으로 작화를 위한 준비가 끝났습니다.



MSB와 HMI 연결

MSB 와 HMI 는 3 가닥으로 연결합니다.



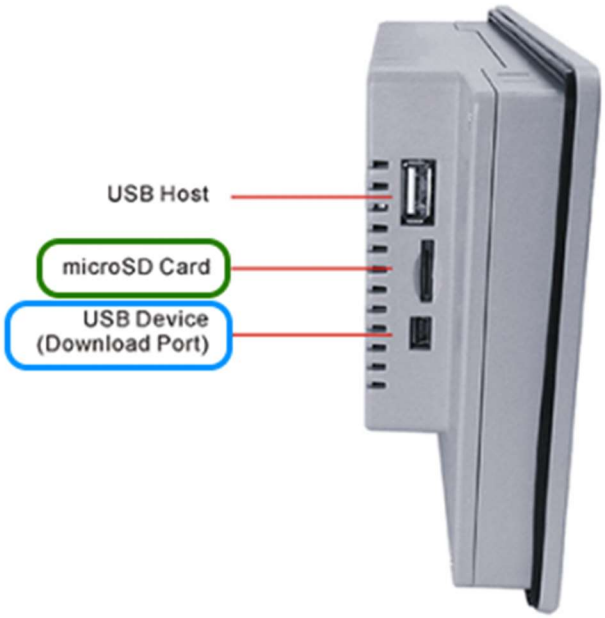


위 사진처럼 MSB 제품에 동봉된 케이블을 이용해서 연결하시면 됩니다.

HMI와 PC 연결

ComfileHMI Editor 로 작화한 화면을 HMI 쪽으로 다운로드하려면, PC 와 HMI 사이에도 연결이 필요합니다. 저희 회사에서 판매하는 HMI USB CABLE 이 필요합니다.

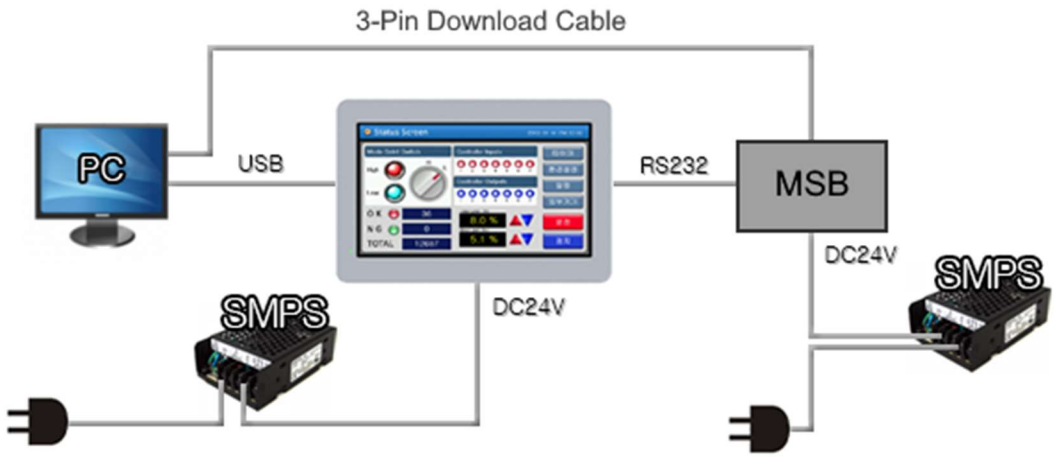
ComfileHMI 제품 우측에 케이블을 연결하세요.



CHA 시리즈 제품은 microSD 카드도 장착해주어야 합니다. 제품에 동봉된 microSD 카드를 슬롯에 장착해주세요.

연결 총정리

지금 여러분의 책상위에는 아래 그림과 같이 연결되어 있어야합니다.



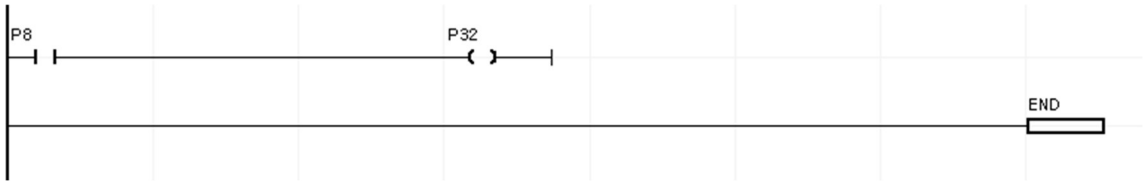
위와 같이 연결이 다 되어있다면, 이제부터는 MSB 와 HMI 에 다운로드하는 일만 남아있습니다.

MSB에 소스코드 다운로드하기

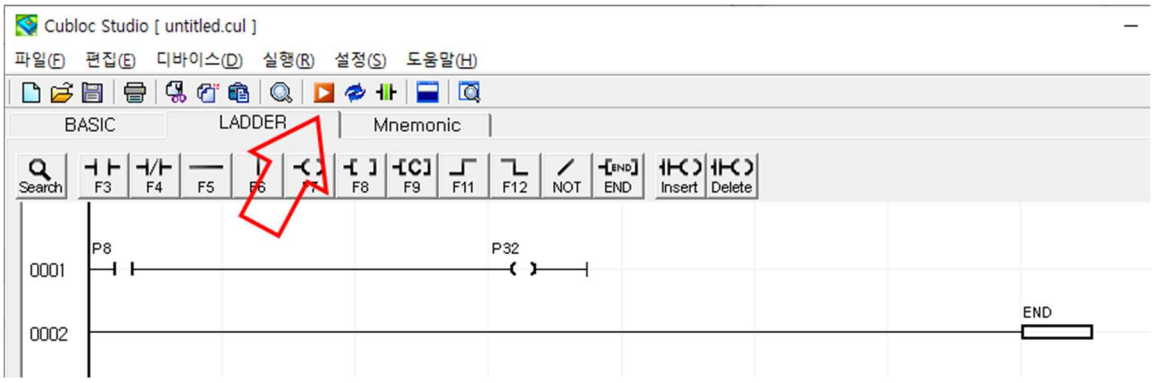
CUBLOC STUDIO 에 BASIC 쪽에는 다음과 같은 소스를 입력하세요.

```
#include "MSB6XX"              ' MSB6XX 시리즈를 위한 디바이스 선언.
Opencom 1,115200,3,200,200     ' 채널1을 115200,8,none,1stopbit로  오픈
Set Modbus 1,1,100             ' 모드버스 RTU 시작, 어드레스는 1, 수신응답은 100 (약 10mS)
Usepin 8,In                    ' 8번포트는 레더에서 입력으로 사용
Usepin 32,Out                  ' 32번 포트는 레더에서 출력으로 사용
Usepin 33,Out                  ' 32번 포트는 레더에서 출력으로 사용
Set Ladder On                  ' 레더시작

Do                             ' 메인루프
Loop
```

레더로직은 아래와 같이 입력하세요. 8 번포트 입력을 그대로 32 번 포트로 전달하는 레더입니다.



그 다음 빨간색 실행버튼을 눌러서 다운로드하세요.



만약 다운로드가 안된다면, 앞서 설명한 연결도를 보시고 빠진 부분이 있나 살펴보세요.

윈도우 모바일 센터 설치

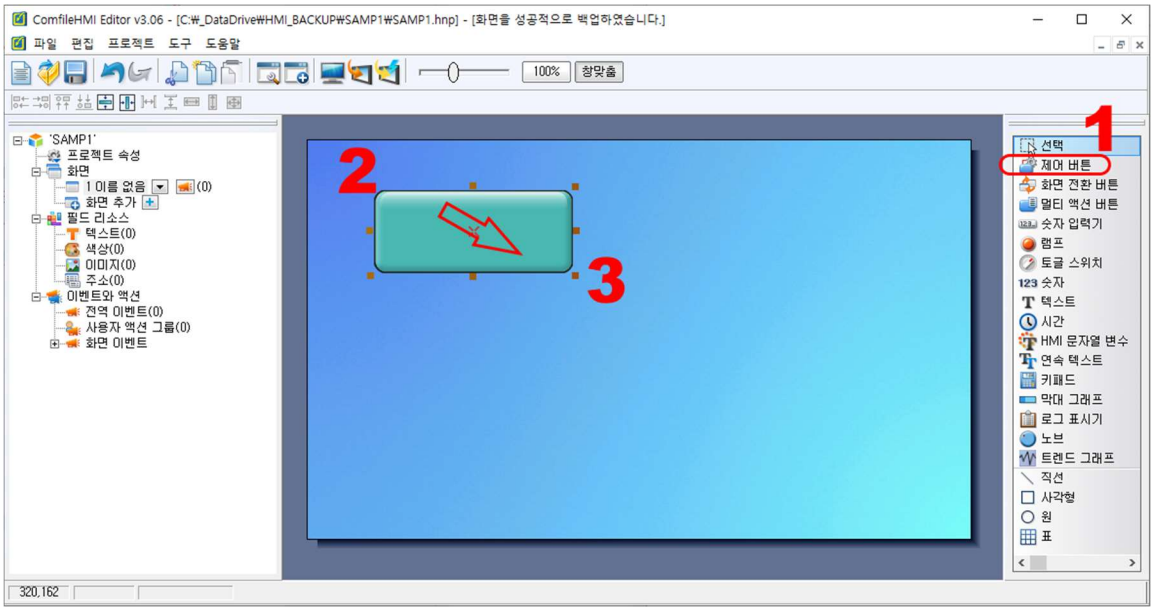
HMI 와 PC 를 USB 케이블로 연결하면 <윈도우 모바일 센터>가 자동으로 설치됩니다. 설치가 끝나면 아래와 같은 화면이 표시되는데, 여기에서 <장치를 설정하지 않고 연결>을 누르면 됩니다.

일단 연결된 뒤에는 ComfileHMI Editor 가 알아서 관리하기 때문에, 여러분은 특별히 신경쓰실 필요는 없습니다. 간혹 업데이트된 윈도우 10 PC 에서 인식을 못하는 경우가 있는데, 그럴때에는 아래 링크에 있는 지시를 따라서 패치를 진행하시면 됩니다.

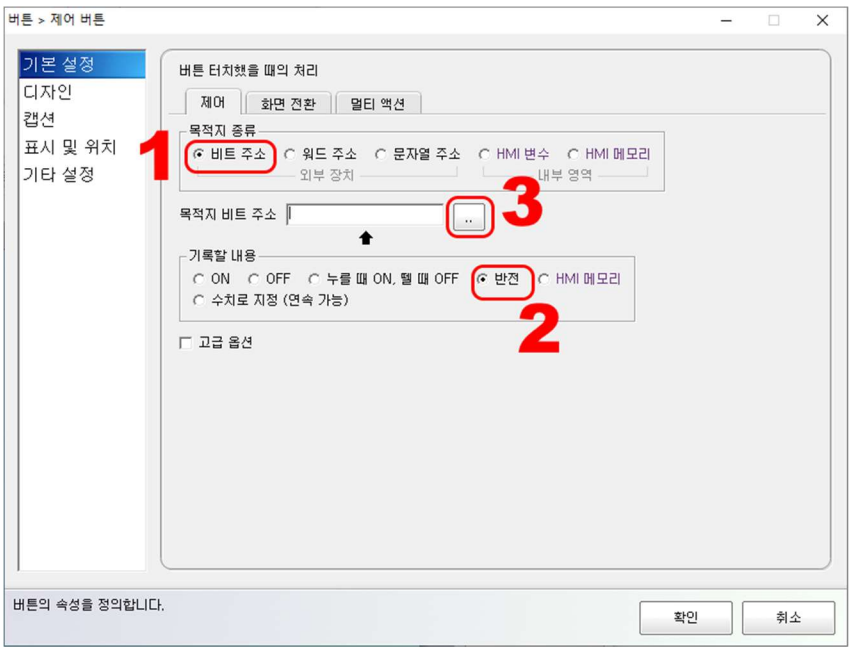
http://comfilewiki.co.kr/ko/doku.php?id=comfilehmi:hmieditor_win10bat:index

버튼 작화

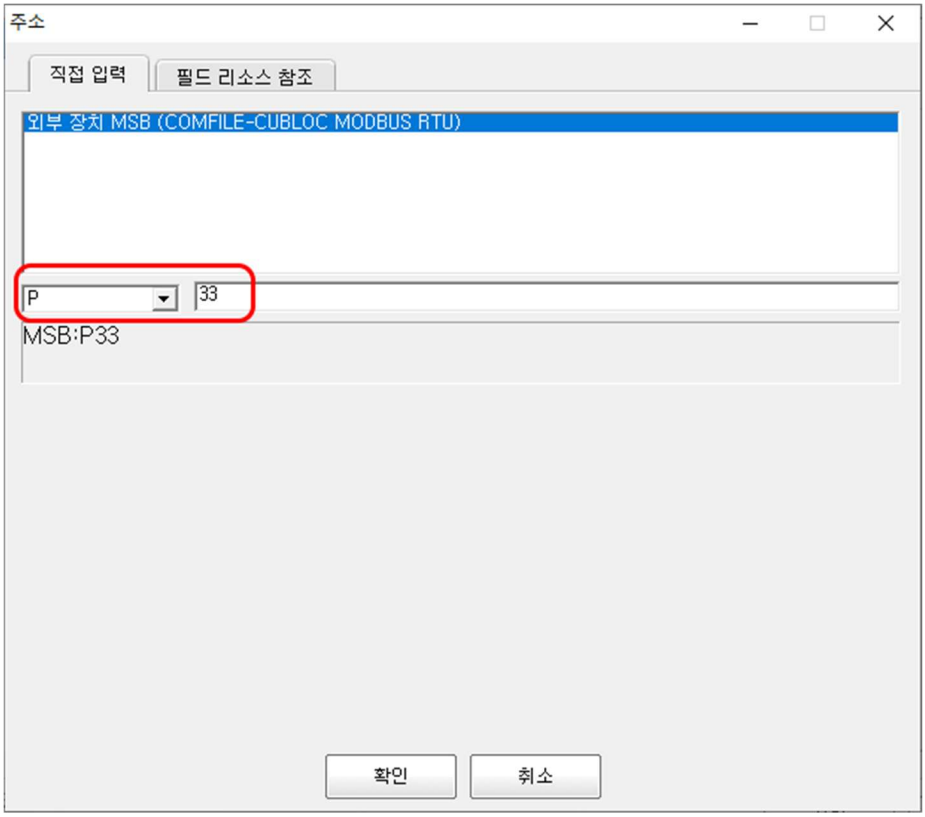
가장 먼저 버튼을 그려보겠습니다. 1 번위치에서 <제어버튼>을 선택하고, 2 번위치에서 클릭한뒤 3 번위치까지 끌면 버튼이 표시됩니다.



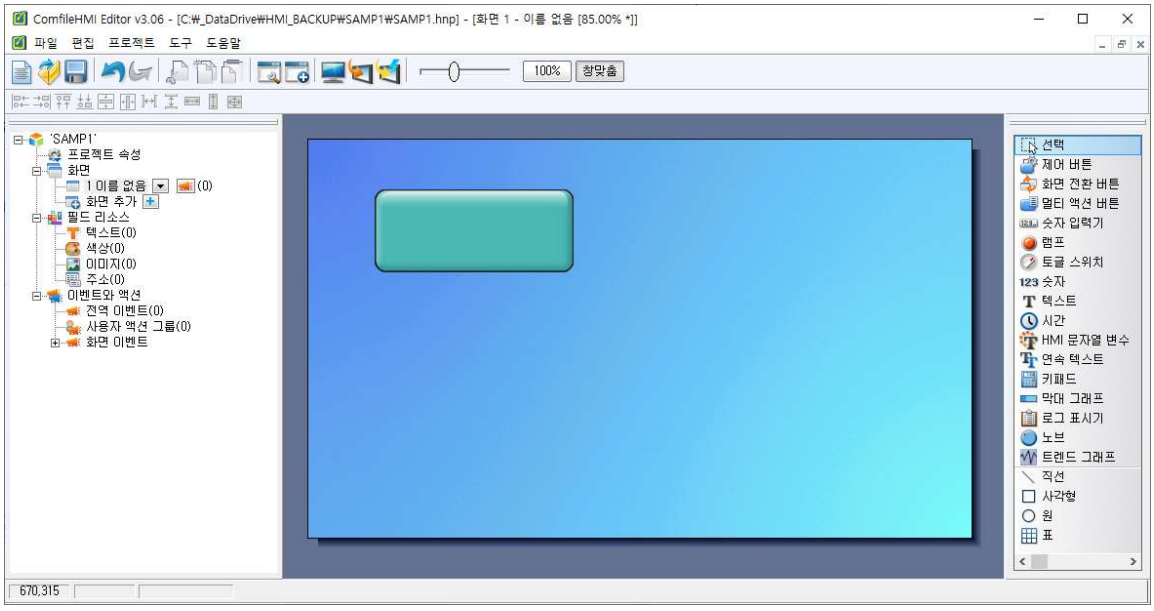
그려진 버튼을 더블클릭하면 아래와 같은 화면이 뜹니다. 1 위치의 비트주소를 선택하고 2 위치의 반전을 선택하고, 3 위치의 조그마한 박스를 클릭하세요.



그러면 주소를 입력하는 창이 나오는데, 아래처럼 P33 을 선택하세요.

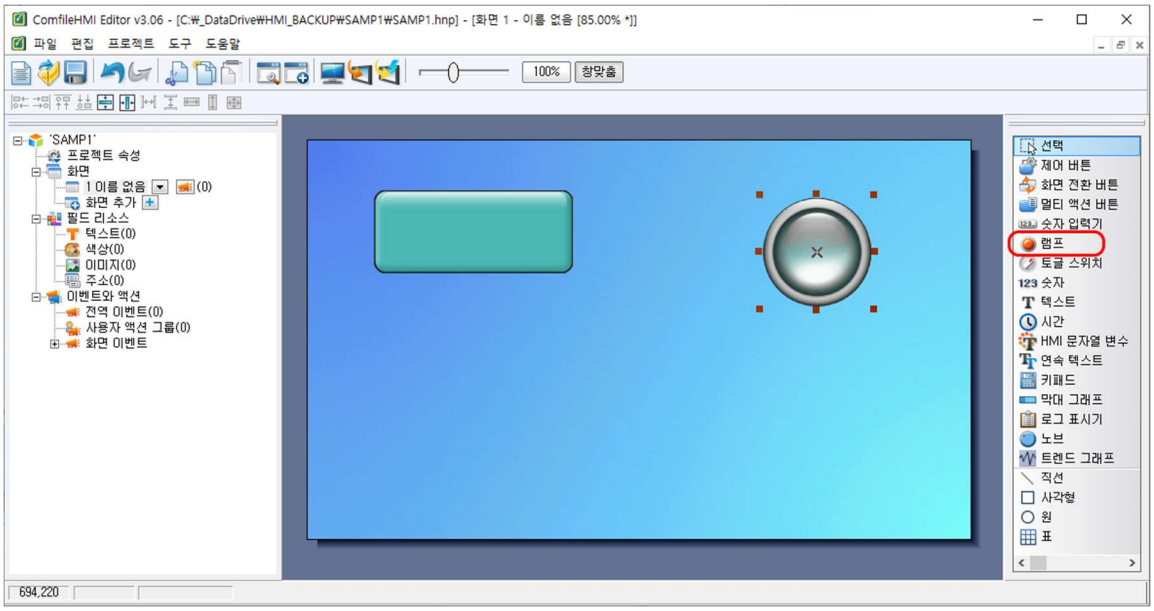


<확인>을 눌러서 열려진 창을 모두 닫으면 화면은 다음과 같은 상태가 됩니다. 버튼 하나를 그리고 이 버튼을 MSB 의 P33 과 연결한 것입니다.

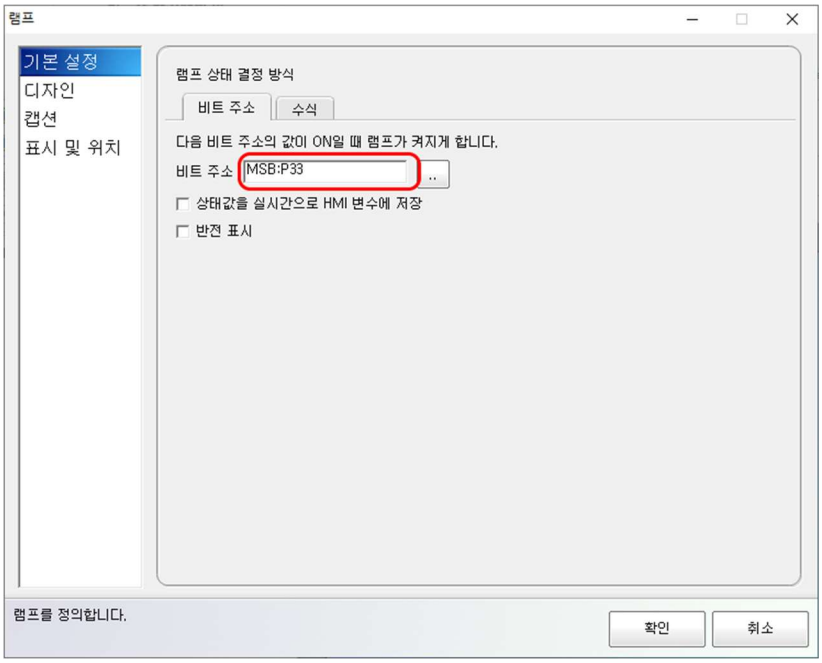


버튼 하나로는 심심하니까, 램프도 하나 그려보겠습니다.

화면 우측에서 램프를 선택하고 화면 빈공간에 클릭->드레그해서 램프를 그려보세요. 다음과 같이 표시됩니다.

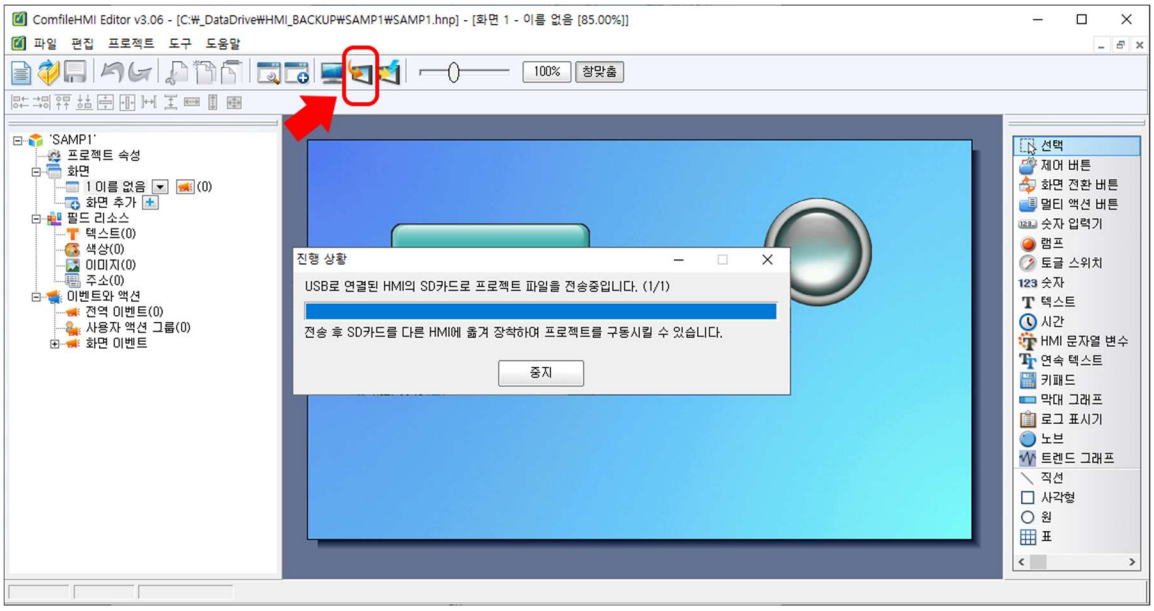


이제 램프를 더블클릭해서 램프를 P33 과 연결하세요. 방법은 버튼과 동일합니다.

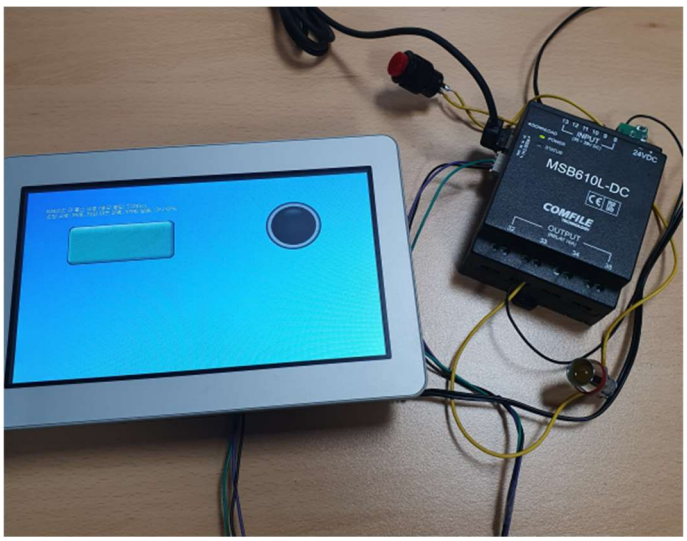


프로젝트 전송

이제 작화된 화면을 HMI 로 전송해보겠습니다. 화면 위 프로젝트전송 아이콘을 클릭하면 전송이 시작됩니다.

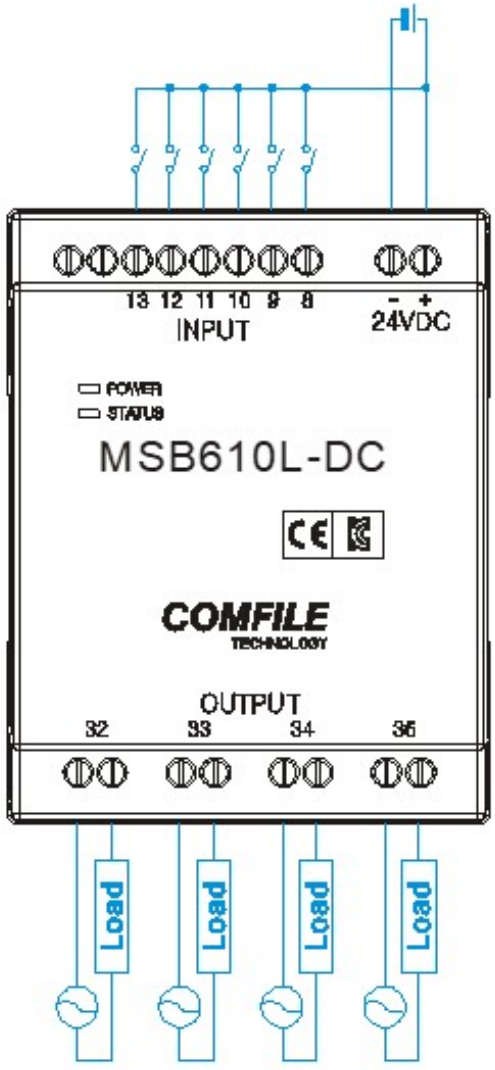


전송이 끝나면 곧바로 실행됩니다. 이제 HMI 의 버튼을 누르면 MSB 의 P33 이 켜졌다 꺼졌다 합니다. 동작확인을 위해서 P33 에 램프를 하나 연결해 두었습니다.



https://youtu.be/cCkDUhQM9vc <-- 동작 동영상

MSB610L-DC 입출력 연결방법



방향	포트번호	갯수	종류	설명
입력	8 ~ 13	6	24VDC	24V 입력이 있으면 1, 없으면 0 이 됩니다.
출력	32 ~ 35	4	릴레이 10A 접점	1 을 기입하면 ON, 0 을 기입하면 OFF

<끝>