MP STUDIO MANUAL

V2.1.5

MP STUDIO는 ㈜아이로직스에서 자체개발한 PLC(MPS와 MPA)에 LD(LADDER LOGIC)으로 프로그램할 수 있는 소프트웨어입니다.

MPS 제품군은 케이스 마감이 없는 PCB 형태이지만 딘레일에 체결이 가능하도록 딘레일 가이드 및 클립으로 구성되어 있는 제품으로서 자사에서 제공하는 MP STUDIO 소프트웨어로 LD(LADDER LOGIC)으로 프로그램 할 수 있는 PLC입니다.

MPA 제품군은 케이스 마감이 있는 형태이며 딘레일도 체결이 가능합니다. MPS 제품군보다는 다소 가격이 높지만 케이스가 마감이 있고 노이즈 제거 기능을 강화하였고 고성능 아날로그 입력(15비트), 아날로그 출력(16비트), 펄스 출력, PT100Ω 모듈을 추가하여 사용할 수 있습니다.

[제품명 구분 방법]

☞ 아날로그 입력 "X"는 최대 5개 모듈을 사용할 수 있습니다.
☞ 온도센서 입력 "F"는 최대 5개 모듈을 사용할 수 있습니다.
☞ 아날로그 입력 및 온도센서 입력은 총 합이 최대 5개 모듈을 사용할 수 있습니다.

☞ 아날로그 출력 "Y"는 최대 4개 모듈을 사용할 수 있습니다.
☞ 펄스출력 "K"는 최대 2개 모듈을 사용할 수 있습니다.
☞ 아날로그 출력 및 펄스출력은 총 합이 최대 12 포인트를 초과할 수 없습니다.

▣ 알림

본 사용설명서는 MPS, MPA 제품군을 프로그램하고, 디버깅할 수 있는 MP STUDIO를 사용하는 방법을 간략히 설명한 매뉴얼입니다.

저희가 취급하고 있는 제품군은 MPS, MPA와 MPINO, MPAINO가 있습니다.

	MPS, MPA 제품군	MPINO, MPAINO 제품군
프로그램 작성 툴	MP STUDIO	ARDUINO IDE
		MPINO STUDIO
프로그램 언어	LADDER LOGIC	ARDUINO IDE : ARDUINO C
		MPINO STUDIO : ARDUINO C + LADDER LOGIC

개발자의 도움이 되고자 블로그를 운영하고 있습니다.
링크: https://blog.naver.com/ilogics

▣ 주의사항

본 사용설명서는 MP STUDIO의 기능변경으로 인하여, 내용이 변경될 수 있습니다.

- 본 제품은 프로그램 다운로드 및 모니터링(디버깅)을 지원합니다. 반대로 제품으로부터 프로그램을 다시 PC로 업로드하는 기능을 지원하지 않습니다. 어떠한 방법으로도 업로드를 받을 수 없으니 프로그램의 최종버전을 잘 보관해주시기 바랍니다.

- 폐사의 제품을 사용하다 발생한 손해 및 손실에 대하여 폐사는 민,형사적 법적 책임이 없음을 명시합니다. 폐사의 제품을 사용하기 전에 충분한 안전검사 및 테스트를 마치고 인적/물적 사고가 발생할 수 있는 장비에는 2차 안전장치를 설치하여 사용해주시기 바랍니다.

- 사용 전에 안전을 위한 주의사항의 반드시 읽고 정확하게 사용하여 주십시오.
- 사용설명서를 읽고 난 뒤에는 제품을 사용하는 사람이 항상 볼 수 있는 곳에 잘 보관하십시오.

<제목 차례>

1. 기초 사용방법 8
1.1. 프로그램 및 다운로드 8
1.2. 단축키 9
1.3. 다운로드 케이블 9
2. FAQ 11
2.1. 프로그램 업로드를 지원하나요 ? 11
2.2. 안전성에 문제가 없나요 ? 11
2.3. 주로 어느 기업들이 사용하고 있나요? 11
2.4. 속도가 느리지 않나요 ? 11
2.5. LS산전 PLC와 호환이 되나요 ? 11
3. 메모리 12
3.1. 데이터 메모리 12
3.2. 사용자 정의 메모리 14
3.3. 특수 비트 메모리 16
4. MP STUDIO 17
4.1. 접점 생성 18
4.2. 펑션블럭 생성 20
4.3. 도구상자 21
4.3.1. 프로젝트 21
4.3.2. 보기 23
4.3.3. 검색 24
4.3.4. 실행 25
4.3.5. 설정 34
5. 접점 42
5.1. 접점 종류 42
5.2. N.O접점 및 N.C접점 43
5.2.1. N.O 접점 43
5.2.2. N.C 접점 43
5.3. 코일출력 44
5.3.1. 코일 44
5.3.2. SET코일 44
5.3.3. RESET코일 44
5.4. 반전접점 45
5.5. 상승/하강 검출접점 (양변환/음변환 검출접점) 46
5.5.1. 상승 검출접점 (양변환 검출접점) 46
5.5.2. 하강 검출접점 (음변환 검출접점) 47
5.6. 비교 접점 48

5.7. 자기유지 회로 ····· 50
5.8. 워드 메모리의 비트 접점 사용 ····· 51
6. 유용한 기능 ····· 52
6.1. ETHERNET 다운로드 및 모니터링 ····· 52
6.2. FORCE DATA (메모리에 값 쓰기) ····· 53
6.3. EEPROM 사용법 (제품의 초기설정값 적용법) ····· 54
7. 펑션 블럭 ····· 55
7.1. 비교 펑션블럭 ····· 63
7.2. 변환 펑션블럭 ····· 64
7.2.1. H2D (Hexadecimal To Decimal) ····· 65
7.2.2. D2H (Decimal To (Hexadecimal) ····· 66
7.2.3. INV (반전 명령어) ····· 67
7.2.4. NEG (부호 반전 변환) ····· 68
7.3. 수학 펑션블럭 ····· 69
7.3.1. ADD ····· 70
7.3.2. SUB ····· 71
7.3.3. MUL ····· 72
7.3.4. DIV ····· 73
7.3.5. INC ····· 74
7.3.6. DEC_I ····· 75
7.3.7. SCALE ····· 76
7.3.8. FLOOR ····· 77
7.4. 카운터 펑션블럭 ····· 78
7.4.1. CTU ····· 79
7.4.2. CTD ····· 81
7.4.3. CTUD ····· 83
7.4.4. HCNT ····· 85
7.4.5. HRST ····· 86
7.4.6. ENCODER ····· 87
7.4.7. ENMOV ····· 88
7.5. 타이머 펑션블럭 ····· 89
7.5.1. TON (On Delay Timer) ····· 90
7.5.2. TOFF (OFF Delay Timer) ····· 91
7.5.3. TPL (Pulse Timer) ····· 92
7.5.4. TMON (Monostable Timer) ····· 93
7.5.5. TMR (Accumulative Timer) ····· 94
7.6. 로직 컨트롤 펑션블럭 ····· 96
7.6.1. JMP ····· 97

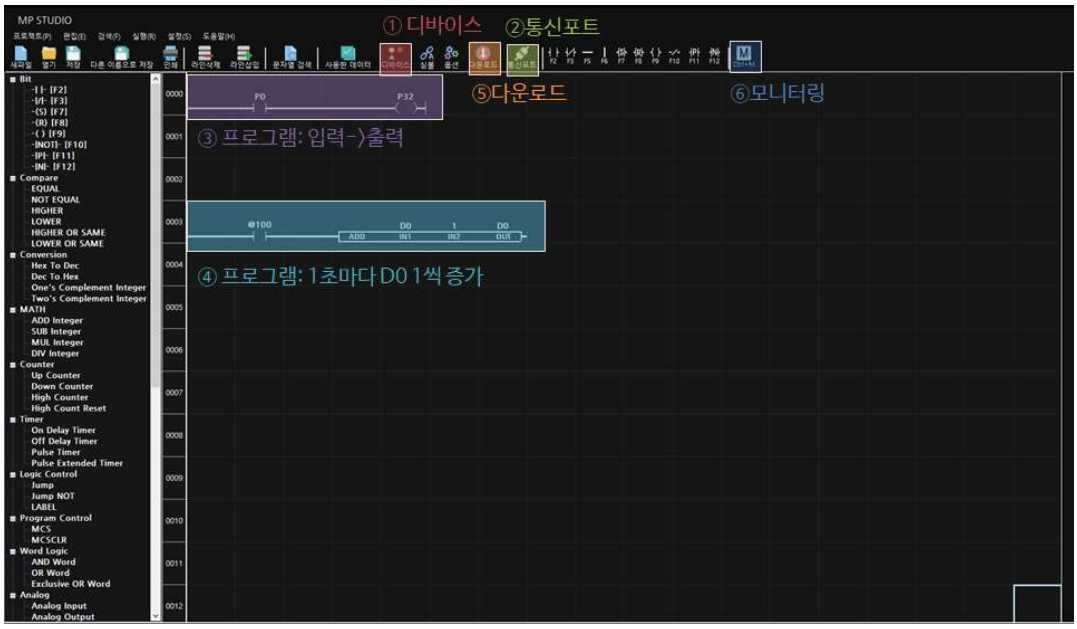
7.6.2. JMPN ····· 98
7.7. 데이터 컨트롤 펑션블럭 (Data Control) ····· 99
7.7.1. MAX ····· 100
7.7.2. MIN ····· 101
7.7.3. SUM ····· 102
7.7.4. AVE ····· 103
7.7.5. MUX ····· 104
7.8. 프로그램 컨트롤 펑션블럭 ····· 105
7.8.1. MCS ····· 106
7.8.2. MCSCLR ····· 107
7.9. 워드로직 펑션블럭 ····· 108
7.9.1. WAND ····· 109
7.9.2. WOR ····· 110
7.9.3. WXOR ····· 111
7.10. 아날로그 펑션블럭 ····· 112
7.10.1. ADC (아날로그 입력 : DC 0~5V, DC 0~10V, 0~20mA) ····· 113
7.10.2. ADC2 (아날로그 입력 : 4~20mA, DC 1~5V) ····· 114
7.10.3. DAC (아날로그 출력) ····· 115
7.10.4. NTEMP (NTC 써미스터 온도) ····· 116
7.10.5. PTEMP (PT100Ω 온도센서) ····· 117
7.10.6. PWM (펄스 출력) ····· 118
7.10.7. FPWM (주파수 펄스출력) ····· 119
7.10.8. FDPWM (주파수 및 듀티비 펄스출력) ····· 120
7.10.9. NPWM (제한된 펄스개수 출력) ····· 121
7.10.10. POFF (펄스출력 정지) ····· 123
7.10.11. PSR (펄스출력 상태) ····· 124
7.10.12. PID (Proportional-Integral-Differential) ····· 125
7.10.13. PID_PWM (Proportional-Integral-Differential for PWM) ····· 127
7.11. 이동 펑션블럭 (MOVE) ····· 129
7.11.1. MOV ····· 130
7.11.2. GMOV ····· 131
7.11.3. FMOV ····· 132
7.12. 워드로직 펑션블럭 ····· 133
7.12.1. SHL ····· 134
7.12.2. SHR ····· 135
7.13. 시리얼통신 펑션블럭 ····· 136
7.13.1. 시리얼 통신박식의 종류 ····· 136
7.13.2. 시리얼 통신의 설정 ····· 137

7.13.3. 시리얼 통신 모드 ··· 138
7.13.4. PUT (송신) ··· 139
7.13.5. GET (수신) ··· 140
7.13.6. PUTLEN (송신버퍼개수) ··· 141
7.13.7. GETLEN (수신버퍼개수) ··· 142
8. HMI 연결 / 외부기기 연결 ··· 143
8.1. MODBUS RTU ··· 143
8.1.1. MODBUS 설정 ··· 143
8.1.2. MODBUS RTU 패킷 구성 ··· 143
8.1.3. 펑션코드 ··· 144
8.1.4. CRC ··· 144
8.1.5. START ADDRESS ··· 144
8.1.6. HMI의 시작 어드레스 ··· 145
8.1.7. M2I의 HMI에서 사용할 경우 ··· 146
8.1.8. MODBUS RTU Protocol Packet ··· 147
8.2. COMFILE HMI ··· 153
8.2.1. MP STUDIO 설정 ··· 153
8.2.2. COMFILE HMI 설정 ··· 153
8.2.3. Example ··· 155
8.3. CNET (XBC/LINK) ··· 164
8.3.1. XBC ··· 164
8.3.2. LINK ··· 164
8.3.3. BCC 체크섬 사용 ··· 164
8.3.4. M2I를 사용할 경우 ··· 165
8.3.5. EasyView를 사용할 경우 ··· 177
9. I2C ··· 190
9.1. ISTA ··· 191
9.2. IWR ··· 192
9.3. IRD ··· 193
9.4. ISTP ··· 194
9.5. ISR ··· 195
9.6. I2C Example ··· 196
10. Library ··· 199
10.1. DS3231 (Real Time Clock) ··· 199
10.1.1. DS3231_WR (DS3231 Write) ··· 200
10.1.2. DS3231_RD (DS3231 Read) ··· 201
10.2. CLCD (Character LCD / PCF8574) ··· 202
10.2.1. CLCD 주소 ··· 202

10.2.2. CINI (CLCD Initialize) ····· 203
10.2.3. CLOC (CLCD Location) ····· 204
10.2.4. CSTR (CLCD String) ····· 205
10.2.5. CINT (CLCD Integer) ····· 206
10.2.6. CREAL (CLCD Flating Point) ····· 208
10.2.7. CHEX (CLCD Hexadecimal) ····· 209
10.2.8. CCLS (CLCD Clear) ····· 210
10.2.9. CCON (CLCD Cursor ON) ····· 211
10.2.10. CCOF (CLCD Cursor OFF) ····· 212
10.2.11. CLON (CLCD Light ON) ····· 213
10.2.12. CLOF (CLCD Light OFF) ····· 214
10.2.13. CPUT (CLCD PUT) ····· 215
10.2.14. CGET (CLCD GET) ····· 217
10.3. FND (7SEGMENT) ····· 218
10.3.1. FINI (FND Initialize) ····· 219
10.3.2. FINT (FND Integer) ····· 220
10.3.3. FDATA (FND DATA) ····· 222
10.3.4. FND 시계표시 ····· 224

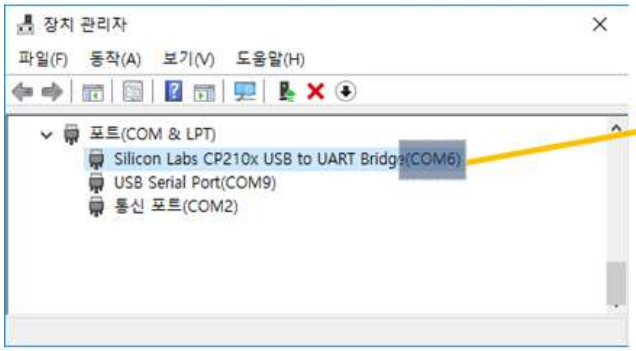
1. 기초 사용방법

1.1. 프로그램 및 다운로드



☞ "디바이스"를 실행하여 사용할 디바이스명을 선택합니다.
(MPS-16A16R 제품의 경우, MPS-16A16R(T)를 선택하시면 됩니다)

☞ "통신포트"를 실행하여 제품과 연결할 시리얼포트 번호를 설정합니다.



< 윈도우 장치관리자 >

< MP STUDIO 통신포트 설정>

☞ LD(Ladder Logic)언어 창에서 NO접점:단축키(F2)를 눌러 입력접점 생성 후, 생성된 접점에서 "ENTER" 키를 눌러 입력창에 디지털입력 포트번호 "P0"을 입력합니다. 이후, NO접점 우측으로 COIL접점:단축키(F9)를 눌러 출력접점 생성 후, 디지털출력 포트번호 "P32"을 입력합니다. (디지털 입력포트 P0의 입력이 있을 때, 디지털 출력포트인 P32를 ON시키는 프로그램입니다)

☞ NO접점:단축키(F2)를 눌러 입력접점 생성 후, 1초에 한번 ON되는 특수비트 "@100"을 입력합니다. 이후, NO접점 우측에서 펑션블럭을 생성하기 위해 왼쪽에 위치한 펑션블럭 툴박스에서 Math에 위치한 "ADD : ADD Integer"를 더블클릭하거나 키보드의 " ["키를

누르고 "ADD"를 입력하여 펑션블럭을 생성 한후, IN1과 IN2에 각각 "D0", "1"을 입력하고 OUT에 "D0"를 입력합니다. (1초마다 D0의 값이 1씩 증가시키는 프로그램입니다)

☞ 도구에서 "다운로드"를 실행하거나 키보드 단축키로 "Ctrl + T"를 실행하면 작성한 LD(Ladder Logic) 프로그램이 제품에 다운로드 됩니다.

☞ 도구에서 "모니터링"을 실행하거나 키보드 단축키로 "Ctrl + M"을 시행하면 제품의 입/출력 및 메모리의 상태를 모니터링(디버깅)할 수 있습니다.

1.2. 단축키

☞ MP STUDIO를 보다 편리하게 사용하기 위해 아래와 같이 단축키를 제공하고 있습니다.

단축키	설 명	단축키	설명
Ctrl + T	프로그램 다운로드	Ctrl + Z	실행취소
Ctrl + M	모니터링	Ctrl + Y	재실행
Ctrl + C	복사	Ctrl + O	프로젝트 열기
Ctrl + V	붙여넣기	Ctrl + S	프로젝트 저장
Ctrl + X	잘라내기	Ctrl + Alt + S	다른이름으로 저장
Ctrl + D	라인 삭제	Ctrl + F	문자열 검색
Ctrl + L	라인 삽입	Ctrl + G	줄 이동
[	펑션블럭 추가	Ctrl + P	인쇄
Ctrl + Enter		Ctrl + Ins	셀 삽입
Alt + Enter	메모리에 값 쓰기	Ctrl + Del	셀 삭제
ENTER	접점명, 메모리 입력	Shift + Enter	빠른 심볼등록

1.3. 다운로드 케이블

☞ 현재 대부분의 다운로드 케이블은 "USB-A To USB –B" 타입 컨넥터를 사용하고 있습니다. 프린터에 연결하는 USB 케이블과 같은 케이블입니다.

☞ 제품에서 다운로드 케이블을 연결했다가 끊었다가를 반복하면 OS에서 해당 포트를 닫아버리는 경우가 종종 발생하여 이러한 문제를 해결하기 위하여 USB 케이블에 USB To TTL 기능을 가진 케이블을 사용하도록 "MP Download Cable"을 제작하여 별도로 판매를 진행하고 있으며, MPS-8A8R(T)-S 및 MPS-16A16R(T) 신제품부터 적용하고 있습니다. 추후, 전 제품의 다운로드 케이블을 "MP Download cable"로 사용할 예정입니다.

제품명	다운로드 케이블	제품명	다운로드 케이블
MPS-8A8R	쇼핑몰 상품명 "Download cable" (USB A To USB B 타입)	MPS-8A4R(T)-S	쇼핑몰 상품명 "MP Download cable"
MPS-16A8R		MPS-8A8R(T)-S	
MPS-16A8R8T		MPS-16A16R(T)	
MPA 제품군			

("Download cable"은 추후 "MP Download cable"로 변경될 수 있습니다)

< Download Cable >

< MP Download Cable >

2. FAQ

2.1. 프로그램 업로드를 지원하나요 ?

▢ 프로그램 업로드를 지원하지 않습니다. 또한, 기본적으로 프로그램 잠금기능을 사용하고 있습니다. 그래서, 개발하신 제품은 다른 누구도 복제할 수 없는 장점이 있지만 소스 파일을 잃어버리면 개발한 프로그램을 복구할 수 없다는 단점을 가지고 있습니다. 때문에, 소스 파일 보관에 신경 써주시기 바랍니다.

2.2. 안전성에 문제가 없나요 ?

▢ 네, 많이 알려지지 않은 회사 및 제품이지만 2014년부터 꾸준히 연구/개발하고 있습니다.

▢ 소프트웨어적으로 창립초기에는 버그가 다소 있었지만, 현재에는 버그를 발견하기 어려울 정도로 안정화되어 있고 최근 2년 이상 버그로 인한 문제가 발생한 적이 없습니다.

▢ 하드웨어적으로 디지털입력 및 출력은 기술적으로 어려운 것이 없기 때문에 대기업 제품들과 큰 차이가 없습니다.

▢ 2019년 07월 삼성반도체의 RF장비를 MPS-16A8R8T 제품으로 자동제어를 위탁개발하였습니다. 현재(2022년 08월)까지 제품의 문제점 및 소손이 발생한적이 없고 지속적 판매를 진행중입니다.

▢ 2019년 07월 굴착기, 포크레인의 전기로 MPS-16A8R8T 제품에 전기를 공급하여 동작되는 자동화 시스템을 위탁개발하였습니다. 현재(2023년 01월)까지 제품의 문제점 및 소손이 발생한적이 없고 지속적 판매를 진행중입니다.

2.3. 주로 어느 기업들이 사용하고 있나요?

▢ 산업 범용 컨트롤러로서 다양한 업종에서 사용하고 있습니다. 최근에 주로 사용되고 있는 업종은 반도체(삼성반도체, 서울반도체 등) 업종입니다.

2.4. 속도가 느리지 않나요 ?

▢ 일반적은 자동제어를 수행하기에 절대 느리지 않습니다. 저희가 위탁개발을 진행한 프로젝트에서는 스캔타임이 4ms 이상을 초과한적이 없습니다. us, ns 단위의 제어를 수행하고자 한다면 PLC 보다는 MICOM으로 개발하길 권장드립니다.

2.5. LS산전 PLC와 호환이 되나요 ?

▢ LS산전 PLC에 프로그램하는 XG5000 소프트웨어는 LS산전에 저작권이 있습니다. 때문에, XG5000 소프트웨어에서 자사의 PLC 제품에 프로그램을 다운로드할 수는 없습니다.

▢ LS산전의 XGK에 사용하는 LD(Ladder Logic)언어와 비슷하기 때문에, 자사의 소프트웨어인 MP STUDIO를 사용하여 프로그램을 옮기는 작업을 진행하여 대체는 대부분 가능합니다.

▢ 양 변환 및 음 변환 검출 접점, 데이터 메모리의 표현방법이 조금 차이점이 있습니다. 그 외 특수한 펑션블럭의 차이가 있습니다.

3. 메모리

3.1. 데이터 메모리

▢ 데이터 메모리의 영역은 P, M, D, C, T로 구분하여 구성되어 있습니다.

메모리 타입	설 명
P	PORT Memory (1 BIT) MPS, MPA 제품군의 디지털 입력 상태값 확인 및 디지털 출력를 제어합니다. 대부분 디지털입력은 P0~P31, 디지털출력은 P32~P63으로 구성되어 있습니다.
M	BIT Type Memory (1 BIT) M0 ~ M9, M10, M11...로 10진수 포인트명으로 사용합니다.
D	WORD Type Memory (16 BIT) D0 ~ D9, D10, D11...로 10진수 포인트명으로 사용합니다.
C	Counter Type Memory (16 BIT) 카운터 메모리로서, C0 ~ C9, D10, C11...로 10진수 포인트명으로 사용합니다.
T	Timer Type Memory (16 BIT) 타이머 메모리로서, T0 ~ T9, T10, T11...로 10진수 포인트명으로 사용합니다.
R	Real(Floating Point) Type Memory (32 BIT) 실수 메모리로서, R0 ~ R9, R10, R11...로 10진수 포인트명으로 사용합니다.

▢ C 메모리와 T 메모리는 각각 카운터 펑션블럭과 타이머 펑션블럭에 사용하는 메모리이지만, D 메모리처럼 일반적인 WORD 메모리로 사용해도 무관합니다.

▢ P, M, D, C, T 메모리는 "B" 접두어로 BYTE 단위로 사용할 수 있습니다.
Ex) BP0 (P0~P7), BM1 (M8~M15), BD0 (D0의 Low Byte), BC1 (C0의 High Byte)
▢ P, M 메모리는 "W" 접두어로 WORD 단위로 사용할 수 있습니다.
Ex) WP0 (P0~P15), WM1 (P16~P31)

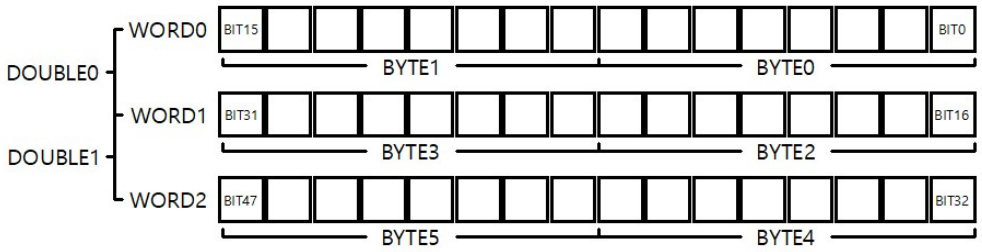
▢ P, M, D, C, T 메모리는"D" 접두어로 DOUBLE WORD 단위로 사용할 수 있습니다.
Ex) DM0 (M0~M31), DM1 (P32 ~ P63), DD2 (D2(Low), D3(High)), DC31 (C31,C32)

▢ D 메모리는 ".(BIT)" 접미어로 BIT 단위로 사용할 수 있습니다.
Ex) D0.0 (D0의 최하위 BIT), D0.15 (D0의 최상위 BIT), D1.0 (D1의 최하위 BIT)
Ex) D 메모리는 16개의 BIT로 구성되므로 D0.16과 같이 15를 초과하여 사용할 수 없습니다.

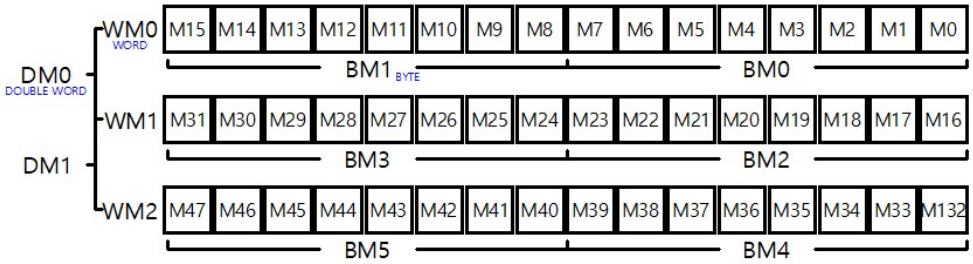
▢ R 메모리는 32비트 부동 소수점 IEEE 754 표준방식을 사용합니다. 부동 소수점은 "부동 소수점의 오류"라는 용어로 부동 소수점의 사칙연산 및 표현에서 원치 않은 오차율이 발생할 수 있습니다. 예를 들어 7.1이 7.10001이 될 수 있습니다. 이러한 문제를 인지하고 사용해 주시기 바랍니다.

▢ BIT, BYTE, WORD, DOUBLE WORD의 표현

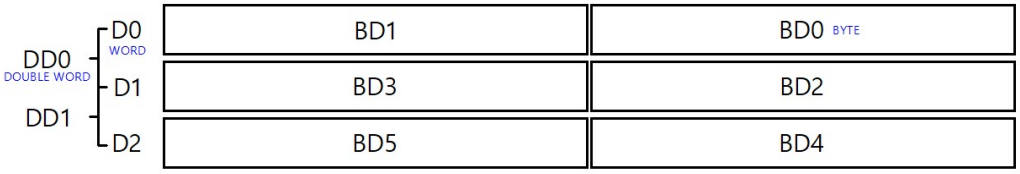
- BIT는 0 또는 1을 표현할 수 있는 메모리의 최소 단위입니다.
- BYTE는 -128 ~ +127를 표현할 수 있으며, 8개의 BIT로 구성됩니다.
- WORD는 -32,768 ~ + 32,767를 표현할 수 있으며, 2개의 BYTE로 구성됩니다.
- DOUBLE WORD는 -2,147,483,648 ~ +2,147,483,647를 표현할 수 있으며, 2개의 WORD로 구성됩니다.



- M 메모리

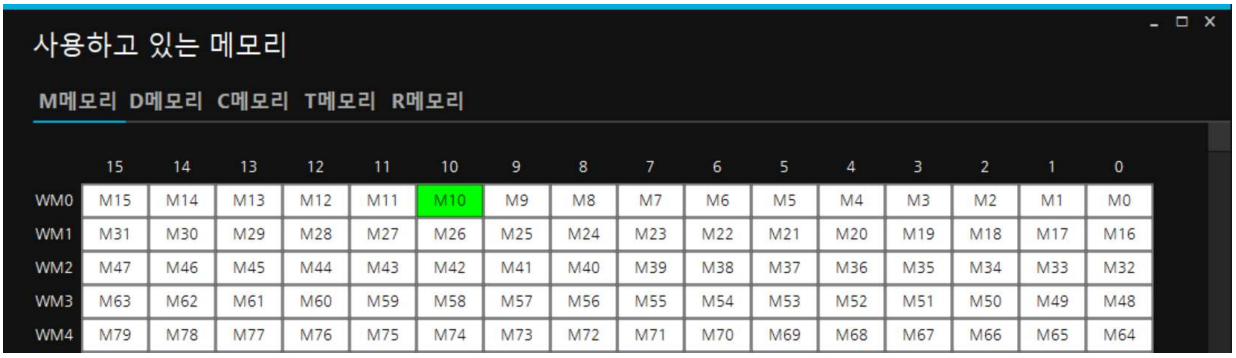


- D, C, T 메모리 (아래는 D메모리를 기준으로 표시하였습니다)



▢ LD(Ladder Logic) 작성창에서 사용한 데이터 메모리는 단축아이콘의 "사용한 데이터"에서 확인할 수 있습니다.

- LD(Ladder Logic)언어 창에서 M10을 사용하고 있다면 아래와 같이 표기됩니다.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WM0	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
WM1	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16
WM2	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32
WM3	M63	M62	M61	M60	M59	M58	M57	M56	M55	M54	M53	M52	M51	M50	M49	M48
WM4	M79	M78	M77	M76	M75	M74	M73	M72	M71	M70	M69	M68	M67	M66	M65	M64

3.2. 사용자 정의 메모리

▢ 메모리 이름을 사용자가 정의할 수 있는 메모리입니다.
별도의 창에서 정의하지 않고 LD(Ladder Logic) 창에서 #, ##, ###을 접두어로 하는 메모리 이름을 작명하여 사용하실 수 있습니다.

▢ 데이터 타입은 아래와 같이 접두어에 의해 정의됩니다.

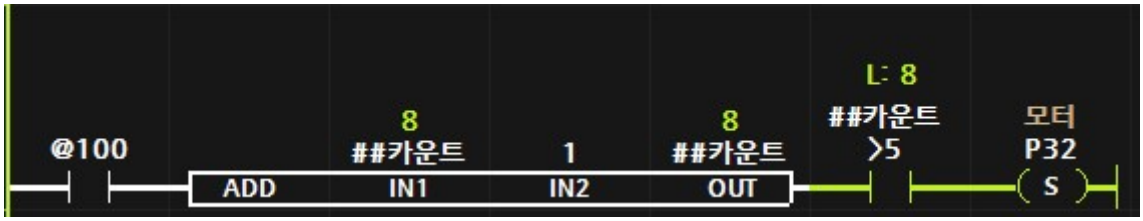
- # : BIT (비트)
- ## : WORD (워드)
- ### : DOUBLE WORD (더블워드)

▢ Ex) 비트 메모리 사용 (#이름)

- 시작버튼:P0가 ON되면, 사용자 비트 메모리:#Motor가 ON됩니다. 이후, 시작버튼:P0가 OFF되어도 사용자 비트 메모리:#Motor는 자기유지회로에 의해 ON이 유지됩니다. 또한 사용자 비트 메모리:#Motor가 ON되는동안 모터:P32가 ON됩니다.
- 정지버튼:P1이 ON되면, 사용자 비트 메모리:#Motor와 모터:P32는 OFF됩니다.

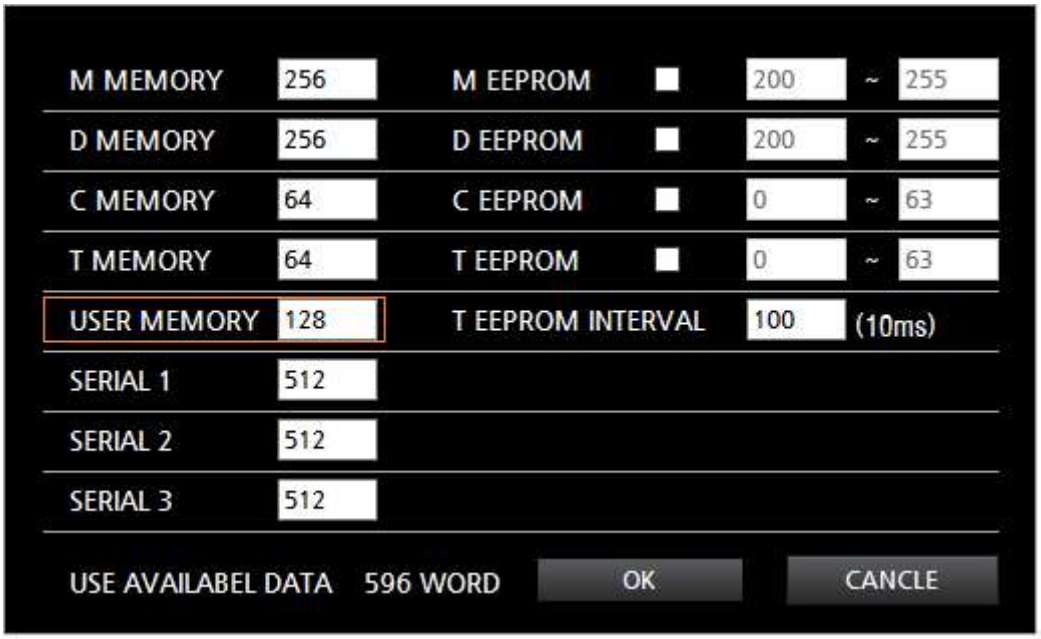
- 시작버튼:P0가 ON될때마다 카운터:D0 메모리가 1씩 증가됩니다. 카운터:D0 메모리가 10을 초과하면 모터:P32가 ON됩니다.

▢ Ex) 워드 메모리 사용 (##이름)



- 1초에 한번씩 사용자 워드 메모리인 "##카운터"를 1씩 증가시킵니다. ##카운터가 5를 초과하면, 디지털출력포트인 P32:모터가 ON되어 모터가 동작하는 예입니다.

▢ 사용자 정의 메모리는 데이터 메모리 설정창에서 설정한 "USER MEMORY" 크기만큼 사용할 수 있습니다. (단위는 WORD입니다)

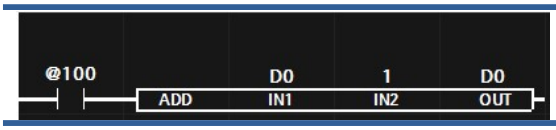


- 사용자 정의 메모리는 최소 10 WORD 이상을 사용해야 합니다.
- 사용자 정의 메모리를 시스템에서 최소 3 WORD를 사용합니다.
- MRF 제품군 또는 IBUS를 사용할 경우, 1개의 슬레이브 어드레스마다 최대 44 WORD 사용자 정의 메모리를 사용합니다. 슬레이브 제품이 2개일 경우에는 최대 88 WORD를 사용합니다. 사용하는 메모리의 양은 사용하는 #P, ##X, ##Y에 따라 달라지게 됩니다.

3.3. 특수 비트 메모리

▢ 특수접점 메모리는 N.O접점 또는 N.C접점에서만 사용할 수 있습니다.

- **@BEGIN** : 전원이 투입된 이후 첫 스캔(한번)만 On됩니다.
- **@ON, TRUE :** 항상 ON 상태의 특수 메모리입니다.
- **@OFF, FALSE** : 항상 OFF 상태의 특수 메모리입니다.
- **@(n)** : n의 시간의 간격으로 1스캔동안 ON됩니다. (n은 10ms 단위)
  @(n)시간은 **최대 8개의 종류**를 사용할 수 있습니다.

	1초마다 ADD 펑션블럭이 실행되어 D0의 값이 1씩 증가합니다.

- **@F(n)** : n의 시간의 간격으로 ON/OFF를 반복합니다. (n은 10ms 단위)
  @F(n)시간은 **최대 8개의 종류**를 사용할 수 있습니다.

	2초동안 P32코일이 ON, 2초동안 P32코일이 OFF 됨을 반복합니다.

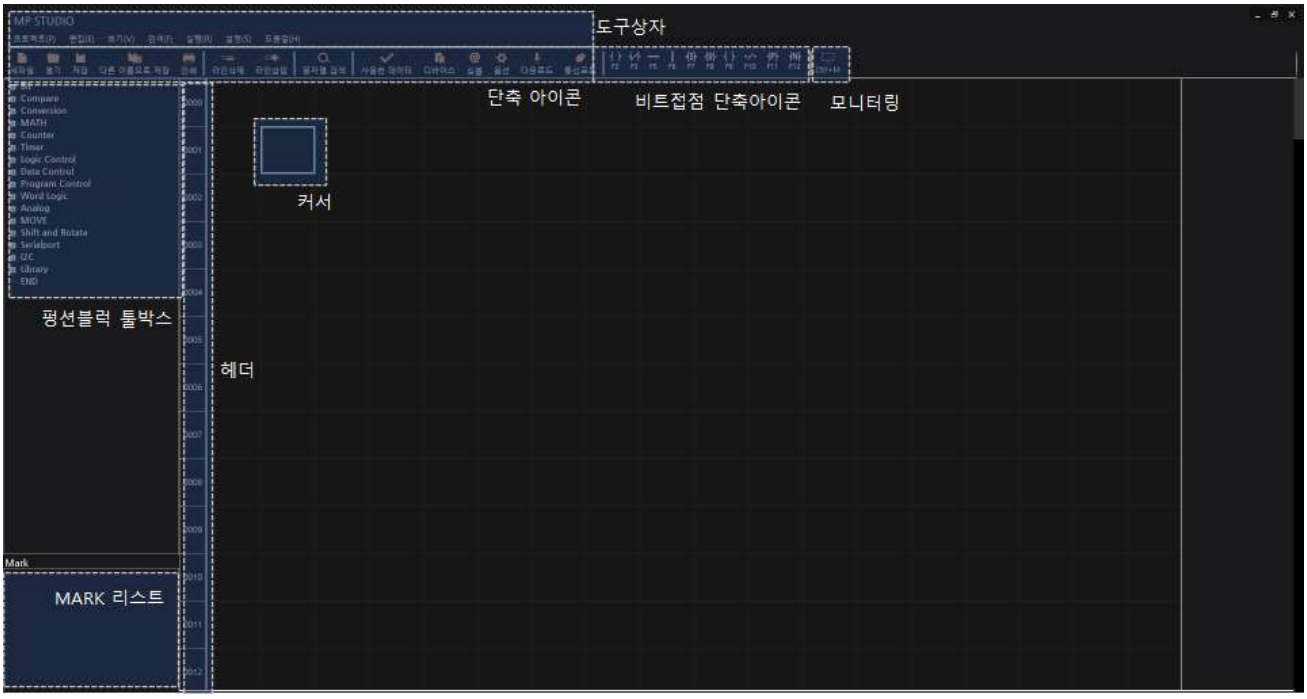
- **@PWM(n)** : 고속펄스포트에 펄스가 출력되고 있는 동안에만 ON이 됩니다.

4. MP STUDIO

▢ MP는 **M**EGA **P**LC의 약자입니다.

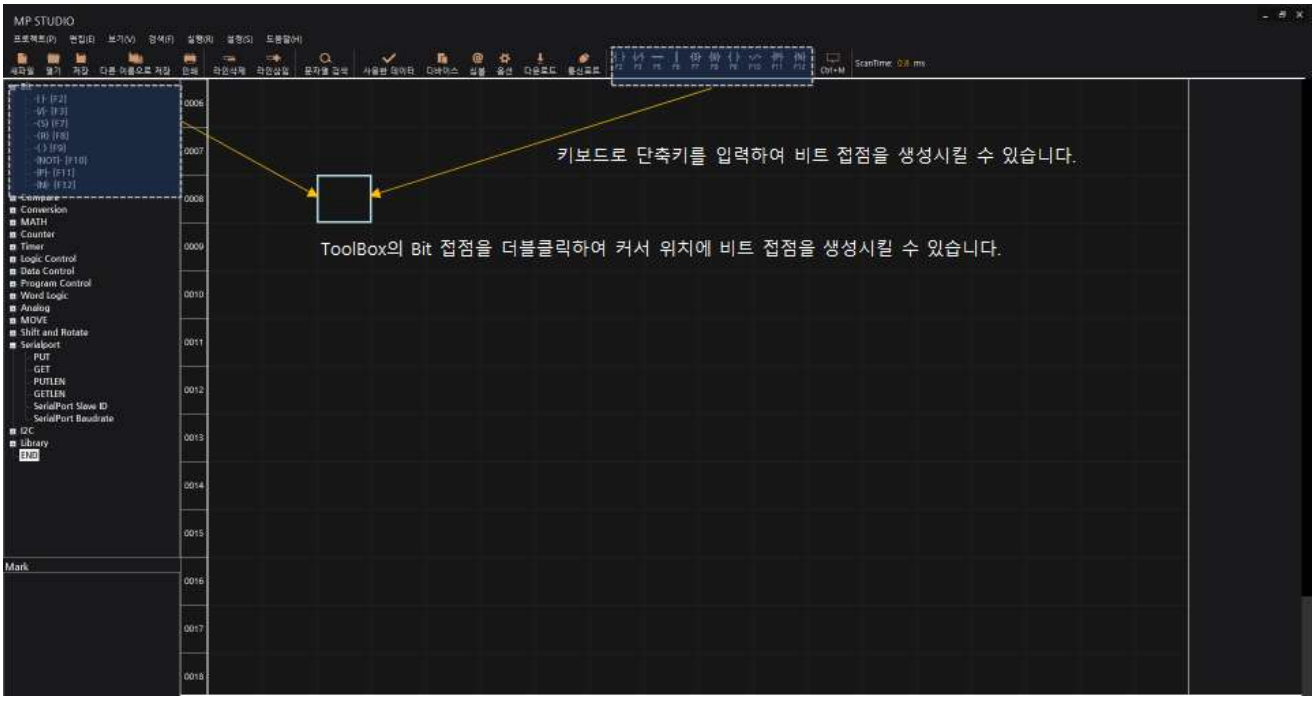
▢ LD(Ladder Logic)언어를 사용하여 프로그램하고 MPS, MPA 제품군에 프로그램을 다운로드(삽입)할 수 있는 소프트웨어입니다.

▢ 프로그램을 PC로 불러오는 업로드는 지원하지 않습니다. MPS, MPA 제품군에 다운로드 된 프로그램은 저희 ㈜아이로직스에서도 업로드를 할 수 없습니다. 장점으로는 타 업체에서 프로그램 복사 등을 할 수 없습니다. 단점으로는 작성한 프로그램 소스파일이 없으면 프로그램 재개발을 해야 하므로 프로그램 소스파일을 잘 보관해주시길 바랍니다.



4.1. 접점 생성

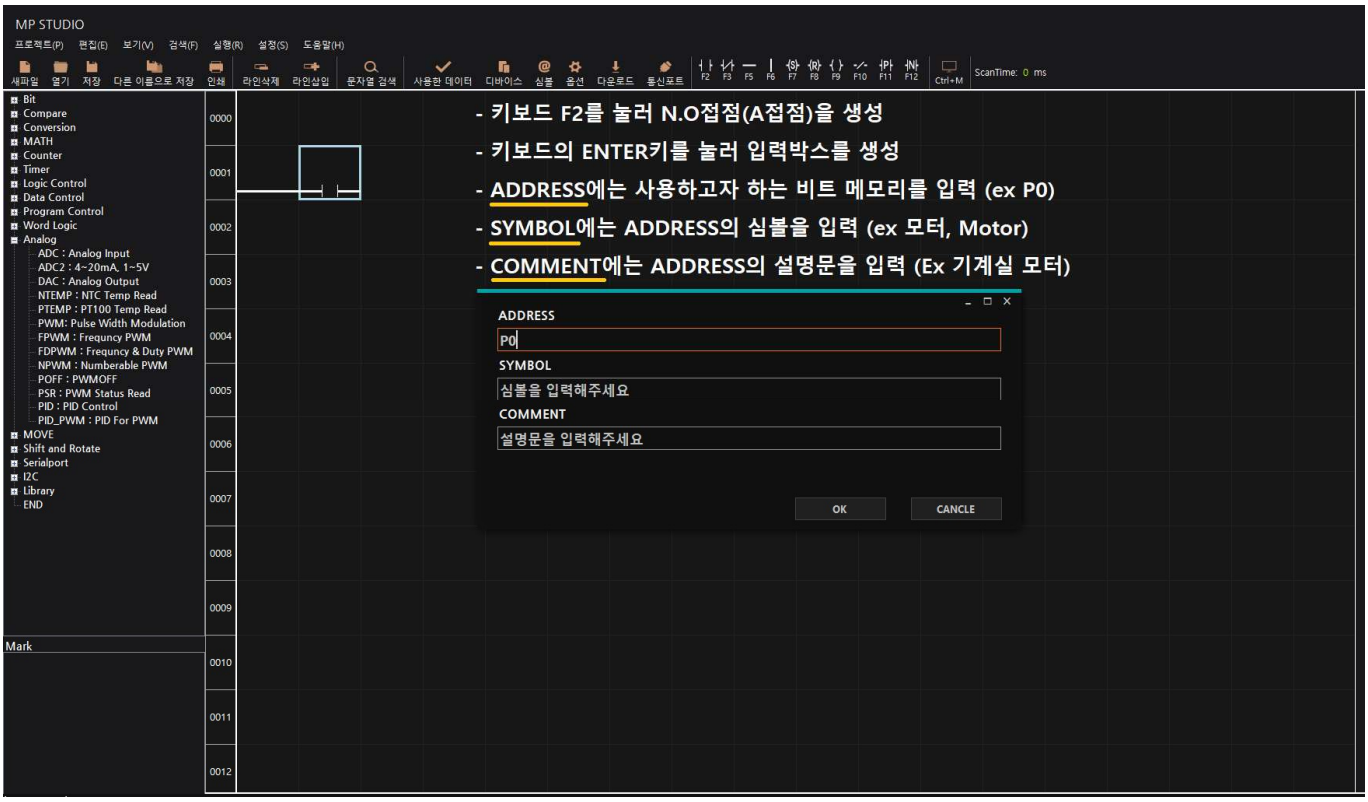
▢ 비트 접점은 MP STUDIO의 왼쪽에 위치한 "ToolBox"에서 더블클릭하여 생성시킬 수 있습니다. 또는, 키보드로 단축키를 입력하여 비트 접점을 생성시킬 수 있습니다.



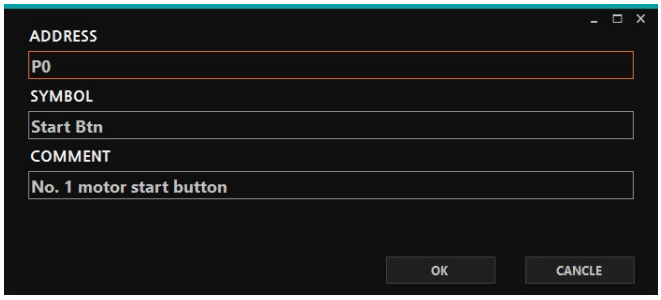
▢ 비트접점 단축키

아이콘	이름	설 명	단축키
┤ ├	N.O 접점 (A접점) (Normal Open)	평상시에는 끊어져있다가, 접점값이 "HIGH" 일때, 연결이 되는 접점	F2
┤/├	N.C 접점 (B접점) (Normal Close)	평상시에는 연결되어있다가. 접점값이 "HIGH" 일때, 연결이 끊어지는 접점	F3
──	가로 연결선	가로 방향으로 접점을 연결해주는 선	F5
|	세로 연결선	세로 방향으로 접점을 연결해주는 선	F6
-(S)-	SET COIL	입력조건이 성립이 되면, "HIGH"를 출력	F7
-(R)-	RESET COIL	입력조건이 성립이 되면, "LOW"를 출력	F8
-( )-	COIL	입력조건이 되면, "HIGH"를 출력하고, 조건이 되지 않으면, "LOW"를 출력하는 코일	F9
-/-	NOT 접점	입력조건이 "LOW"일 경우, 연결이 된다. 입력조건이 "HIGH"일 경우, 연결이 끊긴다.	F10
┤P├	상승 엣지 접점 (양 변환 검출접점)	입력조건이 "LOW"에서 "HIGH"로 변할 때, 한 스캔동안만 연결되고 끊기는 접점	F11
┤N├	하강 엣지 접점 (음 변환 검출접점)	입력조건이 "HIGH"에서 "LOW"로 변할 때, 한 스캔동안만 연결되고 끊기는 접점	F12

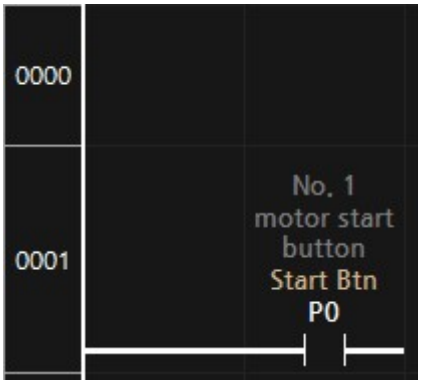
▢ 접점명은 접점에서 키보드의 "ENTER"키를 입력하여 생성된 입력박스에 접점명을 입력합니다.



▢ 접점명을 심볼로 등록을 했다면, 심볼을 입력할 수도 있습니다.



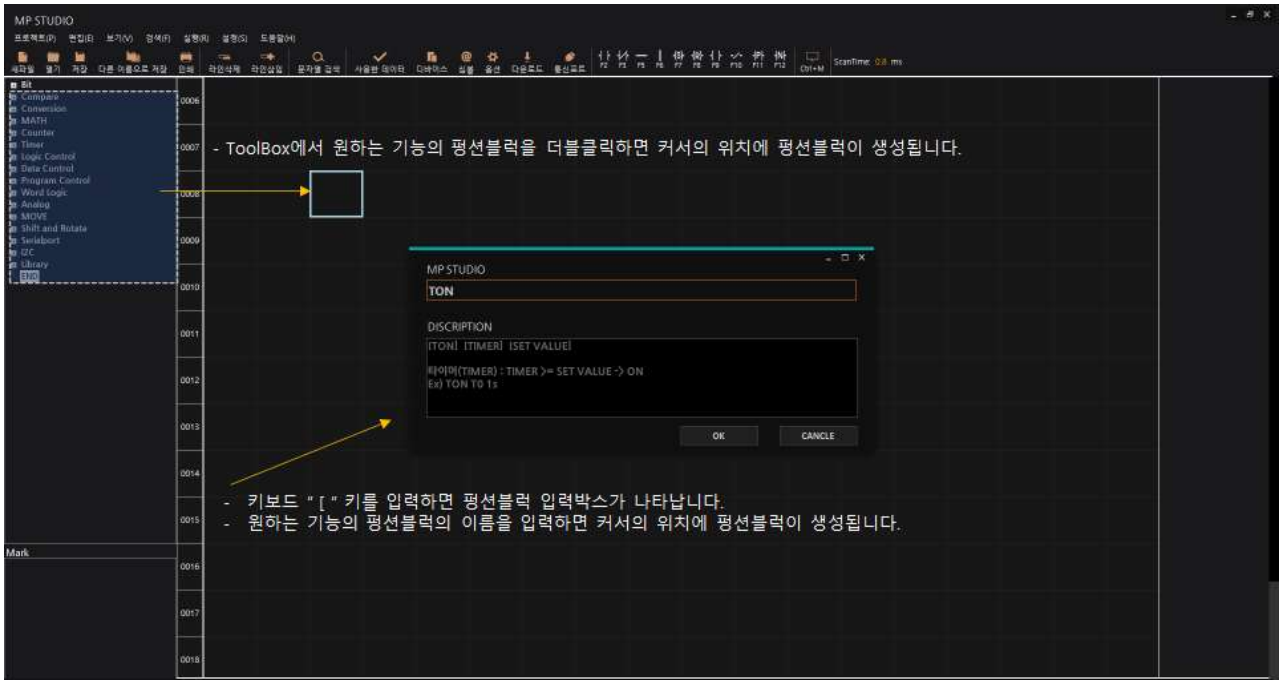
< 입력박스에 P0의 심볼 "Start Btn" 입력 >



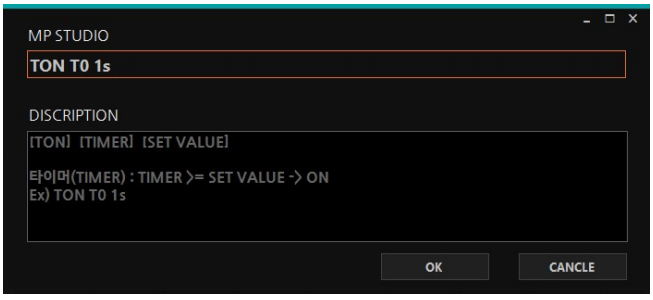
< 접점명 P0 및 심볼 "Start Btn" >

4.2. 펑션블럭 생성

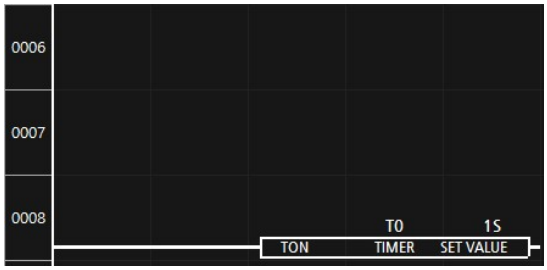
▢ 펑션블럭은 MP STUDIO의 왼쪽에 위치한 "ToolBox"에서 더블클릭하여 생성시킬 수 있습니다. 또는, 키보드의 " [" 키 또는 "Ctrl + Enter"로 펑션블럭을 생성시킬 수 있습니다.



▢ 펑션블럭 입력박스에 펑션블럭 이름과 파라미터를 모두 입력하면 펑션블럭이 생성될 때 입력한 파라미터가 입력되어 생성됩니다.



< 펑션블럭 입력박스에 TON T0 1S 입력 >



< TON 펑션블럭 생성 >

4.3. 도구상자

4.3.1. 프로젝트

▢ 새프로젝트
- 새로운 프로젝트를 생성합니다.

▢ 프로젝트 열기
- 프로그램이 저장된 확장자가 mpl인 파일에 저장되어 있는 프로그램을 불러옵니다.

▢ 프로젝트 저장
- 현재 열고 있는 저장파일에 작성한 프로그램을 저장합니다.

▢ 다른이름으로 저장
- 작성한 프로그램을 다른이름으로 저장합니다.

▢ 인쇄
- 작성한 프로그램을 인쇄합니다.

▢ 페이지 설정
- 인쇄할 페이지의 세부설정을 합니다.

▢ 인쇄 미리보기
- 인쇄할 페이지를 미리볼 수 있습니다.

▢ 최근 열어본 프로젝트
- 최근에 열어본 프로젝트를 열기할 수 있습니다. 최근에 열어본 프로젝트가 자동으로 리스트에 최상위부터 추가가 되며, 최대 5개의 프로젝트가 표시됩니다. 5개의 프로젝트를 초과하면 리스트에서 최하위의 프로젝트가 삭제되고 리스트에서 최상위에 새로 열기 한 프로젝트가 추가됩니다.

4.3.1.1.1. 편집

▢ 실행취소

- LD(Ladder Logic)언어 창에서 마지막으로 수정한 동작을 되돌리기 합니다.

▢ 재실행

- 실행취소하여 되돌리기한 동작을 다시 실행합니다. (실행취소를 실행취소 시킵니다)

▢ 잘라내기

- LD(Ladder Logic)언어 창에서 선택한 그리드를 잘라내기 합니다. (복사 + 삭제)

▢ 복사

- LD(Ladder Logic)언어 창에서 선택한 그리드를 클립보드로 복사합니다.

▢ 붙여넣기

- LD(Ladder Logic)언어 창에서 그리드 위치에 클립보드에 복사된 프로그램 내용을 붙여넣기 합니다.

▢ 삭제

- LD(Ladder Logic)언어 창에서 선택한 그리를 삭제합니다.

▢ 라인 삭제

- LD(Ladder Logic)언어 창에서 그리드 위치의 한 개의 가로(Row) 줄을 삭제합니다.

▢ 라인 삽입

- LD(Ladder Logic)언어 창에서 그리드 위치에 한 개의 가로(Row) 줄을 삽입합니다.

▢ 셀 삽입

- LD(Ladder Logic)언어 창에서 선택한 그리드에 한 개의 셀을 삽입합니다.

▢ 셀 삭제

- LD(Ladder Logic)언어 창에서 선택한 그리드에 한 개의 셀을 삭제합니다.

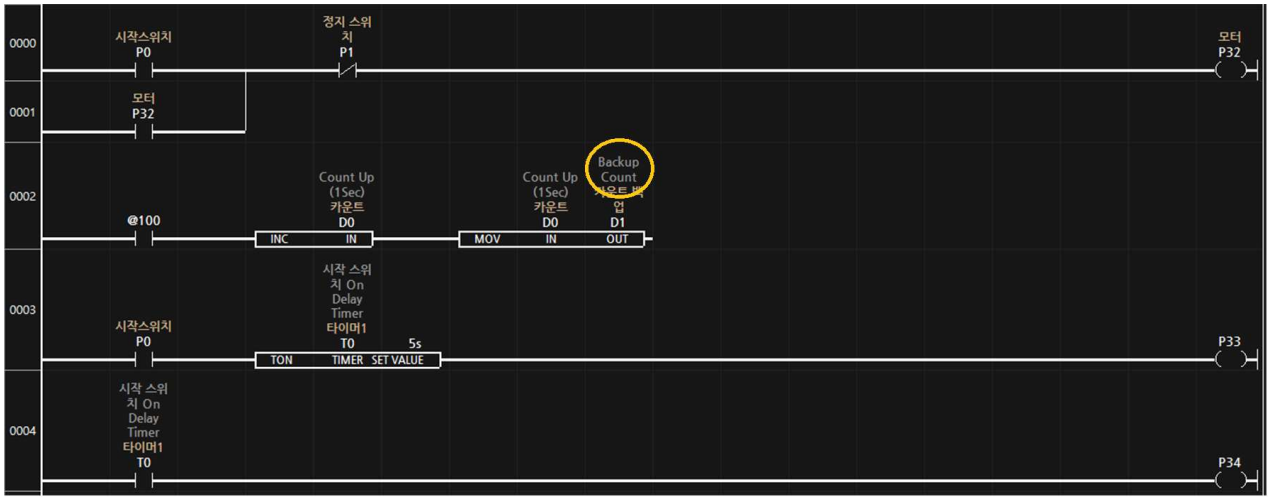
4.3.2. 보기

▢ 심볼

- 하기와 같이 데이터 메모리의 심볼을 표시할지를 설정할 수 있습니다.

▢ 설명문(심볼)

- 하기와 같이 데이터 메모리의 설명문을 표시할지를 설정할 수 있습니다.



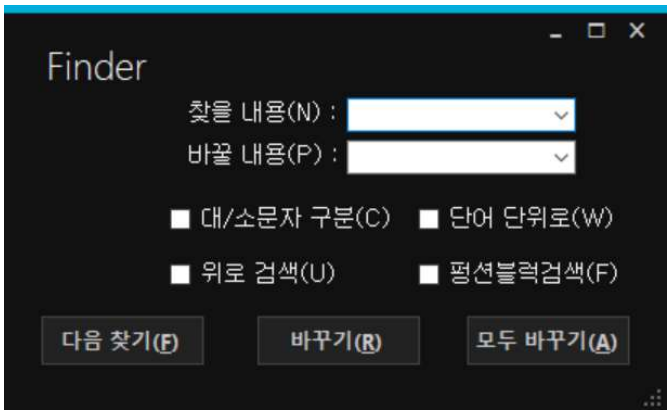
▢ Mark

- MP STUDIO의 왼쪽 툴박스 하단에 있는 Mark 리스트 창을 보이게/안보이게 설정할 수 있습니다.

4.3.3. 검색

▢ 문자열 검색

- LD(Ladder Logic)언어 창에서 작성한 문자를 검색합니다.



- 검색할 문자열을 "찾을 내용"의 박스에 입력하여 "다음 찾기"를 실행하면 문자를 검색하여 검색된 위 치로 화면이 이동됩니다.

- "대/소문자 구분" 항목을 체크하면 검색할 문자열의 대문자, 소문자를 구분하여 검색하게 됩니다.

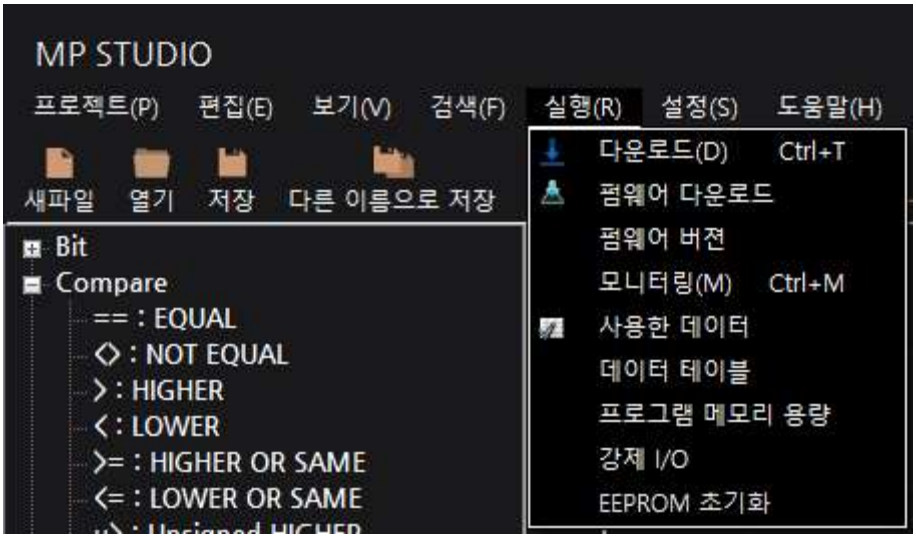
- "단어 단위로" 항목을 체크하지 않을 경우 "Motor"를 검색하면 LD(Ladder Logic)언어 창에서 "Motor2"일 경우도 검색을 하지만, 체크할 경우 "Motor"인 경우만 검색하고 "Motor2"의 경우는 검색하지 않게 됩니다.

- "찾을 내용"과 "바꿀 내용"에 각각 문자열을 입력하고 "바꾸기"를 실행하면 LD(Ladder Logic)언어 창에서 문자열을 검색하면서 문자열을 하나씩 변경합니다. "모두 바꾸기"를 실행하면 LD(Ladder Logic)언처 창에서 검색된 모든 문자열을 변경합니다.

- "위로 검색" 옵션은 문자열을 검색할 때 아래 줄에서 위쪽으로 검색하겠다는 것입니다. 체크를 하지 않을 경우, 위 줄에서 아래쪽으로 검색하게 됩니다.

- "펑션블럭검색"은 펑션블럭명 "ADD", "MOV" 으로 검색하여 사용하고 있는 펑션블럭을 검색할 수 있습니다.

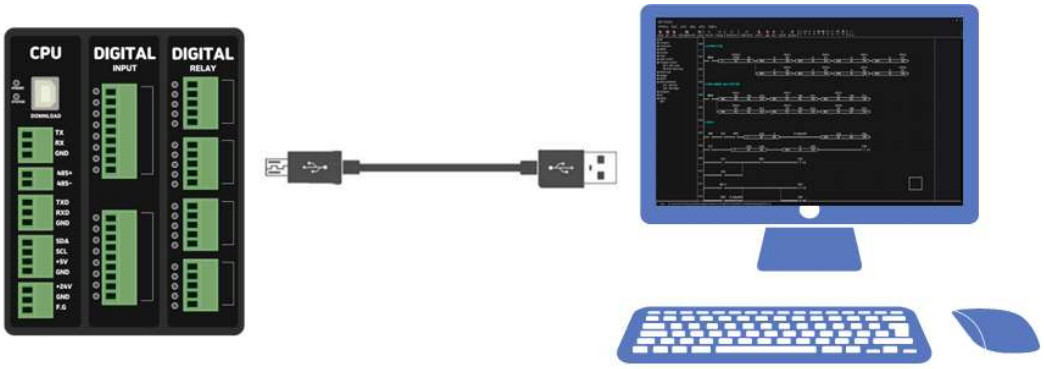
4.3.4. 실행



4.3.4.1. 프로그램 다운로드

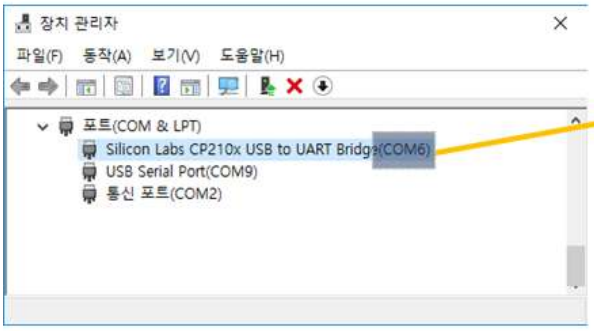
- LD(Ladder Logic)언어 창에서 작성한 프로그램을 MPS, MPA 제품군으로 다운로드 합니다.

- PC와 MPS, MPA 제품군은 다운로드 케이블로 연결되 있어여 합니다.



< MPS, MPA 제품군 > < 컴퓨터 >

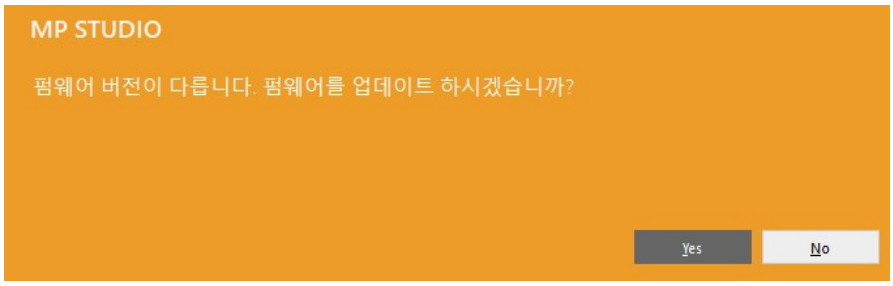
- 윈도우의 장치관리자에서 다운로드 케이블에 부여된 COM 포트 번호를 MP STUDIO의 통신포트 설정의 다운로드 COM 포트 번호로 설정해야 합니다.



< 윈도우 장치관리자 >

< MP STUDIO 통신포트 설정>

- 만약, 펌웨어 버전이 다릅니다. 펌웨어를 업데이트 하시겠습니까? 라는 메시지박스가 나타난다면 YES 버튼을 눌러 제품의 펌웨어를 MP STUDIO 버전에 맞는 펌웨어 버전으로 업데이트해 주시기 바랍니다. 펌웨어 업데이트 이후에, 다시 한번 프로그램 다운로드를 진행해 주셔야지 제품으로 프로그램이 다운로드 됩니다.



4.3.4.2. 펌웨어 다운로드

- 펌웨어 파일을 수동으로 선택하여 MPS, MPA 제품군에 다운로드 합니다. MP STUDIO 버전에 따라 요구되는 펌웨어 버전이 다를 수 있습니다. 요구되는 펌웨어는 MP STUDIO₩Firmware₩ 폴더에 mpf 확장명 파일로 포함되어 있습니다. 펌웨어 파일은 mpsxxx.mpf 또는 mpinoxxx.mpf 파일로 존재하며 제품별 하단과 같은 펌웨어를 사용합니다.

모델명	펌웨어 파일명	모델명	펌웨어 파일명
MPS-8A4R(T)	MPSxxx.mpf	MPS-8A8R	MPINOxxx.mpf
MPS-8A4RS		MPS-16A8R8T	
MPS-8A8R(T)-S		MPA 제품군	
MPS-16A16R(T)			
xxx 는 펌웨어 버전			

- 프로그램 다운로드를 수행하게 되면 펌웨어 버전의 업데이트가 요구되는 경우에는 "펌웨어 다운로드를 수행하겠습니까?" 라는 메시지박스가 나타납니다. 이때, YES 버튼을 누르면 자동으로 요구되는 펌웨어가 제품에 다운로드 되기 때문에 특수한 경우가 아니라면 펌웨어 다운로드를 수행하지 않아도 됩니다.

4.3.4.3. 펌웨어 버전 확인

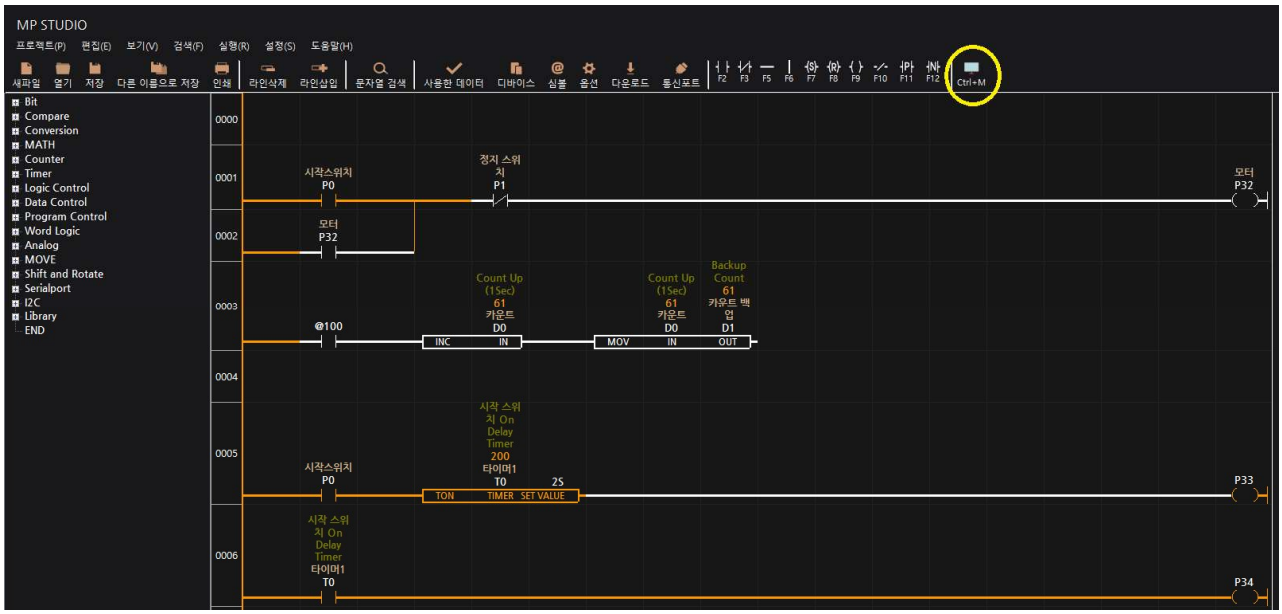
- MPS, MPA 제품군에 다운로드 되어 있는 펌웨어 버전의 정보를 확인할 수 있습니다.

4.3.4.4. 모니터링

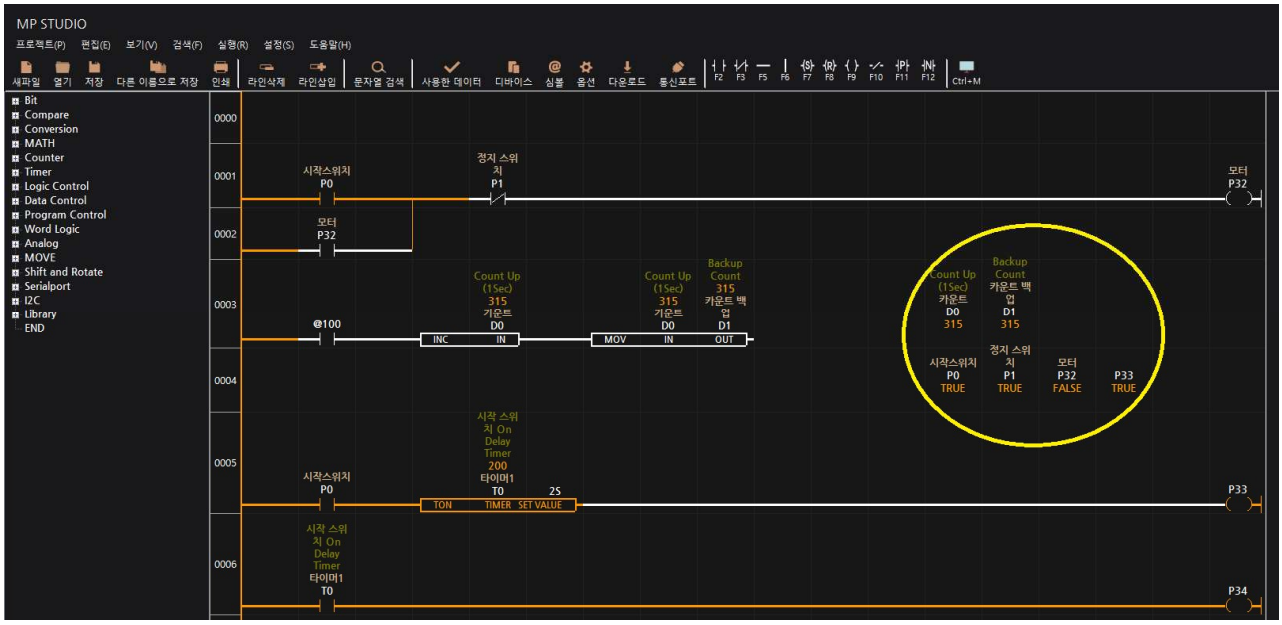
- MPS, MPA 제품군의 데이터 메모리의 값을 실시간으로 LD(Ladder Logic)언어 창에 디스플레이 합니다.

- “CTRL + M” 단축키를 사용하여 모니터링을 ON/OFF할 수 있습니다.

- 모니터링이 동작중일 때에는 상단의 단축아이콘중 마지막에 위치해 있는 모니터링 버튼의 색상이 파란색으로 변경됩니다.



- 그리드에 데이터 메모리를 기입하여 모니터링 할 수도 있습니다.



- 모니터링 동작중 LD(Ladder Logic) 작성차에서 마우스 우클릭하여 나타나는 메뉴를 사용하여 모니터링의 표시방법을 변경할 수 있습니다.

	Monitoring Off : 모니터링을 OFF 합니다.
	Signed Decimal : 부호가 있는 형태로 모니터링 값을 표시합니다.
	Unsigned Decimal : 부호가 없는 형태로 모니터링 값을 표시합니다.
	Hexadecimal : 16진수 형태로 모니터링 값을 표시합니다.
	Binary : 2진수 형태로 모니터링 값을 표시합니다.
	메모리에 값 쓰기 : 워드, 더블워드에서 마우스 우클릭하여 사용시 해당 데이터 메모리에 값을 쓰기합니다.

※ 실수영역 R 메모리는 표시방법의 선택과 상관없이 실수값으로 표현됩니다.

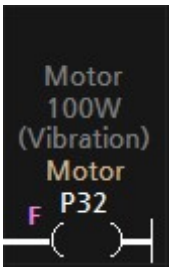
- 강제 I/O를 사용하고 있는 포트는 "F"로 강제 I/O를 적용하고 있다고 표시가 됩니다.

< A접점 강제 ON >

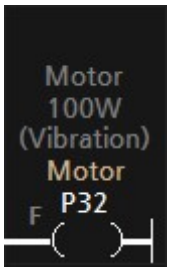
< A접점 강제 OFF >

< B접점 강제 ON >

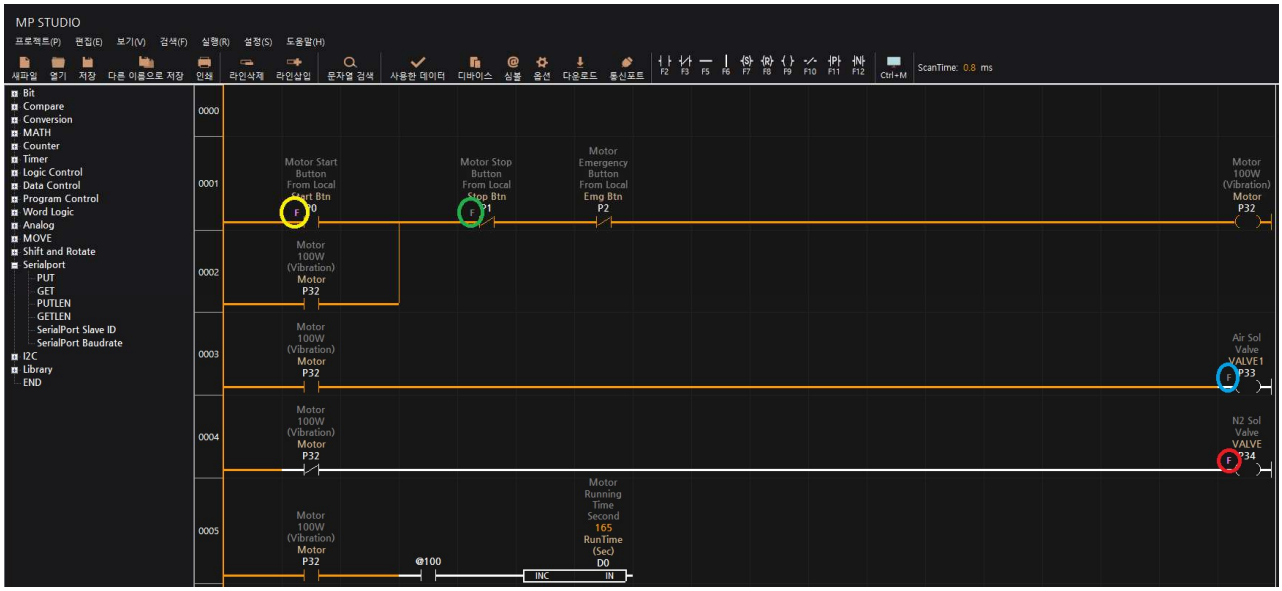
< B접점 강제 OFF >



< COIL 강제 ON >

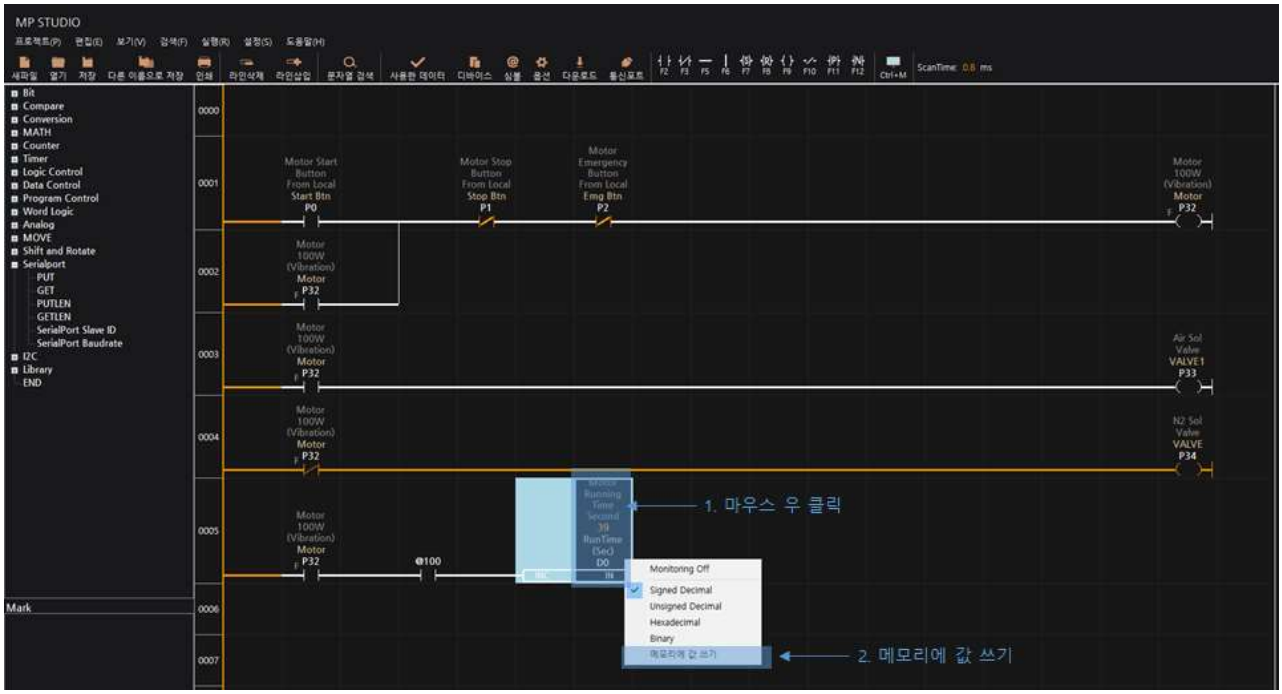


< COIL 강제 OFF >



- 강제 I/O의 "F"문자열은 A, B, COIL, SET COIL, RESET COIL 접점에만 표시됩니다.
- 출력포트에 강제 I/O로 ON을 시키면 분홍색 "F"문자열로 강제 I/O가 ON이 되었다고 표기 되지만 모니터링의 동작상태를 강제 I/O랑 상관 없이 표시됩니다.

- 모니터링 동작중 워드 또는 더블워드 메모리에서 마우스 우클릭하여 나타나는 메뉴에서 "메모리에 값 쓰기"를 실행하여 데이터 메모리에 값을 쓰기할 수 있습니다.

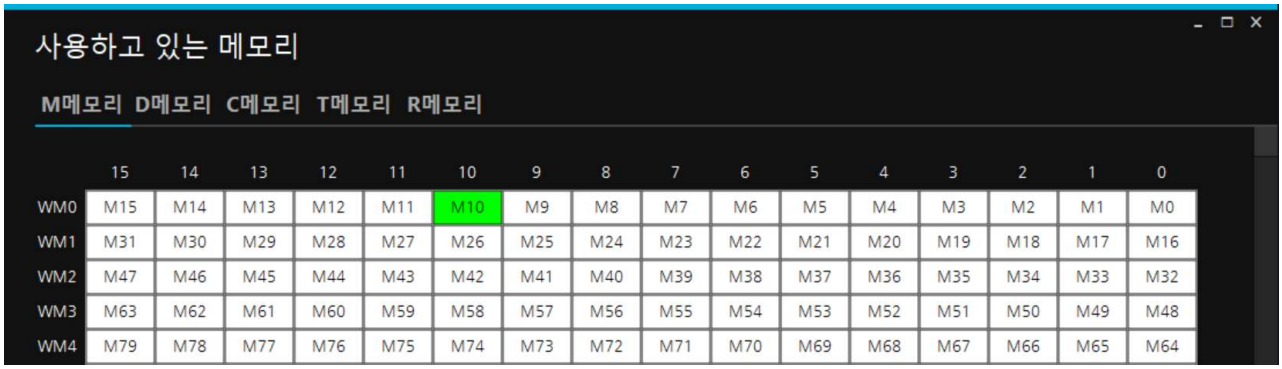


- "메모리에 값 쓰기"를 실행하면 나타나는 팝업창에 쓰기 하고 싶은 값을 입력하면 PLC 제품의 데이터 메모리 값이 변경됩니다.

4.3.4.5. 사용한 데이터 확인

- LD(Ladder Logic)언어 창에서 사용한 데이터 메모리의 영역을 확인할 수 있습니다.

- LD(Ladder Logic)언어 창에서 M10을 사용하고 있다면 아래와 같이 표기됩니다.



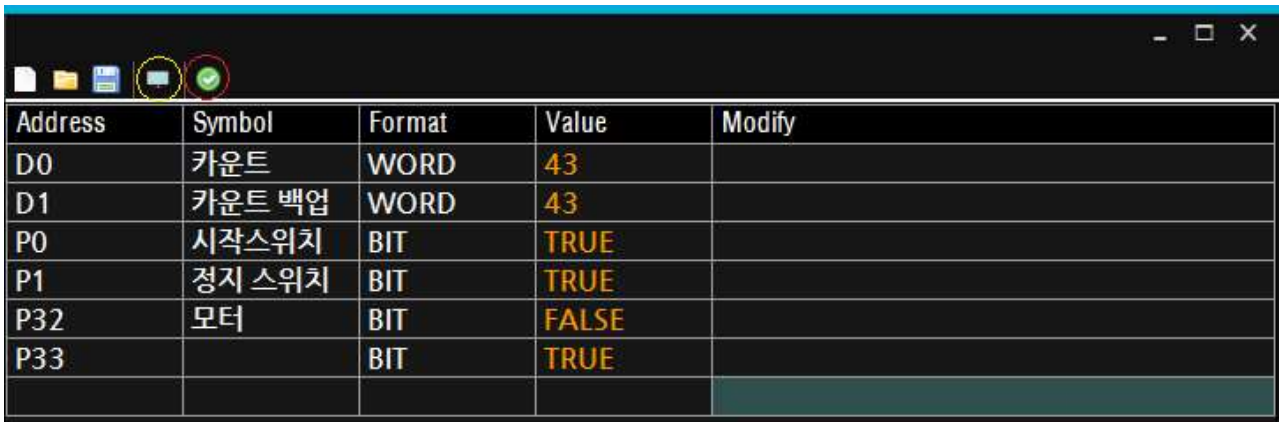
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WM0	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
WM1	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16
WM2	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32
WM3	M63	M62	M61	M60	M59	M58	M57	M56	M55	M54	M53	M52	M51	M50	M49	M48
WM4	M79	M78	M77	M76	M75	M74	M73	M72	M71	M70	M69	M68	M67	M66	M65	M64

- LD(Ladder Logic)언어 창에서 D0를 사용하고 있다면 아래와 같이 표기됩니다.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0	D0.15	D0.14	D0.13	D0.12	D0.11	D0.10	D0.9	D0.8	D0.7	D0.6	D0.5	D0.4	D0.3	D0.2	D0.1	D0.0
D1	D1.15	D1.14	D1.13	D1.12	D1.11	D1.10	D1.9	D1.8	D1.7	D1.6	D1.5	D1.4	D1.3	D1.2	D1.1	D1.0
D2	D2.15	D2.14	D2.13	D2.12	D2.11	D2.10	D2.9	D2.8	D2.7	D2.6	D2.5	D2.4	D2.3	D2.2	D2.1	D2.0
D3	D3.15	D3.14	D3.13	D3.12	D3.11	D3.10	D3.9	D3.8	D3.7	D3.6	D3.5	D3.4	D3.3	D3.2	D3.1	D3.0
D4	D4.15	D4.14	D4.13	D4.12	D4.11	D4.10	D4.9	D4.8	D4.7	D4.6	D4.5	D4.4	D4.3	D4.2	D4.1	D4.0

4.3.4.6. 데이터 테이블

- MPS, MPA 제품군의 데이터 메모리의 값을 테이블에서 모니터링할 수 있습니다.



Address	Symbol	Format	Value	Modify
D0	카운트	WORD	43	
D1	카운트 백업	WORD	43	
P0	시작스위치	BIT	TRUE	
P1	정지 스위치	BIT	TRUE	
P32	모터	BIT	FALSE	
P33		BIT	TRUE	

- Address에 데이터 메모리를 기입하면 Symbol, Format이 자동 기입됩니다.

- 노란색으로 표기한 모니터링 단축아이콘을 실행하여 MPS, MPA 제품군의 데이터 메모리의 값을 데이터 테이블에서 실시간으로 확인할 수 있습니다.

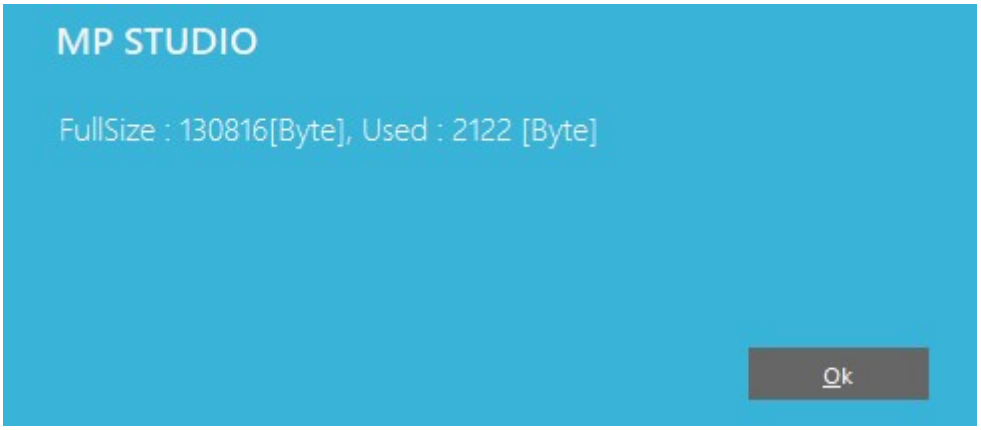
- Modify에 메모리 값을 입력하고 빨간색으로 표기한 Force 단축아이콘을 실행하여 MPS, MPA 제품군의 데이터 메모리 값을 수정할 수 있습니다. (단, Format이 BIT인 메모리는 현 버전에서는 데이터 테이블에서 Force를 지원하지 않습니다. MP STUDIO의 실행->강제I/O에서 Force를 사용할 수 있습니다)

- 삭제는 삭제하고자 하는 줄에서 DEL 키를 눌러 삭제할 수 있습니다.

- 작성한 데이터 테이블은 *.DT 확장명으로 열기/저장할 수 있습니다.

4.3.4.7. 프로그램 메모리 용량 확인

- 제품에 다운로드 되어 있는 프로그램 메모리 용량을 확인할 수 있습니다.
- FullSize : 프로그램 메모리 용량
- Used : 사용하고 있는 프로그램 메모리 용량 + 펌웨어 메모리 용량(256 ~ 1,024 Byte)

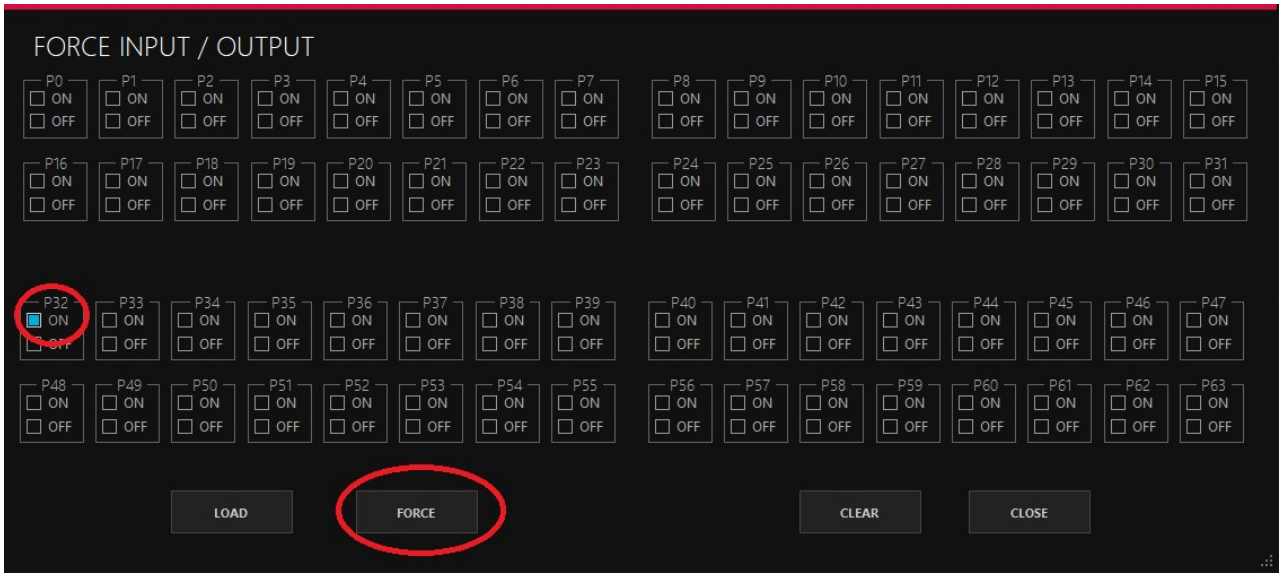


4.3.4.8. 강제 I/O

- 제품의 디지털 입력포트에 연결된 전기신호를 무시하고 강제로 디지털입력 상태값을 On/Off 시킬 수 있습니다.
- LD(Ladder Logic)언어 실행 결과와 상관없이 제품의 디지털 출력포트의 출력을 강제로 On/Off 시킬 수 있습니다.

[ON 체크 : 강제로 ON 시킵니다]
[OFF 체크 : 강제로 OFF 시킵니다]
[ON / OFF 체크 : 두 개 모두 체크할 경우, OFF 체크가 우선시 됩니다]

- 아래 그림처럼 디지털출력:P32의 ON을 체크하고 FORCE 버튼을 실행하면 LD(Ladder Logic)언어의 시퀀스를 무시하고 강제로 제품의 디지털출력:P32를 ON 시킵니다.



- LOAD 버튼은 제품에 적용되어 있는 FORCE 상태를 불러와서 체크박스에 적용시킵니다.

- CLEAR 버튼은 체크박스의 체크를 모두 해제시킵니다. (제품의 FORCE 상태가 해제되는 것은 아닙니다)

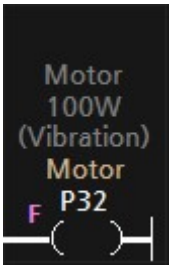
- 모니터링시 강제 I/O를 사용하고 있는 접점에 "F" 문자열이 표시됩니다.

< A접점 강제 ON >

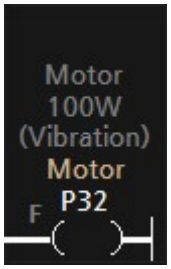
< A접점 강제 OFF >

< B접점 강제 ON >

< B접점 강제 OFF >



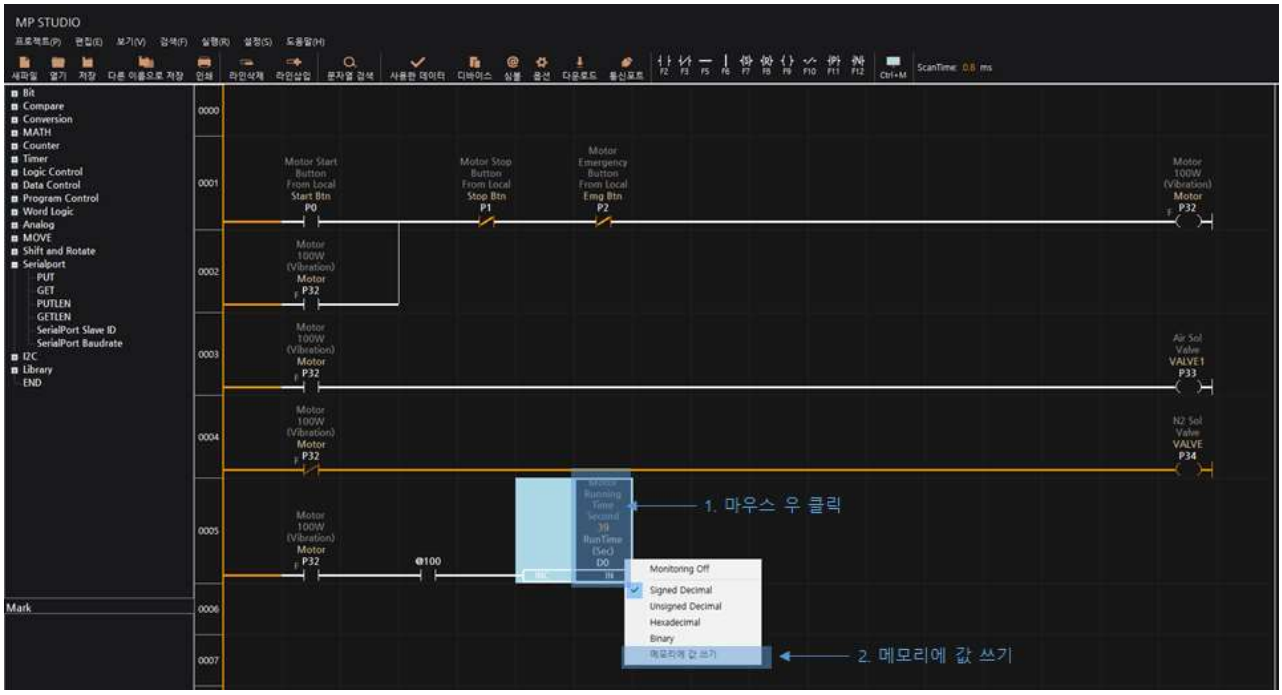
< COIL 강제 ON >



< COIL 강제 OFF >

- 강제 I/O의 "F"문자열은 A, B, COIL, SET COIL, RESET COIL 접점에만 표시됩니다.
- 출력포트에 강제 I/O로 ON을 시키면 분홍색 "F"문자열로 강제 I/O가 ON이 되었다고 표기되지만 모니터링의 동작상태를 강제 I/O랑 상관 없이 표시됩니다.

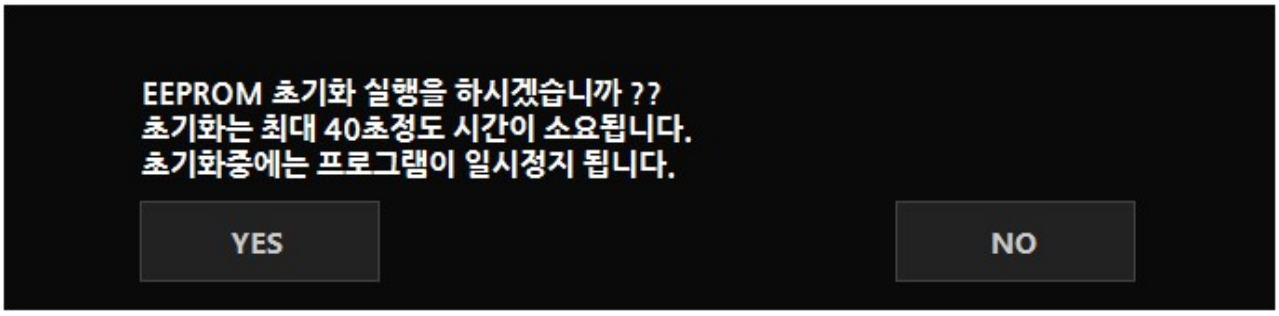
- 모니터링 동작중 워드 또는 더블워드 메모리에서 마우스 우클릭하여 나타나는 메뉴에서 "메모리에 값 쓰기"를 실행하여 데이터 메모리에 값을 쓰기할 수 있습니다.



- "메모리에 값 쓰기"를 실행하면 나타나는 팝업창에 쓰기 하고 싶은 값을 입력하면 PLC 제품의 데이터 메모리 값이 변경됩니다.

4.3.4.9. EEPROM 초기화

- 제품의 비휘발성 메모리인 EEPROM을 모두 0으로 초기화 합니다. EEPROM 초기화는 최대 40초의 시간이 소요되며 초기화하는 동안에 MP STUDIO 및 제품은 일시정지되어 제어할 수 없습니다.

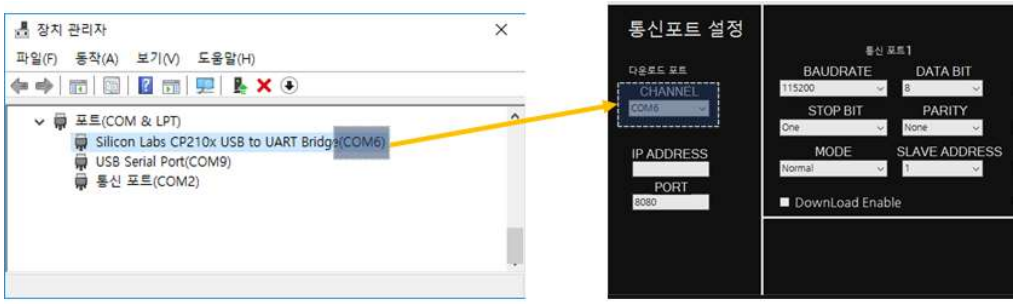


4.3.5. 설정

4.3.5.1. 통신포트 설정

▢ 다운로드 케이블을 통하여 프로그램을 다운로드

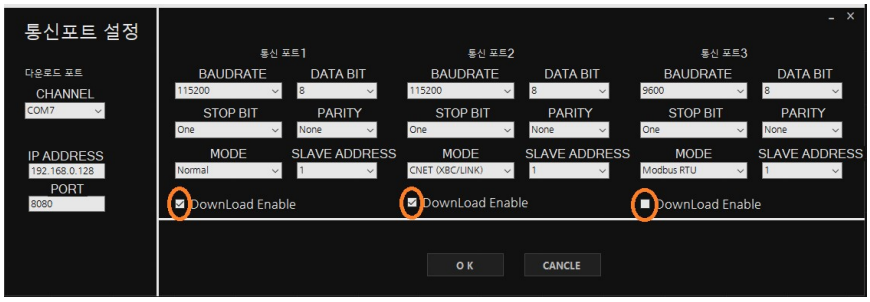
- 다운로드 포트의 CHANNEL에 윈도우의 장치관리자에서 컴퓨터와 제품을 연결하면 생성되는 COM포트를 선택하여 LD(Ladder Logic)언어 창에서 작성한 프로그램을 해당 COM포트를 통하여 다운로드할 수 있습니다.



< 윈도우 장치관리자 > < MP STUDIO 통신포트 설정>

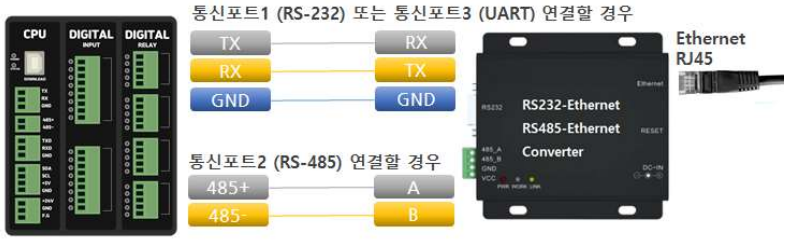
▢ ETHERNET을 통하여 프로그램을 다운로드 및 모니터링

- 통신포트 설정에서 다운로드 포트로 사용하고자 하는 포트의 "Download Enable"을 체크하고 1회 이상 프로그램 다운로드를 진행하여 제품에 Download Enable을 적용할 수 있습니다.



< 다운로드 포트로 사용할 통신포트의 "Download Enable"을 체크 >

- MPS/MPA 제품군은 Ethernet을 지원하지 않기 때문에, 별도의 Ethernet To RS-232, RS-485, UART Converter를 사용하여 연결할 수 있습니다. 별도의 컨버터가 없을 경우 자사에서 판매하고 있는 M-ETHER(Ethernet To RS-232 or RS-485 Converter)를 사용해 주시기 바랍니다.

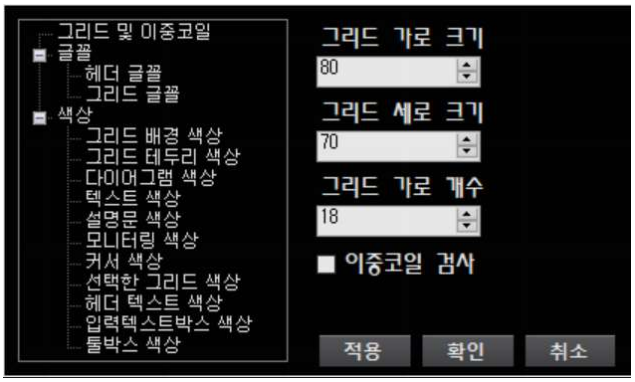


▢ 통신포트1, 통신포트2, 통신포트3

- 통신포트1 ~ 3에서 제품의 RS-232, RS-485, UART의 통신의 파라미터를 설정할 수 있습니다. MPS-8A8R(T)-S 및 MPS-16A16R(T) 제품 이외의 제품은 통신포트1:RS-232, 통신포트2:RS-485, 통신포트:UART를 제공하고 있습니다.

Parameter	Comment	
Baudrate	통신속도입니다. 단위는 bps(bit/sec) 이며, 외부제품의 Baudrate와 동일하게 설정해야 합니다.	
Data Bit	데이터 패킷을 구성하는 Bit 수량입니다. 일반적으로 8 Bit를 사용합니다. 외부제품의 Data Bit와 동일하게 설정해야 합니다.	
Stop Bit	데이터 패킷에 Stop Bit의 개수입니다. 일반적으로 One을 사용합니다. 외부제품의 Stop Bit와 동일하게 설정해야 합니다.	
Parity Bit	데이터 패킷에 오류가 있는지 검출하는 제어방식입니다. 일반적으로 None을 사용합니다. 외부제품의 Parity Bit와 동일하게 설정해야 합니다.	
Mode	Normal	PUT, GET 등의 펑션블럭으로 프로토콜을 사용자가 정의하여 사용할 수 있습니다.
	Modbus RTU Slave	산업 범용 프로토콜로서, Modbus RTU Master를 지원하는 외부제품과 통신할 수 있습니다. 일반적으로 HMI와 연결하실 때 사용합니다.
	Cubloc Modbus RTU Slave	Comfile HMI 제품에서 사용하는 프로토콜입니다.
	CNET(XBC/LINK)	LS산전에서 제공하는 프로토콜입니다. CNET(XBC/LINK) Master를 지원하는 외부제품과 통신할 수 있습니다. 일반적으로 HMI와 연결하실 때 사용합니다.
	IBUS Master	자사에서 개발한 프로토콜입니다. 자사 제품끼리 통신을 하기 위해 사용합니다. 1개의 Master에서 최대 32개의 Slave 제품을 LD(Ladder Logic)언어로 제어할 수 있습니다.
	IBUS Slave	
Slave Address	Master 제품(외부제품)과 통신을 할 경우 사용합니다. CNET(XBC/LINK) 프로토콜에서는 국번을 뜻합니다. 1대의 Master에 여러대의 Slave가 연결되므로 Slave는 각각 다른 Slave Address로 설정해야 합니다.	
Download Enable	LD(Ladder Logic)언어 창에서 작성한 프로그램을 해당 통신포트로 다운로드가 가능하도록 설정합니다. Download Enable을 설정하면, 다운로드 포트 이외에도 전 통신포트를 통하여 프로그램을 다운로드할 수 있습니다. Download Enable은 Baudrate를 115,200으로 설정해야 합니다.	

4.3.5.2. 옵션



▢ 그리드 가로 크기 및 세로 크기

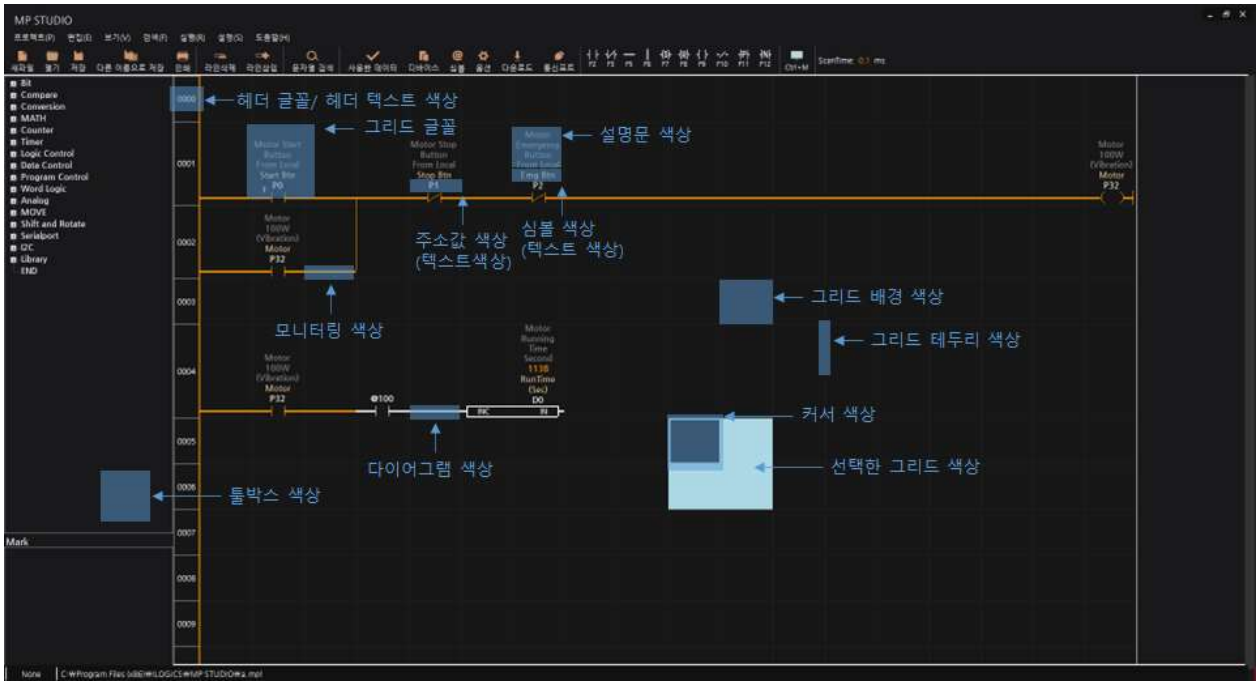
▢ 그리드 가로 개수

- 가로로 몇 개의 그리드를 사용할지 설정할 수 있다. 최대 25개 까지 설정할 수 있다.

▢ 이중코일 검사

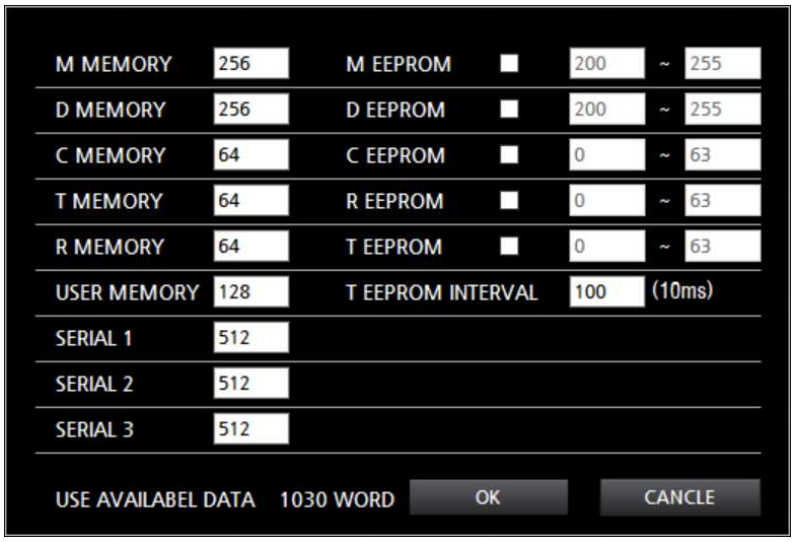
- 다운로드할 때 작성한 Ladder Logic에서, 이중코일이 있는지 검사의 유무를 설정한다.

▢ 색상



4.3.5.3. 데이터 메모리 설정

▢ 데이터 메모리의 크기를 워드 단위로 조절할 수 있습니다.



▢ **"M MEMORY"**는 일반적으로 BIT 단위로 사용하며, 100으로 설정하면 100(Word) x 16(Bit) = 1,600(Bit)개의 M메모리를 사용할 수 있습니다.

▢ **"D MEMORY"**는 일반적으로 WODR 단위로 사용하며, 256으로 설정하면 256(Word)개의 D 메모리를 사용할 수 있습니다.

▢ **"C MEMORY"**는 일반적으로 COUNTER 펑션블럭(CTU, CTD, CTUD)에 사용하며, 64로 설정하면 64(Word)개의 C 메모리를 사용할 수 있습니다. C 메모리를 D 메모리처럼 사용할 수도 있습니다.

▢ **"T MEMORY"** 일반적으로 TIMER 펑션블럭(TON, TOFF, TPL, TMON, TMR)에 사용하며, 64로 설정하면 64(Word)개의 T 메모리를 사용할 수 있습니다. T 메모리를 D 메모리처럼 사용할 수도 있습니다.

▢ **"R MEMORY"**는 32비트 실수 메모리이며, 64로 설정하면 64(Double Word)개의 R 메모리를 사용할 수 있습니다.

▢ **"USER MEMORY"**는 사용자가 정의할 수 있는 메모리입니다. LD(Ladder Logic)언어 작성창에서 #, ##, ###을 사용하여 각각 Bit, Word, Double Word 메모리를 생성하여 사용할 수 있습니다. (Ex) #Motor , ##RunTime, ###Count) 128을 설정하면 128(Word)개의 유저 메모리를 사용할 수 있습니다.

▢ **"SERIAL1, SERIAL2, SERIAL3"**은 각각 통신채널1, 통신채널2, 통신채널3의 송신 및 수신버퍼 크기입니다. 512로 설정하면 256(Word)개의 송신버퍼와 256(Word)개의 수신버퍼를 사용할 수 있습니다. 통신채널 송신 및 수신버퍼의 크기가 부족할 경우 HMI 등의 외부제품과 통신에서 문제가 발생할 수 있습니다.

▢ **"USER AVAILABLE DATA"**는 현재 설정되어 있는 데이터 메모리에서 추가적으로 할당할 수 있는 여분의 메모리입니다.

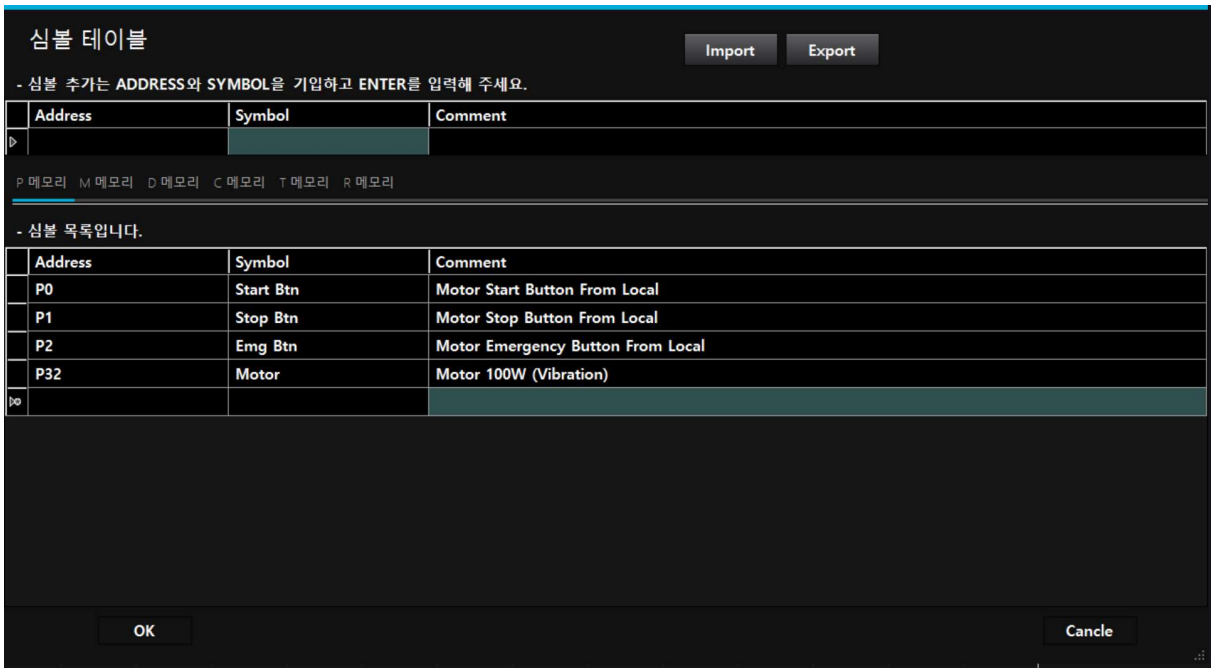
▢ EEPROM을 사용하여 정전시 메모리 값을 유지시킬 수 있습니다. EEPROM을 설정하면 설정된 메모리 상태값이 변경될 때 비휘발성 메모리인 EEPROM에 데이터값을 저장합니다. EEPROM 메모리는 비휘발성 메모리인 장점이 있지만 FLASH 메모리처럼 최소 100,000번 이상의 쓰기 동작에 해당 섹터가 불량이 되어 읽기 동작에 쓰레기 값이 읽혀질 수 있습니다. 때문에 비번히 변경되는 데이터 값에 EEPROM을 설정하면 안됩니다. 때문에 EEPROM 영역은 메모리의 뒷부분에 사용자가 구분하여 사용할 수 있도록 하는 것을 추천드립니다.

▢ **"M EEPROM"**을 200 ~ 255으로 설정하면, M3200 ~ M4095 메모리를 정전유지 메모리로 사용할 수 있습니다.
(200(Word) x 16(Bit) = M3200, 255(Word) x 16(Bit) + 15(Bit) = M4095

▢ **"D EEPROM"**을 200 ~ 255으로 설정하면, D200 ~ D255 메모리를 정전유지 메모리로 사용할 수 있습니다.

▢ **"T EEPROM INTERVAL"**은 타이머 펑션블럭이 동작중일 때, 10ms마다 변경되는 값이므로 지속적인 EEPROM에 쓰기 동작으로 인한 불량 섹터 발생을 최소화하기 위하여 INTERVAL을 설정할 수 있습니다. INTERVAL을 100으로 설정하면 1초(1000ms) 마다 T MEMORY값이 변경되었을 때 EEPROM에 쓰기 동작을 하여 정전유지를 수행합니다.

4.3.5.4. 심볼 테이블



☐ **"심볼 추가"**는 상단의 심볼 추가 테이블에서 ADDRESS, SYMBOL을 입력하고 키보드의 ENTER키를 누르면 하단의 심복 목록탭에 심볼이 추가됩니다. 심볼을 모두 추가하고 하단의 OK버튼을 실행해야 추가된 심볼이 적용됩니다.

☐ 하단의 심볼 목록 테이블에서도 ADDRESS, SYMBOL을 입력하여 심볼을 추가할 수 있습니다.

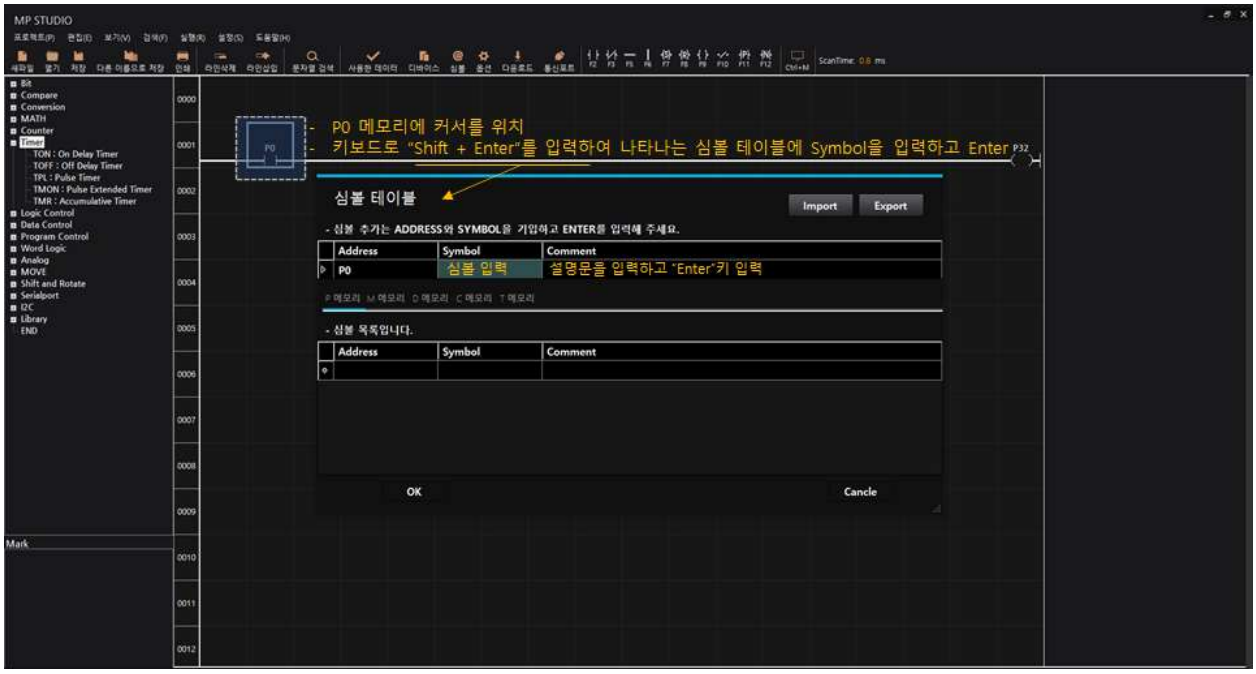
☐ 심복 목록에서 보여지는 메모리는 심볼 테이블의 중간쯤에 위치해 있는 **"P 메모리, M 메모리, D 메모리, C 메모리, T 메모리, R 메모리"** 탭으로 메모리 타입을 구분하여 표시됩니다.

☐ **"Export"** 버튼으로 심볼 목록 테이블을 *.csv 파일로 내보내기할 수 있습니다. 내보내기한 파일을 Excel, Notepad 등으로 심볼 목록을 수정하고 Import하여 대량의 심볼을 보다 편리하게 작업할 수 있습니다.

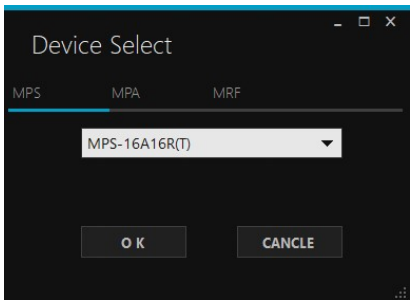
☐ 심볼은 아래와 같은 특징이 있습니다.

- 대/소문자를 구분하지 않습니다.
- 띄어쓰기가 허용됩니다.
- 첫글자는 "@"로 시작할 수 없습니다.
- "<", ">", "=" 와 같은 부등호를 사용할 수 없습니다.

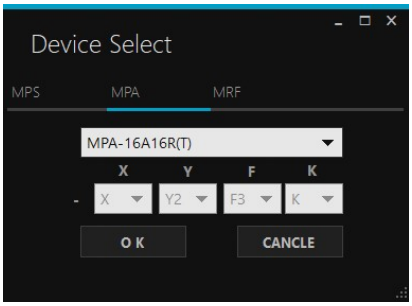
▢ LD(Ladder Logic)창에서 커서를 메모리에 위치하고 "Shift + Enter"를 입력하여 빠른 심볼 등록을 할 수 있습니다.



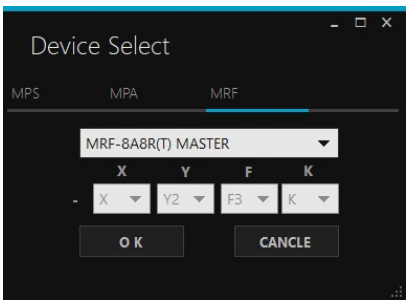
4.3.5.5. 장치명 (디바이스 선택)



< MPS >



< MPA >



< MRF >

▢ 장치명은 MPS, MPA, MRF 탭으로 구성되어 있습니다. 사용하고자 하는 제품명의 앞 세글자와 동일한 탭을 선택하시고 디바이스 목록에서 제품명을 선택하시고 OK 버튼을 눌러야 합니다.

▢ MPS-16A16**R**과 MPS-16A16**T**과 같은 제품은 소프트웨어적으로 동일한 특성을 가지므로 MPS-16A16**R(T)**로 표시해 놓았습니다.

▢ MPA 및 MRF 제품군에서 특수기능모듈을 추가했을 경우 아날로그입력(X), 아날로그출력(Y), PT100Ω 온도센서입력(F), 고속펄스출력(K)을 선택해야 합니다.

5. 접점

5.1. 접점 종류

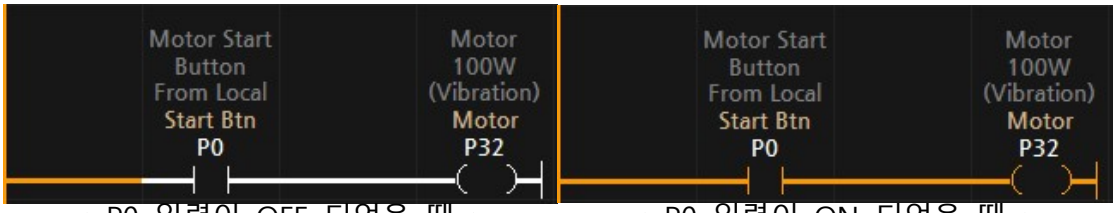
아이콘	이름	설 명	단축키
┤ ├	N.O 접점 (A접점) (Normal Open)	평상시에는 끊어져있다가, 접점값이 "HIGH" 일때, 연결이 되는 접점	F2
┤/├	N.C 접점 (B접점) (Normal Close)	평상시에는 연결되어있다가. 접점값이 "HIGH" 일때, 연결이 끊어지는 접점	F3
———	가로 연결선	가로 방향으로 접점을 연결해주는 선	F5
|	세로 연결선	세로 방향으로 접점을 연결해주는 선	F6
-(S)-	SET COIL	입력조건이 성립이 되면, "HIGH"를 출력	F7
-(R)-	RESET COIL	입력조건이 성립이 되면, "LOW"를 출력	F8
-( )-	COIL	입력조건이 되면, "HIGH"를 출력하고, 조건이 되지 않으면, "LOW"를 출력하는 코일	F9
-/-	NOT 접점	입력조건이 "LOW"일 경우, 연결이 된다. 입력조건이 "HIGH"일 경우, 연결이 끊긴다.	F10
┤P├	Positive Edge 접점	입력조건이 "LOW"에서 "HIGH"로 변할 때, 한 스캔동안만 연결되고 끊기는 접점	F11
┤N├	Negative Edge 접점	입력조건이 "HIGH"에서 "LOW"로 변할 때, 한 스캔동안만 연결되고 끊기는 접점	F12

5.2. N.O접점 및 N.C접점

5.2.1. N.O 접점

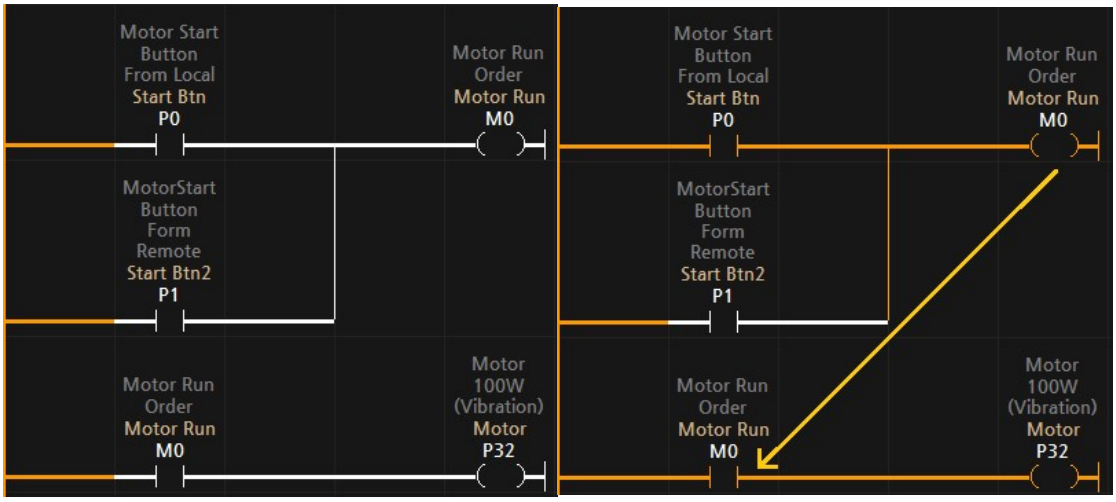
▢ Normal Open 접점으로서, 평상시에는 열려있다가 접점비트가 ON되면 닫히는 접점입니다.

▢ 디지털입력:P0가 ON이되면, 디지털출력:P32가 ON되고 디지털입력:P0가 OFF되면, 디지털출력:P32가 OFF되는 프로그램을 모니터링하면 아래와 같습니다.



< P0 입력이 OFF 되었을 때 > < P0 입력이 ON 되었을 때 >

▢ 디지털입력:P0 또는 P1이 ON 되었을 때, 비트메모리:M0를 ON 시키고, M0 가 ON이 되면, 디지털출력:P32를 ON 시키는 프로그램을 모니터링 하면 아래와 같습니다.



< P0 입력이 OFF 되었을 때 > < P0 입력이 ON 되었을 때 >

5.2.2. N.C 접점

▢ Normal Close 접점으로서, 평상시에는 닫혀있다가 접점비트가 ON되면 열리는 접점입니다.

▢ 디지털입력:P0가 OFF되면 비트메모리:M0가 ON되고, P0가 ON되면, M0가 OFF되는 프로그램을 모니터링하면 아래와 같습니다.

< P0 입력이 OFF 되었을 때 > < P0 입력이 ON 되었을 때 >

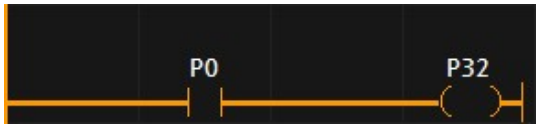
5.3. 코일출력

5.3.1. 코일

▢ 코일의 출력조건이 ON이면 출력비트를 ON시키고, 코일의 출력조건이 OFF이면 출력비트를 OFF시킵니다.

▢ 디지털입력:P0가 ON되어 코일의 출력조건이 ON되어 디지털출력:P32가 ON되고, P0가 OFF되어 코일의 출력조건이 OFF되어 P32가 OFF되는 프로그램을 모니터링하면 아래와 같습니다.

< P0 입력이 OFF 되었을 때 >



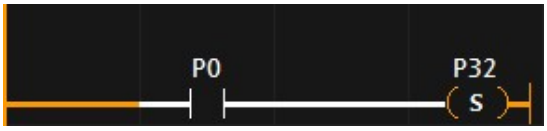
< P0 입력이 ON 되었을 때 >

5.3.2. SET코일

▢ SET코일의 출력조건이 ON이면 출력비트를 ON시킵니다.
(SET코일의 출력조건이 OFF이라도 출력비트는 OFF되지 않습니다)

▢ 디지털입력:P0가 ON되면 디지털출력:P32가 ON되고 이후, P0가 OFF되어도 P32가 계속 ON되는 프로그램을 모니터링하면 아래와 같습니다.

< P0 입력이 ON 되었을 때 >



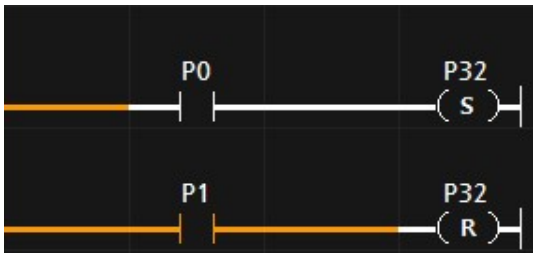
< P0 입력이 OFF 되어도 출력유지 >

5.3.3. RESET코일

▢ RESET코일의 출력조건이 ON이면, 출력비트를 OFF시킵니다.

▢ 디지털입력:P0가 ON되면 디지털출력:P32가 ON되고 이후, 디지털입력:P1이 ON되면, 디지털출력:P32를 OFF시키는 프로그램을 모니터링하면 아래와 같습니다.

< P00 입력이 ON 되었을 때 >



< P01 입력이 ON 되었을 때 >

5.4. 반전접점

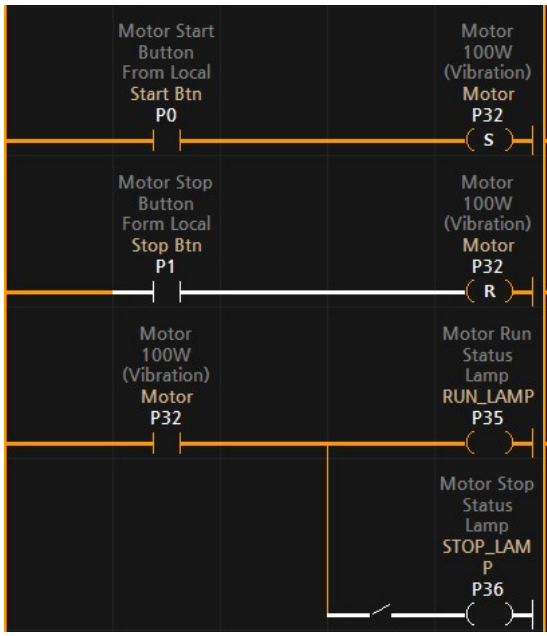
▢ 입력된 상태를 반전시키는 접점입니다.

▢ 디지털입력:P0가 ON되면 반전접점 때문에 출력조건이 ON상태에서 OFF상태로 반전되어 디지털출력:P32가 OFF되고, P0가 OFF되면 출력조건이 반전되어 P32가 ON되는 프로그램을 모니터링하면 아래와 같습니다.

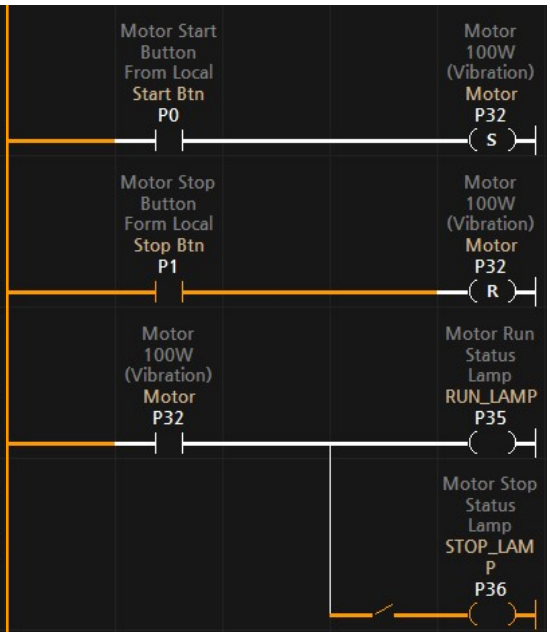
< P0 입력이 ON 되었을 때 >

< P0 입력이 OFF 되었을 때 >

▢ Start Btn:P0가 ON되어 Motor:P32가 ON되어 Motor가 동작중일 때에는 Run_Lamp:P35가 On되고 이후, Stop Btn:P1이 ON되어 Motor:P32가 OFF되어 Run_Lamp:P35가 OFF되고 Stop_Lamp:P36이 ON되는 프로그램을 모니터링하면 아래와 같습니다.



< P0 입력이 ON 되었을 때 >



< P1 입력이 ON 되었을 때 >

5.5. 상승/하강 검출접점 (양변환/음변환 검출접점)

5.5.1. 상승 검출접점 (양변환 검출접점)

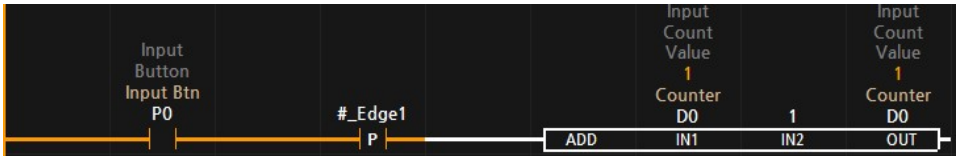
▢ 입력상태가 OFF에서 ON이 되었을 때, 한 스캔(한 번)만 ON되는 접점

▢ 검출 접점의 이전 상태값이 OFF였는지 ON이었는지 확인하기 위해 1개의 비트 메모리를 사용합니다. M비트 메모리를 입력하거나 메모리 값을 입력하지 않으면 컴파일 동작에서 자동으로 사용자 비트 메모리 #Edge(n)가 기입됩니다. 비트 메모리에 P 메모리를 사용할 수 없습니다.

▢ 사용자 비트 메모리 #Edge는 Ladder Logic 유효성 검사 및 컴파일 할 때마다 최상위 좌측부터 #Edge0, #Edge1로 자동기입됩니다.

▢ P0:Input Btn이 ON되었을 때 한 번만 D0:Counter 워드메모리를 1 증가시킵니다. 상승 검출접점:#_Edge1가 없다면 P0가 ON되고 있는 동안 D0는 계속 1씩 증가하게 됩니다. 상승검출접점을 사용함으로서 P0가 ON되고 있어도 상승검출접점 뒤에 있는 ADD 펑션블럭은 한번만 실행되어 D0는 1이 됩니다. 이후, P0가 OFF 되었다가 다시 ON 되었을 때 D0는 2가 됩니다.

< P0 입력이 OFF일 때는 ADD 펑션블럭이 실행되지 않습니다 >



< P0 입력이 OFF -> ON될 때 ADD 펑션블럭은 한번 실행됩니다 >

▢ 검출접점은 비트 메모리 자원을 사용해야 합니다. 때문에, M비트 또는 사용자 비트 메모리를 입력해야 합니다. 위 프로그램에서는 사용자 비트 메모리:#Edge1을 사용하였습니다. 별도의 메모리를 입력하지 않으면, 사용자 비트 메모리로 자동 입력됩니다.

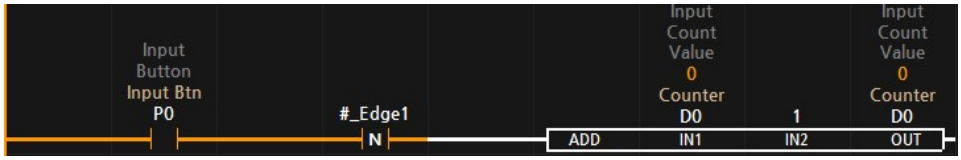
5.5.2. 하강 검출접점 (음변환 검출접점)

▢ 입력상태가 ON에서 OFF가 되었을 때, 한 스캔(한 번)만 ON 시키는 접점입니다.

▢ 검출 접점의 이전 상태값이 OFF였는지 ON이었는지 확인하기 위해 1개의 비트 메모리를 사용합니다. M비트 메모리를 입력하거나 메모리 값을 입력하지 않으면 컴파일 동작에서 자동으로 사용자 비트 메모리 #Edge(n)가 기입됩니다. 비트 메모리에 P 메모리를 사용할 수 없습니다.

▢ 사용자 비트 메모리 #Edge는 Ladder Logic 유효성 검사 및 컴파일 할 때마다 최상위 좌측부터 #Edge0, #Edge1로 자동기입됩니다.

▢ P0:Input Btn이 ON되다가 OFF되었을 때 한 번만 D0:Counter 워드메모리를 1 증가시키는 프로그램입니다. 검출접점의 동작은 상승 검출접점에서 보다 자세하게 설명하고 있습니다.



< P0 입력이 ON 되어도 ADD 펑션은 실행하지 않습니다 >

< P0 입력이 ON -> OFF로 변할 때 ADD 펑션은 한번 실행됩니다 >

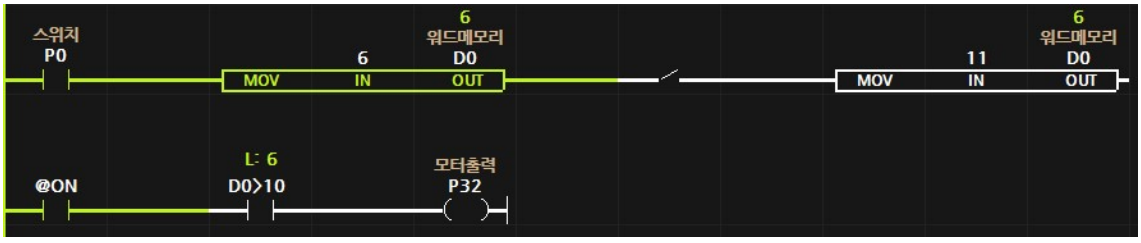
5.6. 비교 접점

▢ 메모리 값을 비교하여 참일 경우에 상태 값을 ON 시킵니다.

접점 명			펑션블럭	설 명
D0	>	D1	HIGHER_I	D0가 D1보다 클 경우, 접점 신호 ON
D0	<	D1	LOWER_I	D0가 D1보다 작을 경우, 접점 신호 ON
D0	>=	D1	SHIGHER_I	D0가 D1보다 크거나 같을 경우, 접점 신호 ON
D0	=>	D1		
D0	<=	D1	SLOWER_I	D0가 D1보다 작거나 같을 경우, 접점 신호 ON
D0	=<	D1		
D0	==	D1	EQUAL_I	D0와 D1이 같을 경우, 접점 신호 ON
D0	=	D1		
D0	!=	D1	NEQUAL_I	D0와 D1이 다를 경우, 접점 신호 ON
D0	<>	D1		

▢ N.O접점 또는 N.C접점에 비교문을 사용할 수 있습니다.

▢ 디지털입력 P0:스위치가 ON이 되면 D0 워드메모리에 6을 저장하고, P0:스위치가 OFF되면 D0 워드메모리에 11을 저장합니다. 이때, D0 워드메모리값이 10보다 클 경우 디지털출력 P32:모터출력을 ON시키고 D0 워드메모리값이 10보다 작거나 같으면 P32:모터출력을 OFF시키는 프로그램입니다.



< P0 입력이 ON 되었을 때 >

< P0 입력이 OFF 되었을 때 >

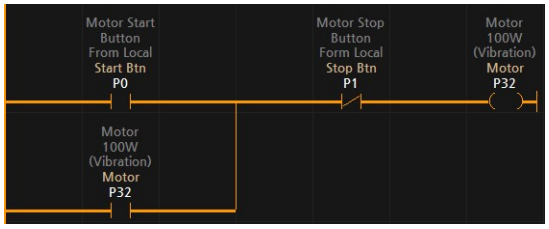
▢ 심볼 또는 사용자 워드 메모리 이름을 사용할 수도 있습니다. 심볼을 사용할 경우 아래와 같이 사용하시기 바랍니다.

<그림 3-6-3 비교접점에 심볼 사용 >

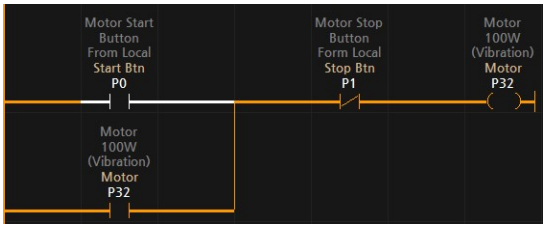
<그림 3-6-4 비교접점에 사용자 워드 메모리 사용 >

5.7. 자기유지 회로

▢ P0:Start Btn이 ON되면 P32:Motor가 ON되면서 P32의 보조접점이 ON된다. 이후, P0를 OFF해도 P32의 보조접점이 ON되어 있어 P32는 계속 ON된다. 이후, P1:Stop Btn을 ON시키면 P32는 OFF된다. 이렇게 P32:Motor의 보조접점에 의해 ON상태가 유지되는 회로를 자기유지회로라고 한다.

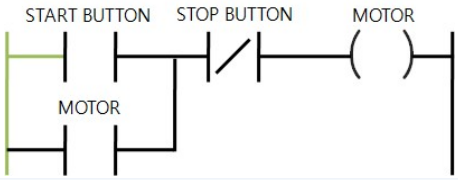
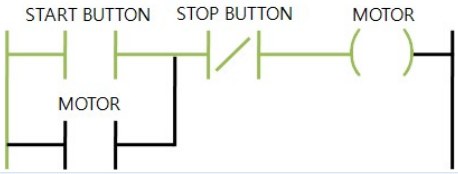
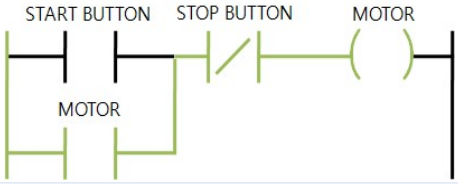
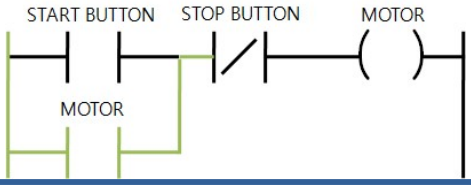


< P0를 ON 시키면 P32가 ON >



< P0를 OFF 시켜도 P32는 ON유지 >

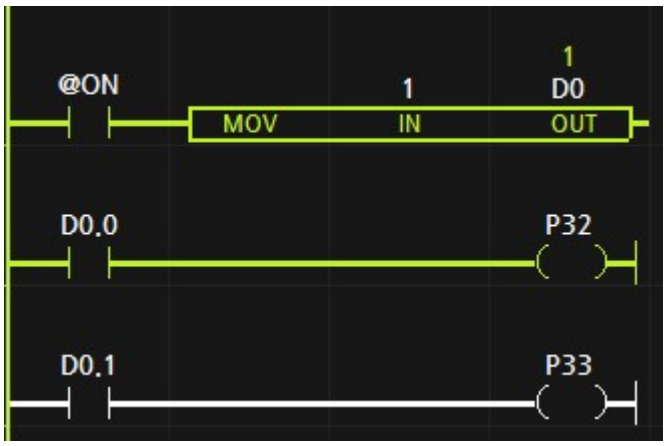
② 동작순서

	버튼을 누르지 않고 있으면, "MOTOR" 코일은 ON조건을 성립하지 않으므로 OFF되고 있다.
	N.O접점인 "START BUTTON"을 누르면,"START BUTTON"은 ON되고 이후에 N.C접점인 "STOP BUTTON"은 버튼을 누르지 않을 때는 연결된 접점이므로, ON된다. 따라서 "MOTOR" 코일은 ON된다.
	"START BUTTON"을 누르다가 띄우면, "START BUTTON"은 OFF된다. 하지만, "MOTOR" 코일이 ON되어, 동시에 ON된 "MOTOR" N.O접점이 ON되어 있어, "MO TOR" 코일은 계속 ON되어진다.
	N.C접점인 "STOP BUTTON"은 누르지 않을 때는 계속 ON되어 있다. 이후, 정지하고 싶을 때, 누르면 이어진 선이 끊기면서 "MOTOR"코일을 OFF시키며, 동시에 "MOTOR" N.O접점도 OFF되어진다.

5.8. 워드 메모리의 비트 접점 사용

▢ **WP, WM, D 워드 메모리의 비트를 제어를 할 수 있습니다.**

- 워드 메모리의 접미어에 " .0~15"를 사용하여 비트 메모리를 지정할 수 있습니다.
- WP0는 디지털입력포트 P0 ~ P15이며, WP0.1은 P1을 의미합니다.
- MW1은 내부 M메모리 M16 ~ M31이며, WM1.15는 M31을 의미합니다.
- 워드 메모리 D10은 D10.0 ~ D10.15의 비트 메모리로 사용할 수 있습니다.
- 워드 메모리 D0에 1을 저장하여 D0.0 비트 메모리가 ON되므로 출력 P32를 ON되는 프로그램입니다.



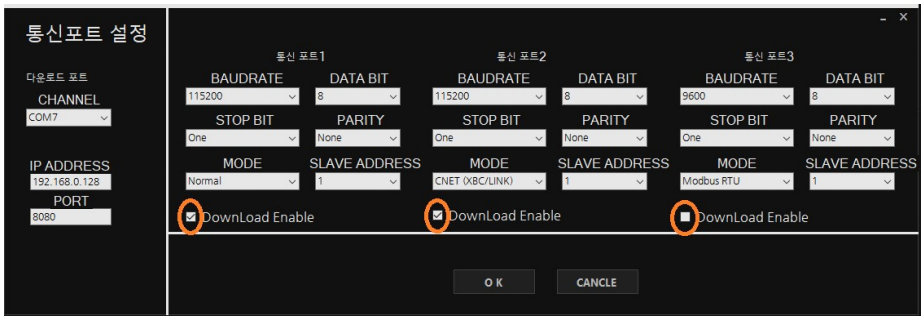
< D 워드 메모리의 비트 제어 예 >

6. 유용한 기능

6.1. ETHERNET 다운로드 및 모니터링

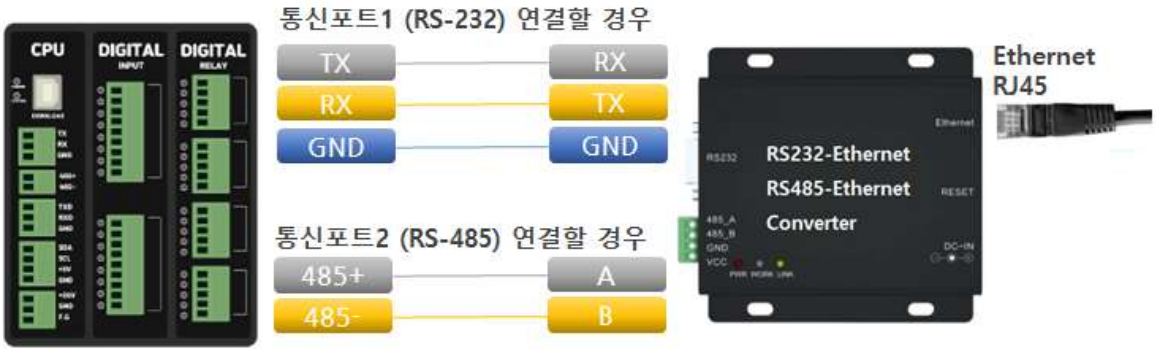
▢ MPS/MPA 전제품군은 다운로드 포트(USB포트) 이외에도 통신포트1(RS-232), 통신포트2(RS-485), 통신포트3(UART)을 통하여 프로그램 다운로드 및 모니터링을 수행할 수 있습니다.

▢ 통신포트 설정에서 다운로드 포트로 사용하고자 하는 포트의 "Download Enable"을 체크하고 1회 이상 프로그램 다운로드를 진행하여 제품에 Download Enable을 적용할 수 있습니다.



< 다운로드 포트로 사용할 통신포트의 "Download Enable"을 체크 >

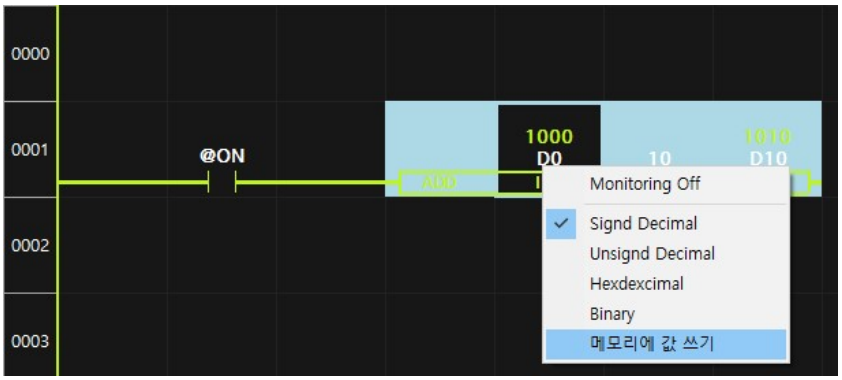
▢ MPS/MPA 제품군은 Ethernet을 지원하지 않기 때문에, 별도의 Ethernet To RS-232, RS-485, UART Converter를 사용하여 연결할 수 있습니다. 별도의 컨버터가 없을 경우 자사에서 판매하고 있는 M-ETHER(Ethernet To RS-232 or RS-485 Converter)를 사용해 주시기 바랍니다.



6.2. FORCE DATA (메모리에 값 쓰기)

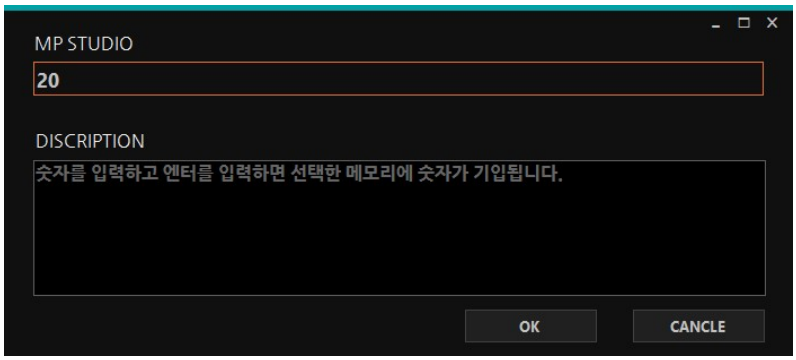
▢ 모니터링 상태에서 PLC의 메모리에 특정 값을 입력할 수 있습니다.

- 모니터링 상태에서 변경하고자 하는 메모리를 선택하고 키보드 ALT + ENTER 입력 또는 마우스 우클릭하여 "메모리에 값 쓰기"를 선택합니다.



<그림 4-14-1 메모리에 값 쓰기 (모니터링 상태에서 마우스 우클릭) >

- 변경하고자 하는 특정 값을 입력합니다.



<그림 4-14-2 메모리에 값 쓰기 (변경할 값 입력) >

- 20을 입력하면 D0의 값이 20이 됩니다.

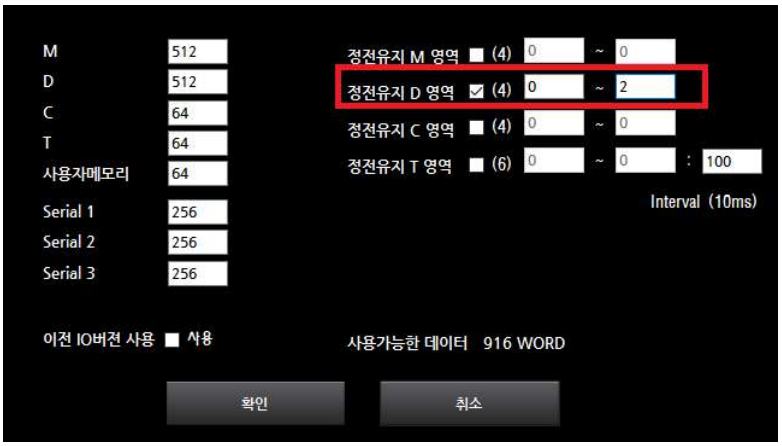
<그림 4-14-3 메모리에 값 쓰기 (입력한 값으로 변경된 D0 메모리) >

6.3. EEPROM 사용법 (제품의 초기설정값 적용법)

▢ 제품을 양산할 경우, 제품의 동작을 위한 파라미터 값을 제품마다 기입해줘야 하는 번거로움을 EEPROM으로 해결할 수 있습니다.

▢ 정전유지 D영역중 1개의 워드를 사용하는 방법으로 EEPROM의 초기값은 0이므로 사용자가 임의의 숫자를 정의하여 정의한 숫자와 EEPROM의 값이 다를 경우 MOV 펑션블럭으로 파라미터 값에 초기값을 이동시키고 EEPROM에 사용자가 정의한 숫자를 MOV 펑션블럭으로 이동시킵니다.

▢ MP STUDIO -> 설정 -> 데이터메모리에서 정전유지 D영역의 사용을 체크하고 범위를 0~2로 설정합니다.

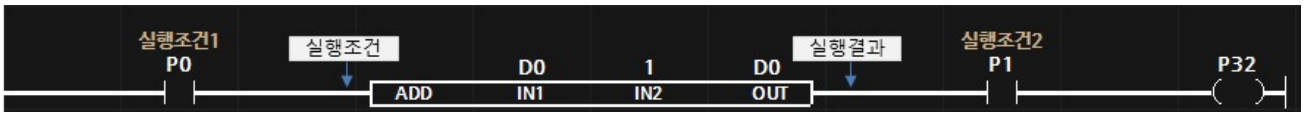


▢ LADDER LOGIC을 아래와 같이 D0가 100이 아닐 때 초기값을 D1에는 1, D2에는 2로 만들고 다시 D0를 100으로 만듭니다.

▢ 위와 같이 할 경우, 처음에는 D0가 0이므로 D1과 D2는 각각 1과 2가 저장되고 D0는 100이 저장 됩니다. D0가 100이 아닐 경우는 제품이 처음 실행될 때 한번 뿐이므로 이후에는 HMI 등의 제품으로 D1 및 D2의 값을 변경할 수도 있습니다.

7. 펑션 블럭

▢ 펑션블럭 실행 조건과 결과



< 펑션블럭의 실행조건과 실행결과 >

▢ 다양한 연산 및 기능을 펑션블럭을 통하여 실행할 수 있습니다.

▢ 펑션블럭은 펑션블럭의 왼쪽의 접점상태가 ON이 될 때 실행되며, OFF일때는 실행하지 않습니다. 또한, 펑션블럭 실행결과를 펑션블럭의 오른쪽으로 ON또는 OFF로 출력됩니다.

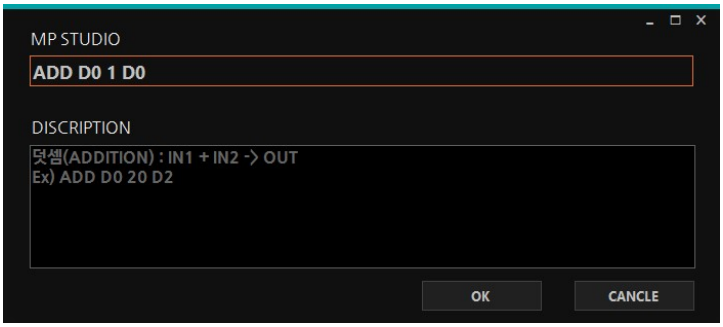
▢ 대부분의 펑션블럭은 실행조건의 접점상태가 실행결과의 접점상태로 출력되지만, 비교펑션, 타이머, 카운터 등은 실행결과가 실행출력됩니다. 또한, 실행중 오류가 발생할 경우, OFF로 실행출력합니다. 예를들어, 펄스출력 펑션블럭 (PWM)의 채널입력 매개변수에 D0(워드 메모리)를 입력해놓았는데, D0가 100일 경우 PWM 출력포트 범위(MPS-8A8R일 경우 0~6)를 넘었으므로 실행출력은 OFF로 출력됩니다.

▢ 펑션블럭의 기능은 각각의 펑션블럭 설명 페이지에서 상세히 확인하실 수 있습니다.

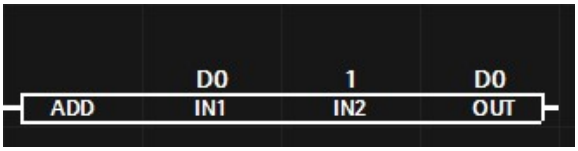
▢ 펑션블럭은 MP STUDIO의 왼쪽에 위치한 "ToolBox" 또는 단축키 " ["를 사용하여 생성할 수 있습니다.

▢ 키보드 " [" 또는 "Ctrl + Enter"를 눌러 생성되는 펑션블럭 입력박스에서 펑션블럭 이름을 입력하면 펑션블럭이 생성됩니다. 또는 펑션블럭 이름과 파라미터 모두를 입력하면 펑션블럭과 파라미터가 기입되어 펑션블럭이 생성됩니다.

▢ 펑션블럭 이름과 매개변수 모두 입력했을 경우



< 펑션블럭 입력박스에서 이름 및 파라미터 입력>



< 생성된 펑션블럭 >

▢ 파라미터를 변경하기 위해서는 파라미터에 커서를 위치하고 "ENTER"키를 실행하면 나타나는 입력박스에 파라미터값을 입력하여 변경할 수 있습니다.

카테고리	펑션블럭명	펑션블럭 이미지	설 명
비교 펑션블럭 Compare	=	= IN1 IN2	(Signed) IN1과 IN2의 값이 같으면 실행결과 ON을 출력
	<>	<> IN1 IN2	(Signed) IN1과 IN2의 값이 다르면 실행결과 ON을 출력
	>	> IN1 IN2	(Signed) IN1의 값이 IN2보다 크면 실행결과 ON을 출력
	<	< IN1 IN2	(Signed) IN1의 값이 IN2보다 작으면 실행결과 ON을 출력
	>=	>= IN1 IN2	(Signed) IN1의 값이 IN2보다 크거나 같으면 실행결과 ON을 출력
	<=	<= IN1 IN2	(Signed) IN1의 값이 IN2보다 작거나 같으면 실행결과 ON을 출력
	u>	u> IN1 IN2	(Unsigned) IN1의 값이 IN2보다 크면 실행결과 ON을 출력
	u<	u< IN1 IN2	(Unsigned) IN1의 값이 IN2보다 작으면 실행결과 ON을 출력
	u>=	u>= IN1 IN2	(Unsigned) IN1의 값이 IN2보다 크거나 같으면 실행결과 ON을 출력
	u<=	u<= IN1 IN2	(Unsigned) IN1의 값이 IN2보다 작거나 같으면 실행결과 ON을 출력

변환 펑션블럭 Conversion	H2D	H2D IN OUT	IN 값을 HEX에서 DEC로 변환하여 OUT에 저장
	D2H	D2H IN OUT	IN 값을 DEC에서 HEX로 변환하여 OUT에 저장
	INV	INV IN OUT	IN 값을 1의 보수를 취하여 OUT에 저장
	NEG	NEG IN OUT	IN 값을 2의 보수를 취하여 OUT에 저장

수학 펑션블럭 MATH	ADD	ADD IN1 IN2 OUT	IN1과 IN2의 합을 OUT에 저장
	SUB	SUB IN1 IN2 OUT	IN1과 IN2의 차를 OUT에 저장
	MUL	MUL IN1 IN2 OUT	IN1과 IN2의 곱을 OUT에 저장
	DIV	DIV IN1 IN2 OUT REM	IN1과 IN2를 나누어 몫을 OUT에 저장, 나머지를 REM에 저장
	INC	INC IN	IN을 1 증가시킴
	DEC	DEC IN	IN을 1 감소시킴
	SCALE	SCALE IN IN_MAX IN_MIN OUT OUT_MAX OUT_MIN	IN_MIN~IN_MAX의 범위에서 IN값을 OUT_MIN ~ OUT_MAX의 범위로 변경하여 OUT에 저장
	FLOOR	FLOOR IN OUT	IN의 실수값에서 정수값을 OUT에 저장
카운터 펑션블럭 Counter	CTU	CTU COUNT SETVALUE RESET	실행조건이 ON 될 때마다 COUNT를 1씩 증가시켜 SET VALUE값 이상이 되면 실행출력 및 보조접점이 ON 된다.
	CTD	CTD COUNT SETVALUE SET	실행조건이 ON될 때마다 COUNT를 1씩 감소시켜 0이 되면 실행출력 및 보조접점이 ON 된다
	CTUD	CTUD COUNT CU CD SETVALUE RESET	COUNT값이 CU접점이 ON될 때 1 증가, CD접점이 ON될 때 1 감소되며 SET VALUE 값 이상되면 실행출력 및 보조접점이 ON된다
	HCNT	HCNT CHANNEL OUT	고속카운터 입력 채널의 카운트 값을 OUT에 저장한다
	HRST	HRST CH	고속카운터 입력 채널의 카운트 값을 0으로 초기화 한다
	ENCO	ENCO CHANNEL OUT	엔코더 CHANNEL의 값을 OUT에 저장
	ENMOV	ENMOV CHANNEL IN	엔코더 CHANNEL의 값을 IN 값으로 설정

타이머 Timer	TON	TON TIMER SET VALUE	실행조건이 ON되었을 때, 타이머가 동작하여 SET VALUE 시간 후에 실행출력 및 보조접점을 ON시킨다
	TOFF	TOFF TIMER SET VALUE	실행조건이 OFF되었을 때, 타이머가 동작하여 SET VALUE 시간 후에 실행출력 및 보조접점을 OFF시킨다
	TPL	TPL TIMER SET VALUE	실행조건이 ON되었을 때, 타이머가 동작하여 SET VALUE 시간후 또는 실행조건이 OFF되었을 때 실행출력 및 보조점점을 OFF시킨다.
	TMON	TMON TIMER SET VALUE	실행조건이 ON되었을 때, 타이머가 동작하여 SET VALUE 시간후에 실행출력 및 보조점점을 OFF시킨다.
	TMR	TMR TIMER SET VALUE	실행조건이 ON되었을 때, 타이머가 동작하여 SET VALUE 시간후에 실행출력 및 보조접점을 ON시킨다. (실행조건이 OFF되어도 타이머는 0으로 초기화 되지 않고 유지되며 RESET 코일을 이용하여 초기화할 수 있다.

로직 컨트롤 Logic Control	JMP	JMP LABEL	실행조건이 ON되면 LABEL 명으로 점프한다
	JMPN	JMPN LABEL	실행조건이 OFF되면 LABEL 명으로 점프한다
	LABEL	LABEL LABEL	LABEL명 위치를 정의한다

데이터 컨트롤 Data Control	MAX	MAX IN OUT N	IN부터 N개의 데이터에서 최대값을 OUT에 저장
	MIN	MIN IN OUT N	IN부터 N개의 데이터에서 최소값을 OUT에 저장
	SUM	SUM IN OUT N	IN부터 N개의 데이터의 합을 OUT에 저장
	AVE	AVE IN OUT N	IN부터 N개의 데이터의 평균값을 OUT에 저장
	MUX	MUX IN OUT N	IN부터 N번째의 데이터의 값을 OUT에 저장

프로그램 컨트롤 Program Control	MCS	MCS NUM	실행조건이 ON되었을 때에만 NUM이 일치하는 MCSCLR 펑션블럭까지 COIL 출력제어권을 ON/OFF 시킨다
	MCSCLR	MCSCLR NUM	MCS와 한쌍으로 사용한다

워드로직 Word Logic	WAND	WAND IN1 IN2 OUT	IN1과 IN2를 AND하여 OUT에 저장
	WOR	WOR IN1 IN2 OUT	IN1과 IN2를 OR하여 OUT에 저장
	WXOR	WXOR IN1 IN2 OUT	IN1과 IN2를 XOR하여 OUT에 저장

아날로그 Analog	ADC	ADC CHANNEL MAX VALUE MIN VALUE OUT	CHANNEL포트로 입력된 전기적인 신호를 MIN~MAX의 범위로 SCALE하여 OUT에 저장
	ADC2	ADC2 CHANNEL MAX VALUE MIN VALUE OUT	4~20mA, 1~5V측정용 ADC펑션
	DAC	DAC CHANNEL MAX VALUE MIN VALUE OUT	MIN~MAX 범위에서 OUT을 0~65535 범위로 SCALE하여 CHANNEL포트로 전기적인 신호로 출력
	NTEMP	NTEMP CHANNEL OUT	CHANNEL포트에 연결된 NTC(서미스터) 온도센서의 저항값을 읽어 OUT에 저장 (253의 값은 25.3을 의미한다)
	PWM	PWM PORT DUTY WIDTH	PORT(PWM출력포트)에 WIDTH 폭중 DUTY폭만큼 HIGH가 되는 펄스를 출력한다
	FPWM	FPWM PORT FREQ	PORT(PWM출력포트)에 FREQ 주파수의 펄스를 출력한다
	FDPWM	FDPWM PORT FREQ DUTY	PORT(PWM출력포트)에 FREQ 주파수, DUTY(0~100%)로 펄스를 출력한다
	NPWM	NPWM PORT FREQ N	PORT(PWM출력포트)에 FREQ 주파수로 N개의 펄스를 출력한다
	POFF	POFF PORT	PORT(PWM출력포트)의 PULSE 출력을 정지시킨다
	PSR	PSR OUT	PWM 출력상태를 OUT에 저장한다.
	PID	PID PV SV KP KI KD OUT MAX OUT MIN MV	PID 알고리즘으로 MV값을 조정한다.
	PID_PWM	PID_PWM PORT PV SV KP KI KD MV	PID 알고리즘으로 PWM 펄스 출력포트에 MV값만큼 펄스를 출력한다.

이동 MOVE	MOV	MOV IN OUT	IN을 OUT으로 복사한다
	GMOV	GMOV IN OUT N	IN부터 N개의 데이터를 OUT으로부터 N개로 복사
	FMOV	FMOV IN OUT N	IN을 OUT으로부터 N개에 복사

쉬프트&회전 Shift&Rotate	SHL	SHL IN N OUT	IN을 N번 왼쪽으로 비트를 쉬프트 한다
	SHR	SHR IN N OUT	IN을 N번 오른쪽으로 비트를 쉬프트 한다

시리얼포트 Serialport	PUT	PUT CHANNEL DATA LENGTH	CHANNEL 통신포트로 DATA를 LENGTH 바이트 수만큼 전송한다
	GET	GET CHANNEL LENGTH OUT	CHANNEL 통신포트로부터 수신받은 데이터를 LENGTH 바이트 수만큼 OUT에 저장한다
	PUTLEN	PUTLEN CHANNEL OUT	CHANNEL 통신포트의 송신버퍼에 기록되어 있는 바이트의 수를 OUT에 저장한다
	GETLEN	GETLEN CHANNEL OUT	CHANNEL 통신포트의 수신버퍼에 기록되어 있는 방이트의 수를 OUT에 저장한다

I2C	ISTA	ISTA ADDRESS READ	I2C를 시작하기 위한 펑션으로서 ADDRESS 디바이스와 연결을 시작한다. 이때, READ가 ON이면 수신모드로 OFF이면 송신모드로 시작된다
	ISTP	ISTP	I2C 통신을 정지한다
	IWR	IWR DATA	DATA(바이트)를 전송한다
	IRD	IRD OUT ACK	수신한 DATA(바이트)를 OUT에 저장하고 ACK가 ON이면 ACK신호를 전송한다
	ISR	ISR OUT	I2C 통신의 상태값을 OUT에 저장한다

IOT	IOT_STA	IOT_STA SERIAL	원격감시 및 제어를 위한 IOT 솔루션을 시작한다 (추후 서비스 예정)

DS3231(RTC)	DS3231_WR	DS3231_WR YEAR MONTH DATE DAY HOUR MINUTE SECOND	DS3231 모듈에 시간을 설장한다
	DS3231_RD	DS3231_RD YEAR MONTH DATE DAY HOUR MINUTE SECOND	DS3231 모듈에서 시간을 읽어온다

케릭터LCD Character LCD (CLCD)	CINI	CINI ADR TYPE	CLCD를 ADR주소로 설정한다 (TYPE은 0을 기입한다)
	CLOC	CLOC ADR X Y	ADR 주소의 CLCD의 커서를 X, Y좌표로 이동시킨다
	CSTR	CSTR ADR STRING	ADR 주소의 CLCD에 STRING을 표시한다
	CINT	CINT ADR IN DIGIT ZERO SIGN	ADR 주소의 CLCD에 IN값을 DIGIT 자릿수로 ZERO에 따라 숫자의 앞부분에 0을 함께 SIGN에 따라 숫자의 부호를 함께 표시한다
	CHEX	CHEX ADR IN DIGIT ZERO	ADR 주소의 CLCD에 IN값을 HEX로 DIGIT 자리수로 ZERO에 따라 숫자의 앞부분에 0를 함께 표시한다
	CCLS	CCLS ADR	ADR 주소의 CLCD의 화면을 클리어한다
	CCON	CCON ADR	ADR 주소의 CLCD의 커서를 표시한다
	CCOF	CCOF ADR	ADR 주소의 CLCD의 커서를 표시하지 않는다
	CLON	CLON	ADR 주소의 CLCD의 백라이트를 ON 시킨다
	CLOF	CLOF	ADR 주소의 CLCD의 백라이트를 OFF 시킨다
	CPUT	CPUT ADR ID DEFINE	ADR 주소의 CLCD의 ID 메모리에 DEFINE을 기록시킨다
	CGET	CGET ADR ID	ADR 주소의 CLCD의 ID 메모리의 DEFINE값을 읽어와서 표시한다

FND 7세그먼트	FINI	FINI CLK DIO	FND 7세그먼트를 CLK(Clock), DIO(Data) 핀으로 설정한다
	FINT	FINT IN POSITION LENGTH DOT ZERO SIGN	IN값을 POSITION에 LENGTH 길이만큼 ZERO에 따라 숫자의 앞부분에 0를 함께 SIGN에 따라 숫자의 부호를 함께 표시하고 DOT 위치에 소수점을 표시한다
	FDATA	FDATA DATA POSITION	POSITION에 DATA를 표시한다

7.1. 비교 펑션블럭

- 비교 펑션블럭은 입력 파라미터 IN1과 IN2를 비교하여 실행결과를 출력합니다.

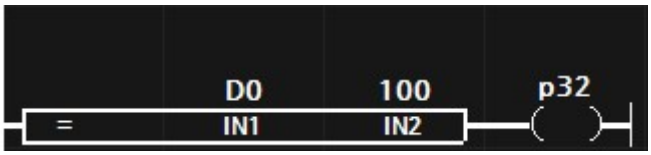
- 비교접점(비트접점)과 같은기능입니다.

펑션블럭	설 명
=	(Signed) IN1값과 IN2값이 같을 때, 실행결과 ON을 출력합니다.
<>	(Signed) IN1값과 IN2값이 다를 때, 실행결과 ON을 출력합니다.
>	(Signed) IN1값이 IN2값보다 클 때, 실행결과 ON을 출력합니다.
<	(Signed) IN1값이 IN2값보다 작을 때, 실행결과 ON을 출력합니다.
>=	(Signed) IN1값이 IN2값보다 크거나 같을 때, 실행결과 ON을 출력합니다.
<=	(Signed) IN1값이 IN2값보다 작거나 같을 때, 실행결과 ON을 출력합니다.
u>	(Unsigned) IN1값이 IN2값보다 클 때, 실행결과 ON을 출력합니다.
u<	(Unsigned) IN1값이 IN2값보다 작을 때, 실행결과 ON을 출력합니다.
u>=	(Unsigned) IN1값이 IN2값보다 크거나 같을 때, 실행결과 ON을 출력합니다.
u<=	(Unsigned) IN1값이 IN2값보다 작거나 같을 때, 실행결과 ON을 출력합니다.

- 비교 결과가 TRUE 이면, 실행출력이 ON이 되고 FALES이면 실행출력은 OFF가 됩니다.

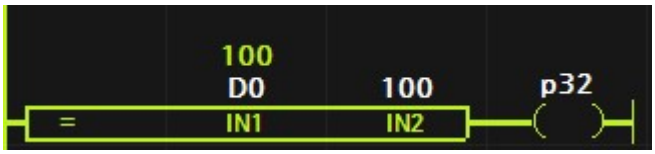
☐ **Example**

- IN1:D0(워드 메모리)값이 IN2:100과 같을 경우, P32 출력코일이 ON 되는 예제입니다. 비교 결과가 다를 경우, P32 코일은 OFF됩니다.
- 펑션블럭은 " ["키를 이용하여 생성할 수 있습니다.



<그림5-1-1 = 펑션블럭>

○ 모니터링을 하면 아래와 같이 결과값을 보실 수 있습니다.



<그림5-1-2 = 펑션블럭 모니터링>

7.2. 변환 펑션블럭

● 메모리의 값을 변환하는 펑션블럭입니다.

● 변환 펑션블럭의 종류

펑션블럭	설 명
H2D	Hexadecimal값을 Decimal값으로 변환합니다.
D2H	Decimal값을 Hexadecimal값으로 변환합니다.
INV	1의 보수값으로 변환합니다.
NEG	2의 보수값으로 변환합니다.

7.2.1. H2D (Hexadecimal To Decimal)

- Hexadecimal값을 Decimal값으로 변환합니다.

< IN >

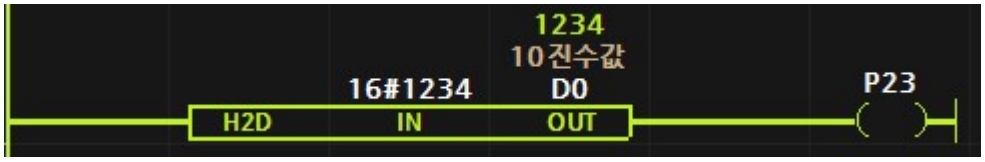
- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

▢ **Example**

- IN:16#1234 16진수를 10진수 1234로 변환하여 OUT:D0에 저장하고, 코일:P23을 ON시킵니다.



〈그림5-5-1 H2D 실행 모니터링〉

7.2.2. D2H (Decimal To (Hexadecimal)

- Decimal값을 Hexadecimal값으로 변환합니다.

< IN >

- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

☐ **Example**

- IN:1234 10진수를 16진수 16#1234로 변환하여 OUT:D1에 저장하고, 코일:P23을 ON시킵니다.

〈그림5-5-2 D2H 실행 모니터링〉

7.2.3. INV (반전 명령어)

- IN 파라미터의 모든 비트를 반전하여 OUT에 저장합니다.

< IN >

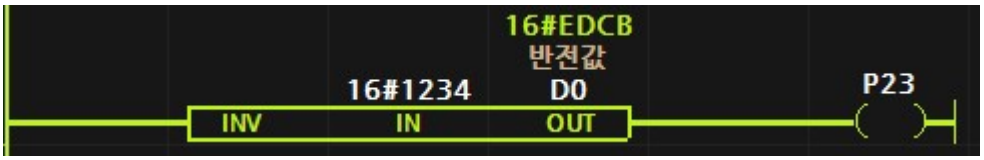
- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

☐ **Example**

- IN:16#1234 16진수를 반전하여 16#EBCD로 변환하여 OUT:D0에 저장하고, 코일:P23을 ON시킵니다.



<그림5-5-3 INV 실행 모니터링>

7.2.4. NEG (부호 반전 변환)

- IN 파라미터의 모든 비트를 반전하고 1을 더하여 OUT에 저장합니다.
- IN 파라미터의 값의 부호를 반전하여 OUT에 저장합니다.

< IN >

- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

☐ **Example**

- IN:1234의 부호를 반전시켜 D0에 저장합니다.

<그림5-5-4 NEG 실행 모니터링>

7.3. 수학 펑션블럭

◦ 수학 펑션블럭의 종류

펑션블럭	설 명
ADD	IN1값과 IN2값을 덧셈하여 OUT으로 출력합니다.
SUB	IN1값과 IN2값을 뺄셈하여 OUT으로 출력합니다.
MUL	IN1값과 IN2값을 곱셈하여 OUT으로 출력합니다.
DIV	IN1값과 IN2값을 나눗셈하여 OUT으로 출력합니다.
INC	IN 값을 1 증가시킵니다.
DEC	IN 값을 1 감소시킵니다.
SCALE	IN_MIN~IN_MAX 범위에서 IN값을 OUT_MIN~OUT_MAX 범위로 변경하여 OUT으로 출력합니다.
FLOOR	IN 실수값에서 정수값을 OUT으로 출력합니다.

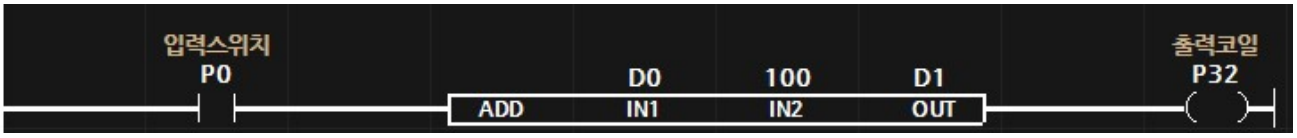
7.3.1. ADD

- IN1과 IN2를 덧셈한 결과값을 OUT에 저장합니다.

☐ **Example**

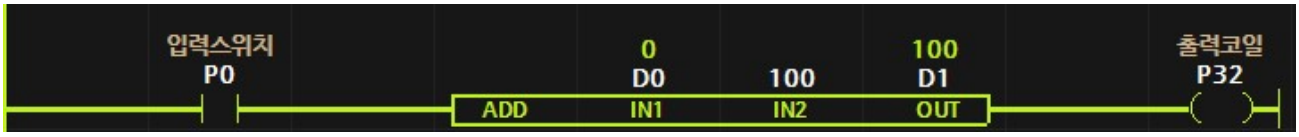
- 입력접점:P0가 ON되면, IN1:D0와 IN2:100을 덧셈하여 OUT:D1에 저장하고 P32 코일이 ON됩니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 ADD를 입력하여 생성할 수 있습니다.



<그림5-2-1 ADD 펑션블럭>

- 모니터링을 하면 아래와 같이 결과값을 보실 수 있습니다.



<그림5-2-2 ADD 펑션블럭 모니터링>

- Signed, Floating Point 형식으로 양수와 음수 및 소수점 계산이 가능합니다.

- IN1과 IN2의 파라미터의 메모리 타입은 OUT 파라미터에 의해 자동계산됩니다. OUT 파라미터가 더블워드일 경우, IN1파라미터가 워드여도 자동으로 워드를 더블워드로 치환해서 계산하게 됩니다.

☐ **Caution**

- IN1과 IN2의 메모리 타입은 OUT 메모리 타입보다 작으면 안됩니다. OUT 메모리 타입이 워드일 경우, IN1과 IN2의 메모리 타입은 워드와 더블워드를 사용할 수 있습니다. OUT 메모리 타입보다 작은 바이트를 사용할 경우 음수 계산에서 오차가 발생합니다.

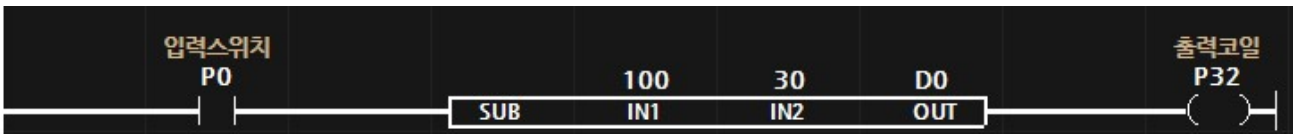
7.3.2. SUB

- IN1과 IN2를 뺄셈한 결과값을 OUT에 저장합니다.

☐ **Example**

- 입력접점:P0가 ON되면, IN1:100과 IN2:30을 뺄셈하여 OUT:D0에 저장하고 P32 코일이 ON됩니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 SUB를 입력하여 생성할 수 있습니다.



<그림5-2-3 SUB 펑션블럭>

<그림5-2-4 SUB 펑션블럭 모니터링>

- Signed, Floating Point 형식으로 양수와 음수 및 소수점 계산이 가능합니다.

- IN1과 IN2의 파라미터의 메모리 타입은 OUT 파라미터에 의해 자동계산됩니다. OUT 파라미터가 더블워드일 경우, IN1파라미터가 워드여도 자동으로 워드를 더블워드로 치환해서 계산하게 됩니다.

☐ **Caution**

- IN1과 IN2의 메모리 타입은 OUT 메모리 타입보다 작으면 안됩니다. OUT 메모리 타입이 워드일 경우, IN1과 IN2의 메모리 타입은 워드와 더블워드를 사용할 수 있습니다. OUT 메모리 타입보다 작은 바이트를 사용할 경우 음수 계산에서 오차가 발생합니다.

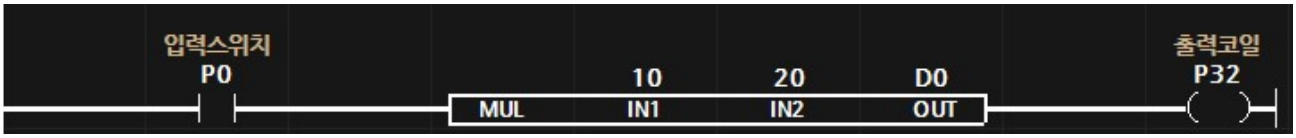
7.3.3. MUL

- IN1과 IN2를 곱셈한 결과값을 OUT에 저장합니다.

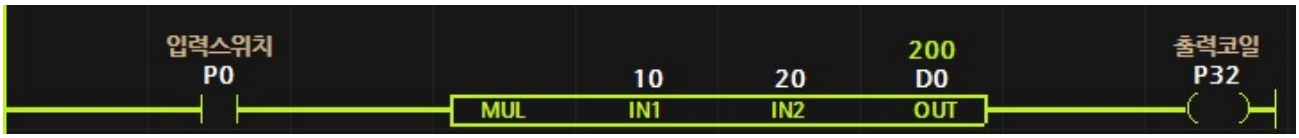
☐ **Example**

- 입력접점:P0가 ON되면, IN1:10과 IN2:20을 곱셈하여 OUT:D0에 출력되고 P32 코일이 ON됩니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 MUL을 입력하여 생성할 수 있습니다.



<그림5-2-5 MUL 펑션블럭>



<그림5-2-6 MUL 펑션블럭 모니터링>

- Signed, Floating Point 형식으로 양수와 음수 및 소수점 계산이 가능합니다.

- IN1과 IN2의 파라미터의 메모리 타입은 OUT 파라미터에 의해 자동계산됩니다. OUT 파라미터가 더블워드일 경우, IN1파라미터가 워드여도 자동으로 워드를 더블워드로 치환해서 계산하게 됩니다.

☐ **Caution**

- IN1과 IN2의 메모리 타입은 OUT 메모리 타입보다 작으면 안됩니다. OUT 메모리 타입이 워드일 경우, IN1과 IN2의 메모리 타입은 워드와 더블워드를 사용할 수 있습니다. OUT 메모리 타입보다 작은 바이트를 사용할 경우 음수 계산에서 오차가 발생합니다.

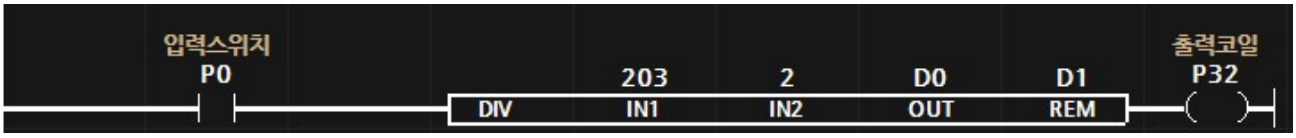
7.3.4. DIV

- IN1과 IN2를 나눗셈한 몫을 OUT에 저장하고, 나머지를 REM에 저장합니다.

☐ **Example**

- 입력접점:P0가 ON되면, IN1:203와 IN2:2를 나눗셈하여 몫을 OUT에 저장하고 나머지를 REM에 저장합니다. 그리고, P32코일이 ON됩니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 DIV를 입력하여 생성할 수 있습니다.



<그림5-2-7 DIV 펑션블럭>

<그림5-2-8 DIV 펑션블럭 모니터링>

- Signed, Floating Point 형식으로 양수와 음수 및 소수점 계산이 가능합니다.

- IN1과 IN2의 파라미터의 메모리 타입은 OUT 파라미터에 의해 자동계산됩니다. OUT 파라미터가 더블워드일 경우, IN1파라미터가 워드여도 자동으로 워드를 더블워드로 치환해서 계산하게 됩니다.

- 파라메타 중에 실수 메모리가 있으면, 실수 연산이 실행되며 실수연산에서는 나머지가 없으므로 REM 파라메타에는 나머지 값이 저장되지 않습니다.

☐ **Caution**

- IN1과 IN2의 메모리 타입은 OUT 메모리 타입보다 작으면 안됩니다. OUT 메모리 타입이 워드일 경우, IN1과 IN2의 메모리 타입은 워드와 더블워드를 사용할 수 있습니다. OUT 메모리 타입보다 작은 바이트를 사용할 경우 음수 계산에서 오차가 발생합니다.

7.3.5. INC

- IN 값을 1 증가시킵니다.

☐ **Example**

- 입력접점:P0가 ON되면, IN:D0를 1증가시킵니다.그리고, P32 코일이 ON 됩니다.

- Positive Edge접점이 없으면 INC_I 펑션블럭은 1초에 약 1000번 실행이 되어 D0는 1이 아닌 1000이 됩니다. 때문에, P0가 ON이 될 때 단 한번 INC_I 펑션블럭이 실행되도록 Positive Edge접점을 INC_I 펑션블럭 앞에 추가하였습니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 INC_I를 입력하여 생성할 수 있습니다.

<그림5-2-9 INC_I 펑션블럭 모니터링>

7.3.6. DEC_I

- IN 값을 1 감소시킵니다.

☐ **Example**

- 입력접점:P0가 ON되면, IN:D0를 1감소시킵니다.그리고, P32 코일이 ON 됩니다.

- Positive Edge접점이 없으면 DEC_I 펑션블럭은 1초에 약 1000번 실행이 되어 D0는 -1이 아닌 -1000이 됩니다. 때문에, P0가 ON이 될 때 단 한번 DEC_I 펑션블럭이 실행되도록 Positive Edge접점을 DEC_I 펑션블럭 앞에 추가하였습니다.

- 펑션블럭은 키보드의 " ["키를 누른 후 DEC_I를 입력하여 생성할 수 있습니다.

<그림5-2-10 DEC_I 펑션블럭 모니터링>

7.3.7. SCALE

● IN_MIN~IN_MAX의 범위에서 IN값을 OUT_MIN ~ OUT_MAX의 범위로 변경하여 OUT에 저장합니다.

< IN > - Signed-

● 스케일 변환할 값을 상수 또는 메모리 입력합니다.

< IN_MAX > -Signed-

● 스케일 변환할 값의 최대값을 상수 또는 메모리 입력합니다.

< IN_MIN > -Signed-

● 스케일 변환할 값의 최소값을 상수 또는 메모리 입력합니다.

< OUT > -Signed-

● 스케일 변환한 값이 저장될 메모리를 입력합니다.

< OUT_MAX > -Signed-

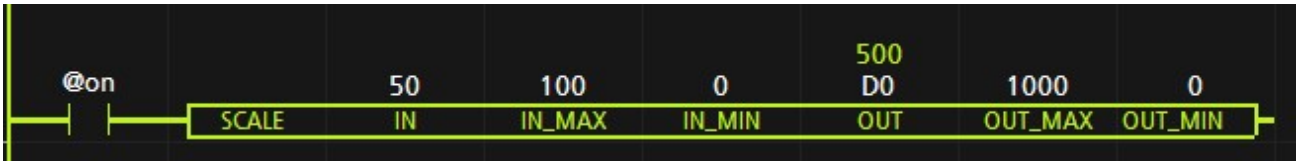
● 스케일 변환한 값의 최대값을 상수 또는 메모리 입력합니다.

< OUT_MIN > -Signed-

● 스케일 변환한 값의 최소값을 상수 또는 메모리 입력합니다.

☐ **Example**

● 0 ~ 100의 범위에서 50의 값을 0 ~ 1000의 범위로 SCALE 변환한 값인 500을 D0 워드 메로리에 저장합니다.



<그림5-3-7 SCALE 펑션블럭 모니터링>

7.3.8. FLOOR

- IN 실수값에서 정수값을 OUT에 저장합니다.

< IN > - Floating Point -

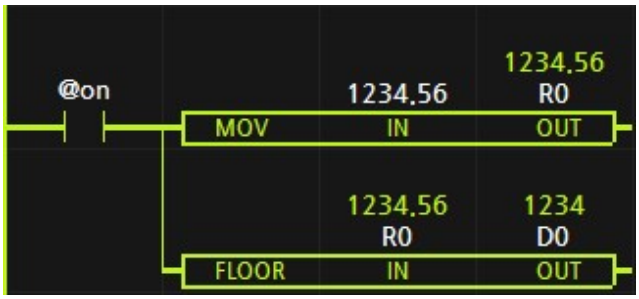
- 실수값 또는 R(실수)메모리를 입력합니다.

< OUT > -Signed-

- 실수값에서 정수값을 저장할 메모리를 입력합니다.

▢ **Example**

- 0 ~ 100의 범위에서 50의 값을 0 ~ 1000의 범위로 SCALE 변환한 값인 500을 D0 워드 메로리에 저장합니다.



< FLOOR 펑션블럭 모니터링 >

7.4. 카운터 펑션블럭

● 업카운터(CTU), 다운카운터(CTD), 업다운카운터(CTUD)는 소프트웨어적으로 카운트 하는 펑션블럭입니다.

● 고속카운터(HCNT)는 하드웨어적으로 카운트 하는 펑션블럭입니다. 10ms 이내의 고속으로 들어오는 입력신호를 카운트하기 위해서는 고속카운터를 사용해야 합니다.
고속카운터는 모든 제품에서 지원하는 기능이 아닙니다. 고속카운터를 지원하는 제품에서 사용할 수 있으며, 고속카운터는 고속카운터 포트에 입력되는 펄스의 상승엣지를 카운트하여 MCU에 저장합니다. 고속카운터 값은 HCNT 펑션블럭으로 읽고 HRST 펑션블럭으로 초기화할 수 있습니다.

● 엔코더(ENCODER)는 엔코더에서 들어오는 신호 A와 B를 사용하여 엔코더 값을 연산합니다. A가 B보다 앞서게 펄스가 입력되면 값이 상승하며 B가 A보다 앞서게 펄스가 입력되면 값이 감소합니다. 현재 ENCODER 펑션블럭은 MPA 제품군과 MPS-8A8R-S, MPS-16A16R 제품에서만 사용할 수 있습니다.

○ 카운터 펑션블럭의 종류

펑션블럭	설 명
CTU	CU 파라미터의 입력상태가 ON될 때, COUNT값이 1씩 가산됩니다. COUNT값이 SET VALUE 이상이 되면, 출력접점이 ON됩니다.
CTD	CD 파라미터의 입력상태가 ON될 때, COUNT값이 1씩 감산됩니다. COUNT값이 0이 되면, 출력접점이 ON됩니다.
CTUD	CU 파라미터의 입력상태가 ON될 때, COUNT값이 1씩 가산됩니다. CD 파라미터의 입력상태가 ON될 때, COUNT값이 1씩 감산됩니다. COUNT값이 SET VALUE 이상이 되면, 출력접점이 ON됩니다.
HCNT	고속카운터 레지스터리에 저장되어 있는 카운트값을 읽어와서 OUT 파라미터에 저장합니다.
HRST	고속카운터 레지스터리의 값을 0으로 초기화 합니다.
ENCO	엔코더의 값을 읽어옵니다.
ENMOV	엔코더에 값을 설정합니다.

7.4.1. CTU

- 실행조건이 ON될때마다, COUNT값이 1씩 가산됩니다. COUNT값이 SET VALUE값 이상이 되면, COUNT 보조비트 메모리가 ON되고, 실행조건이 ON일 때에만 실행출력이 ON됩니다.

< COUNT > - unsigned -

- 카운트 워드 메모리(C0, C1..) 또는 바이트(BC0, BC1,,), 더블워드(DC0..)을 입력합니다.
- 실행조건이 ON될때마다 1씩 가산됩니다.

< SET VALUE > - unsigned -

- 워드 또는 더블워드 메모리를 입력합니다. 업카운터의 목표값을 입력합니다.

< RESET >

- 비트 메모리를 입력합니다.
- RESET 비트가 ON되면, COUNT값이 0으로 초기화 되고, 보조비트 메모리를 OFF시킵니다.

< 보조비트 메모리 >

- COUNT에 입력한 카운트 메모리와 같은 이름으로 보조비트 메모리를 사용합니다.
- Normal Open 접점 또는 Normal Close 점점에 C0, C1.. 으로 사용하실 수 있습니다.
- COUNT값이 SET VALUE값 이상일때, 보조비트 메모리가 ON됩니다.
- RESET 비트가 ON되면, 보조비트 메모리가 OFF됩니다.

▢ **Example**

- 입력스위치:P0가 ON되면, COUNT:C0가 1씩 가산됩니다. 입력스위치:P0가 10회 ON 되는 시점에서 COUNT:C0의 카운트값이 10이 됩니다. 이때, COUNT:C0값이 SET VALUE값의 이상이 되어, 조건이 성립됩니다. 조건이 성립되고 있을 때, 보조비트 메모리:C0가 ON되어, 출럭코일2:P33이 ON됩니다. 또한, 조건입 성립되고 있을 때, CTU 펑션블럭의 실행조건이 ON일때, 실행출력이 ON되어 출력코일1:P32가 ON됩니다.

<그림5-3-2 CTU 펑션블럭 조건 미성립 모니터링>

- 위 모니터링은 입력스위치:P0가 한 번 ON되었을 때 카운트값이 1 증가하였습니다. 카운트값이 SET VALUE인 10에 도달하지 못했으므로 실행결과는 OFF, 출력코일1도 OFF됩니다. 또한, 2번째 줄에 있는 보조비트접점인 C0도 OFF되어 출력코일은 모두 OFF됩니다.

<그림5-3-3 CTU 펑션블럭 조건 성립 모니터링>

- 위 모니터링은 입력스위치:P0 ON이 10번 되어 카운트값이 10으로 증가하였을 때의 모니터링입니다. 카운트값이 SET VALUE인 10에 도달하였으므로 실행출력은 ON되어 출력코일1도 ON됩니다. 2번째 줄에 있는 보조접점인 C0도 ON되어 출력코일은 모두 ON됩니다. 이때, 입력스위치:P0가 OFF되면 출력코일1:P32는 OFF되고 출력코일2:P33은 계속 ON됩니다. 출력코일2:P33은 카운트셋 조건이 미성립되거나 카운트리셋:P1이 ON되면 OFF됩니다.

- RESET 파라미터에는 비트 메모리를 사용할 수 있으며, 사용한 비트 메모리의 상태가 ON이 되었을 때, COUNT값은 0으로 초기화 됩니다. RESET 비트가 계속 ON 상태이면 카운트는 계속 리셋이 되므로 RESET 비트는 펄스로 입력해주시기 바랍니다.

7.4.2. CTD

- 실행조건이 ON될때마다, COUNT값이 1씩 감산됩니다. COUNT값이 0으로 되면, COUNT 보조비트 메모리가가 ON되고, 실행조건이 ON일 때에만 실행출력이 ON됩니다.

< COUNT > - unsigned -

- 카운트 워드 메모리(C0, C1..) 또는 바이트(BC0, BC1,,), 더블워드(DC0..)을 입력합니다.
- 실행조건이 ON될때마다 1씩 감산됩니다.
- 카운트 메로리의 초기값은 SET VALUE값이 상수일 경우에만, SET VALUE값이 됩니다. SET VALUE가 변수일 경우에는, 초기에 SET 비트를 ON해야 합니다.

< SET VALUE > - unsigned -

- 다운카운트의 초기값을 입력합니다.

< RESET >

- 비트 메모리를 입력합니다. RESET 비트가 ON되면, COUNT값이 SET VALUE값으로 초기화 되고, 보조비트 메모리를 OFF시킵니다.

< 보조비트 메모리 >

- COUNT값이 SET VALUE값에서 1씩 감소하여, 0이 되면 보조비트 메모리가 ON됩니다.
- RESET 비트가 ON되면, 보조비트 메모리가 OFF됩니다.

☐ **Example**

- 입력접점:P0가 ON되면, COUNT:C0가 1씩 감산됩니다. P0가 10회 ON이 되는 시점에서 COUNT:C0의 카운트값이 0이 됩니다. COUNT:C0가 0이 되면, 보조 비트접점인 C0가 ON이 되어 출력코이2:P33이 ON됩니다. 또한, 조건이 성립하고 있을 때, 입력스위치:P0가 ON되어 실행조건이 ON이 될 때, 실행출력이 ON되어 출력코일1:P32가 ON됩니다.

<그림5-3-6 CTD 펑션블럭 조건 미성립 모니터링>

- 위 모니터링은 입력스위치:P0가 한 번 ON되었을 때 카운트값이 1 감소하였습니다.

카운트값이 0에 도달하지 못했으므로 보조비트접점인 카운트값:C0는 OFF되어 출력코일2:P33이 OFF됩니다. 또한, 실행출력도 OFF로 출력코일1:P32도 OFF됩니다.

<그림5-3-7 CTD 펑션블럭 조건 성립 모니터링>

- 위 모니터링은 입력스위치:P0가 10번 ON되어, COUNT:C0값이 0이 되었습니다. COUNT:C0값이 0이 되면, 실행출력이 ON되고 보조비트 메모리 C0가 ON되어 출력코일2:P33이 ON됩니다. CTD 펑션블럭의 출력상태는 펑션블럭의 입력상태가 ON되고 조건이 성립되고 있을 때 ON됩니다. 따라서, 조건이 성립되고 입력스위치:P0가 ON되고 있을 때, 출력코일1:P32가 ON됩니다.

7.4.3. CTUD

- 실행조건이 ON되고, CU 파라미터의 비트가 ON될 때 COUNT값이 1씩 가산됩니다.
- 실행조건이 ON되고, CD 파라미터의 비트가 ON될 때 COUNT값이 1씩 감산됩니다.
- COUNT값이 SET VALUE값 이상이 되면 실행출력이 ON되고 보조접점이 ON됩니다.

< COUNT > - unsigned -

- 카운트 워드 메모리(C0, C1..) 또는 바이트(BC0, BC1,,), 더블워드(DC0..)을 입력합니다.
- COUNT값이 SET VALUE값 이상이 되면 실행출력이 ON되고 보조접점이 ON됩니다.
- 실행조건이 OFF되면 COUNT값은 0으로 초기화 됩니다.

< CU > - Bit -

- 실행조건이 ON되고, CU 파라미터의 비트가 ON될 때 COUNT값이 1씩 가산됩니다.
- COUNT값은 SET VALUE값을 초과해도 CU 입력에 의해 1씩 가산됩니다.
- COUNT값이 최대가 되면 더 이상 가산하지 않습니다. (Ex. Word는 65535 최대값)

< CD > - Bit -

- 실행조건이 ON되고, CD 파라미터의 비트가 ON될 때 COUNT값이 1씩 감산됩니다.
- COUNT값이 0가 되면 더 이상 감산하지 않습니다.

< SET VALUE > - unsigned -

- 카운트 설정값을 상수 또는 메모리(바이트, 워드, 더블워드)를 입력합니다.
- COUNT값이 SET VALUE값 이상이 되면 실행출력이 ON되고 보조접점이 ON됩니다.

< RESET > - Bit -

- 비트 메모리(P0, M0 등)를 입력합니다.
- RESET 비트가 ON되면, COUNT값이 0으로 초기화 되고, 보조비트 메모리를 OFF시킵니다.

< 보조비트 메모리 >

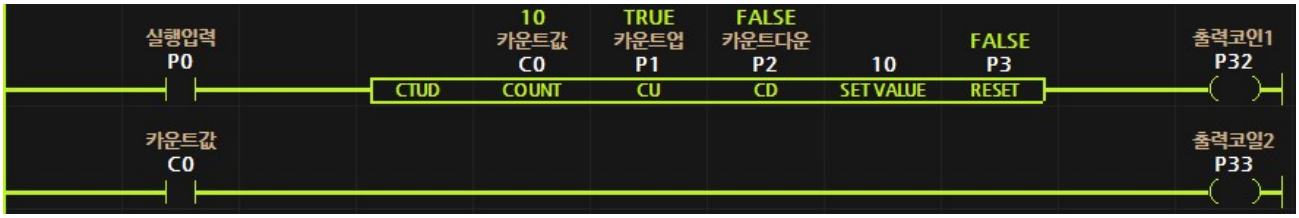
- COUNT값이 SET VALUE값 이상이 되면, 보조비트 메모리가 ON됩니다.
- 보조비트 메모리는 COUNT 파라미터의 메모리 이름을 그대로 사용해야 합니다.
  Ex) C0(워드), BC1(바이트), DC4(더블워드)
- RESET 비트가 ON되거나 실행조건이 OFF되면 보조비트 메모리가 OFF됩니다.

▢ **Example**

● 실행입력:P0가 ON상태에서, 카운트업:P1가 ON될 때 마다 COUNT:C0가 1씩 가산됩니다. P1가 10회 ON이 되는 시점에서 COUNT:C0의 카운트값이 10이 됩니다. COUNT:C0가 10이 되면, 실행출력이 ON되어 출력코일1:P32가 ON되고 보조 비트접점인 C0가 ON 되어 출력코일2:P33이 ON됩니다. 이때, 실행입력:P0가 OFF되면 실행출력 및 보조비트접점이 OFF되어 출력코일은 모두 OFF되고 카운트값:C0는 0으로 초기화 됩니다.

<그림5-3-3-1 CTUD 펑션블럭 조건 미성립 모니터링>

● 위 모니터링은 카운트업:P1가 한 번 ON되어 카운트값:C0가 1 증가하였습니다. 카운트값이 SET VALUE:10에 도달하지 못했으므로 출력상태 및 보조비트접점인 카운트값:C0는 OFF되어 출력코일은 모두 OFF됩니다.



<그림5-3-3-2 CTD 펑션블럭 조건 성립 모니터링>

● 위 모니터링은 카운트업:P1가 10번 ON되어, 카운트값:C0가 10이 되었습니다. COUNT값이 SET VALUE:10이 되면, 실행출력이 ON되어 출력코일1:P32가 ON되고 보조비트 메모리 C0가 ON되어 출력코일2:P33이 ON됩니다.

< Caution >

● 실행조건이 OFF되면 COUNT, 보조비트접점, 실행출력, CU 및 CD의 상태기억 메모리 모두가 OFF됩니다.

7.4.4. HCNT

● 고속카운터 레지스터리에 저장되어 있는 카운트된 값을 읽어와서 OUT에 저장합니다.

● 고속카운터 포트에 입력되는 펄스의 상승엣지를 카운트하여 각 채널의 고속카운터 레지스터리에 카운트 값이 저장됩니다.

● 레지스터리에 저장되어 있는 고속카운터값을 HCNT 펑션블럭을 사용하여 D0(워드 메모리), DD0(워드 메모리) 등의 메모리에 저장하여 사용합니다.

● HRST 펑션블럭으로 레지스터리에 저장되어 있는 카운트값을 0으로 초기화할 수 있습니다.

● 고속카운터는 32비트(0~4,294,967,295)의 범위의 카운트 범위를 사용합니다.

● 고속카운터는 프로그램에서 카운트하기 어려운 빠른 속도로 입력되는 펄스를 카운트하기 위해 사용하는 기능입니다. MP STUDIO에서 작상한 LADDER LOGIC의 프로그램과 별개로 하드웨어적으로 카운트 되어 카운터 레지스터리에 저장됩니다.

< CHANNEL > - Unsigned -

● 고속카운터의 채널번호를 입력합니다. 제품의 HCNT0은 채널0, HCNT1은 채널1 입니다.

< OUT > - Unsigned -

● 카운트된 값을 저장할 워드 또는 더블워드 메모리를 입력합니다.

▢ **Example**

● 입력스위치:P00가 ON이 되면, 고속카운터 CHANNEL:0번 채널에서 카운트된 값을 읽어와서 OUT:D0에 저장합니다. 실행조건이 그대로 실행출력되어 출력코일1:P20은 ON됩니다.

<그림5-3-10 HCNT 펑션블럭 모니터링>

7.4.5. HRST

● 고속카운터 레지스터리에 저장되어 있는 카운트 값을 0으로 초기화합니다.

● 고속카운터는 제품의 고속카운터 포트에 입력되는 펄스를 카운트하여 저장하고 있습니다.

● 고속카운터는 프로그램에서 카운트하기 어려운 빠른 속도로 입력되는 펄스를 카운트하기 위해 사용하는 기능입니다. MP STUDIO에서 작상한 LADDER LOGIC의 프로그램과 별개로 하드웨어적으로 카운트 되어 카운터 레지스터리에 저장됩니다.

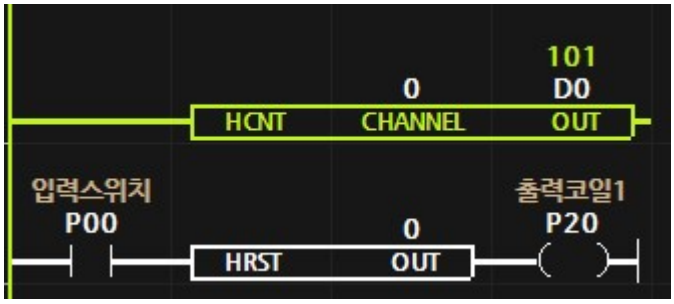
● HCNT 펑션블럭으로 특수 메모리에 저장되어 있는 카운트값을 읽어올 수 있습니다.

< CHANNEL >

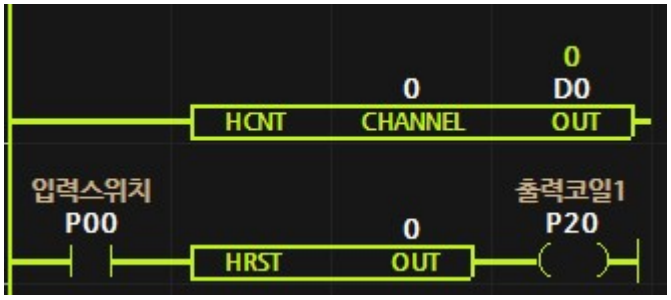
● 고속카운터의 채널번호를 입력합니다. 제품의 HCNT0은 0채널, HCNT1은 1채널입니다.

☐ **Example**

● 입력스위치:P00가 ON이 되면, 고속카운터 CHANNEL:0번 채널에 저장되어 있는 카운트값을 0으로 초기화 합니다.



<그림5-3-11 HRST 실행 전 모니터링>



<그림5-3-12 HRST 실행 후 모니터링>

7.4.6. ENCODER

▢ 엔코더 CHANNEL의 엔코더 값을 읽어와서 OUT에 저장합니다.

▢ 엔코더 포트에 엔코더의 A, B상을 연결하여 엔코더의 값을 32비트(-2,147,483,648 ~ 2,147,483,648) 범위에서 사용할 수 있습니다.

▢ 현재 MPA제품군 및 MPS-8A8R-S, MPS-16A16R 제품에서만 지원합니다.
- ◦ MPA 제품군에서 2개의 앱솔루트형 엔코더를 사용할 수 있습니다.
  - 채널0은 P0를 A펄스, P2를 B펄스의 입력으로 사용합니다.
  - 채널1은 P1를 A펄스, P3를 B펄스의 입력으로 사용합니다.
- ◦ MPS-8A8R-S 제품에서 1개의 앱솔루트형 엔코더를 사용할 수 있습니다.
- ◦ MPS-16A16R 제품에서 1개의 앱솔루트형 엔코더를 사용할 수 있습니다.

▢ 두 개의 펄스(A, B)를 사용하는 엔코더를 연결하여 엔코더값을 확인할 수 있습니다.
- ◦ 두 개의 펄스(A, B)중 A가 B보다 빠르게 입력될 경우 정방향(CW)이므로 엔코더의 값은 입력된 펄스의 수만큼 증가합니다.
- ◦ 두 개의 펄스(A, B)중 A가 B보다 느리게 입력될 경우 역방향(CCW)이므로 엔코더의 값은 입력된 펄스의 수만큼 감소합니다.

▢ 하드웨어 적인 연결방법은 각 제품의 사용설명서를 참조해 주시기 바랍니다.

< CHANNEL >
- ◦ 엔코더 채널번호를 입력합니다.

< OUT > - Signed -
- ◦ 엔코더의 값을 저장할 메모리를 입력합니다.

▢ **Example**
- ◦ CHANNEL:0번 채널에 저장되어 있는 엔코더 값을 D0 워드메모리에 저장합니다.

〈 ENCO 펑션블럭 모니터링〉

7.4.7. ENMOV

▢ 엔코더 CHANNEL의 엔코더 값을 IN값으로 설정합니다.

▢ 엔코더 포트에 엔코더의 A, B상을 연결하여 엔코더의 값을 32비트(-2,147,483,648 ~ 2,147,483,648) 범위에서 사용할 수 있습니다.

< CHANNEL >

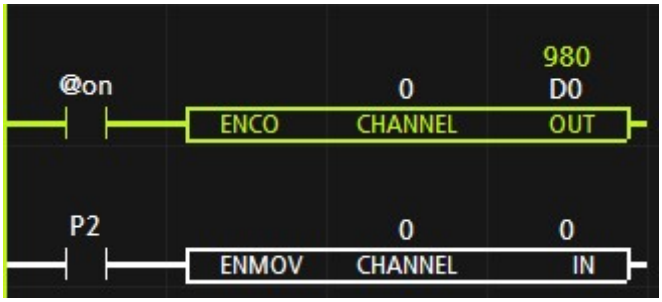
- 엔코더 채널번호를 입력합니다.

< IN > - Signed -

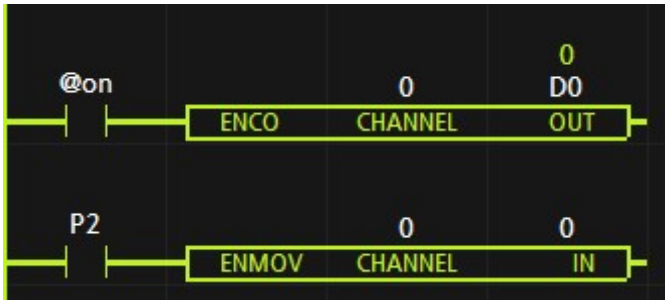
- 변경하고자 하는 값 또는 메모리를 입력합니다.

▢ **Example**

- 엔코더 CHANNEL:0번 채널에 저장되어 있는 엔코더값을 D0 워드메모리에 저장합니다.
- P2가 ON되면 엔코더 CHANNEL:0번 채널의 엔코더 값을 0으로 설정합니다.



< ENMOV 펑션블럭 실행 전 모니터링>



< ENMOV 펑션블럭 실행 후 모니터링>

7.5. 타이머 펑션블럭

● 동작지연, 동장유지, 일정시간동안만 동작 등의 기능을 타이머 펑션블럭을 이용하여 쉽게 프로그램할 수 있습니다.

● 타이머 펑션블럭의 종류

펑션블럭	설 명
TON	입력상태가 ON 되고부터 PT 파라미터 시간동안 Delay 후, 보조비트 메모리 및 출력상태를 ON 시킵니다.
TOFF	입력상태가 ON -> OFF되고부터 PT 파라미터 시간동안 보조비트 메모리 및 출력상태를 ON 되도록 Delay 시킵니다.
TPL	입력상태가 ON이 계속 되고 있어도 보조비트 메모리 및 출력상태를 PT 파라미터 시간동안만 ON시키고 이후, OFF시킵니다. PT 파라미터 시간에 도달하지 않은 상태에서 입력상태가 OFF되면, 보조비트 메모리 및 출력상태를 OFF 시킵니다.
TMON	입력상태가 ON이 되면, 보조비트 메모리 및 출력상태를 PT 파라미터 시간동안 ON시키고 이후, OFF 시킵니다. PT 파라미터 시간에 도달하지 않은 상태에서 입력상태가 OFF되어도 보조비트 메모리 및 출력상태를 PT 시간동안 ON시키고 이후, OFF 시킵니다.

● TIMER 기본 단위
TIMER의 기본 단위는 10ms로서, 10ms마다 1씩 증가합니다.

● TIMER 파라미터는 T 메모리를 사용해야 합니다.
워드 메모리 : T0, T1, T2 ....
더블워드 메모리 : DT0, DT1, DT2....

● TIMER 시간범위
워드 메모리 : 10ms ~ 10분
더블워드 메모리 : 10ms ~ 5965 시간

7.5.1. TON (On Delay Timer)

- SET VALUE 시간만큼 동작지연을 합니다. SET VALUE 시간에 도달하면, 타이머 보조비트 메모리 및 출력상태가 ON됩니다.

< TIMER >

- 타이머 워드 메모리(T0, T1..)와 더블워드 메모리(DT0, DT1...)를 사용합니다.
- 워드 메모리는 10분까지 설정이 가능합니다.
- 더블워드 메모리는 5965시간까지 가능합니다.
- 0부터 SET VALUE까지 10ms마다 1씩 상향 카운트 됩니다.

< SET VALUE > - Unsigned -

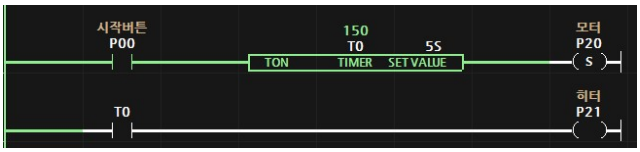
- 워드 및 더블워드를 입력합니다.
- SET VALUE의 값은 1이 10ms를 의미합니다. 즉, 100은 1초입니다.
- SET VALUE에 단위를 기입하여 사용할 수 있습니다. Ex) 20ms , 2S, 3M, 1H
- SET VALUE의 시간은 10ms 미만을 사용할 수 없습니다.

< 보조비트 메모리 >

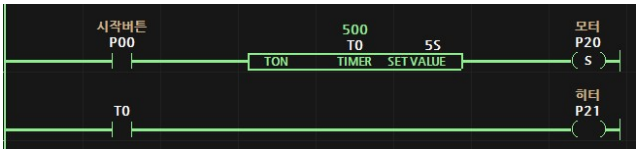
- SET VALUE의 시간만큼 동작지연을 한 이후, 타이머 보조비트 메모리가 ON됩니다.
- T0의 타이머 보조비트 메모리는 T0이고, DT0의 타이머 보조비트 메모리는 DT0입니다.

☐ **Example**

- 시작버튼:P00가 5초동안 ON되면, 모터:P20, 히터:P21이 ON됩니다.



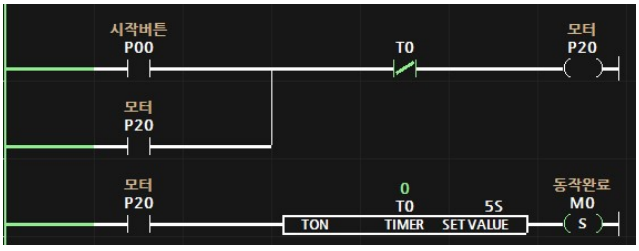
〈그림5-4-1 TON 타이머 진행중 모니터링〉



〈그림5-4-2 TON 완료 모니터링〉

- 시작버튼:P00가 ON이 되면, 모터:P20이 ON됩니다. 모터:P20이 ON되면, TON 펑션블럭의 입력상태가 ON되어 PT:5S (5초)후에 T0:보조비트 메모리가 ON되어 모터:P20은 OFF되고 TON 펑션블럭의 출력상태가 ON되어 동작완료:M0가 SET됩니다.

〈그림5-4-3 TON 타이머 진행중 모니터링〉



〈그림5-4-4 TON 완료 모니터링〉

7.5.2. TOFF (OFF Delay Timer)

- SET VALUE 시간만큼 동작연장을 합니다. 입력상태가 ON되면 출력상태 및 타이머 보조비트 메모리를 ON시킵니다. 이후, 입력상태가 OFF되도 SET VALUE 시간동안 출력상태 및 보조비트 메모리의 ON을 유지시키다 SET VALUE시간에 도달하면 OFF시킵니다.

< TIMER >

- 타이머 워드 메모리(T0, T1..)와 더블워드 메모리(DT0, DT1...)를 사용합니다.
- 워드 메모리는 10분까지 설정이 가능합니다.
- 더블워드 메모리는 5965시간까지 가능합니다.
- 10ms마다 SET VALUE에서 1씩 하향 카운트 됩니다.

< SET VALUE > - Unsigned -

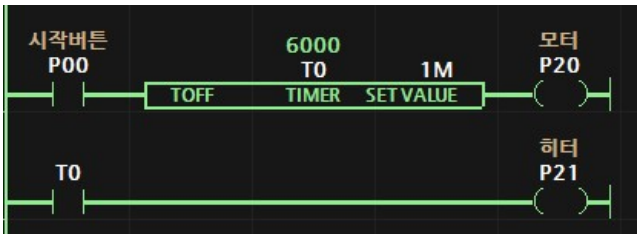
- 워드 및 더블워드를 입력합니다.
- SET VALUE의 값은 1이 10ms를 의미합니다. 즉, 100은 1초입니다.
- SET VALUE에 단위를 기입하여 사용할 수 있습니다. Ex) 20ms , 2S, 3M, 1H
- SET VALUE의 시간은 10ms 미만을 사용할 수 없습니다.

< 보조비트 메모리 >

- 입력상태가 ON될 때, 같이 ON이 됩니다. 이후, 입력상태가 OFF되면 SET VALUE 시간만큼 ON시간을 연장한 이후 OFF됩니다.
- T0의 타이머 보조비트 메모리는 T0이고, DT0의 타이머 보조비트 메모리는 DT0입니다.

☐ **Example**

- 시작버튼:P00이 ON되면, 모터:P20 및 히터:P21이 ON됩니다. 이후, 시작버튼:P00가 OFF되면, SET VALUE:1M (1분) 후에 모터:P20 및 히터:P21가 OFF됩니다.



<그림5-4-5 TOFF 타이머 입력상태 ON 모니터링>

<그림5-4-6 TOFF 입력상태 OFF 모니터링>

7.5.3. TPL (Pulse Timer)

- 펑션블럭의 입력상태가 ON되면 출력상태 및 타이머 보조비트 메모리를 SET VALUE 시간동안만 ON시킵니다. SET VALUE시간에 도달하기 전에 펑션블럭의 입력상태가 OFF되면 출력상태 및 보조비트 메모리는 OFF됩니다.

< TIMER >

- 타이머 워드 메모리(T0, T1..)와 더블워드 메모리(DT0, DT1...)를 사용합니다.
- 워드 메모리는 10분까지 설정이 가능합니다.
- 더블워드 메모리는 5965시간까지 가능합니다.
- 10ms마다 SET VALUE에서 1씩 상향 카운트 됩니다.

< SET VALUE > - Unsigned -

- 워드 및 더블워드를 입력합니다.
- SET VALUE의 값은 1이 10ms를 의미합니다. 즉, 100은 1초입니다.
- SET VALUE에 단위를 기입하여 사용할 수 있습니다. Ex) 20ms , 2S, 3M, 1H
- SET VALUE의 시간은 10ms 미만을 사용할 수 없습니다.

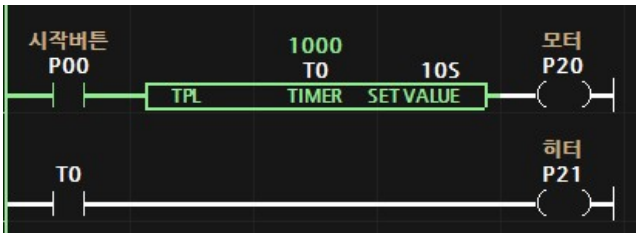
< 보조비트 메모리 >

- 입력상태가 ON될 때, 같이 ON이 됩니다. 이후, 입력상태가 OFF되거나 SET VALUE 시간에 도달하면 OFF됩니다.
- T0의 타이머 보조비트 메모리는 T0이고, DT0의 타이머 보조비트 메모리는 DT0입니다.

☐ **Example**

- 시작버튼:P00이 ON되면, 모터:P20 및 히터:P21이 ON됩니다. 이후, SET VALUE:10S에 도달하면 시작버튼:P00가 ON이 되고 있어도 모터:P20 및 히터:P21가 OFF됩니다.

<그림5-4-7 TPL 타이머 진행중 모니터링>



<그림5-4-8 TPL 타이머 진행완료 모니터링>

7.5.4. TMON (Monostable Timer)

- 펑션블럭의 입력상태가 ON되면 출력상태 및 타이머 보조비트 메모리를 SET VALUE 시간동안만 ON시킵니다. SET VALUE시간에 도달하기 전에 펑션블럭의 입력상태가 OFF되도 출력상태 및 보조비트 메모리는 SET VALUE시간 후에 OFF됩니다.

< TIMER >

- 타이머 워드 메모리(T0, T1..)와 더블워드 메모리(DT0, DT1...)를 사용합니다.
- 워드 메모리는 10분까지 설정이 가능합니다.
- 더블워드 메모리는 5965시간까지 가능합니다.
- 10ms마다 SET VALUE에서 1씩 상향 카운트 됩니다.

< SET VALUE > - Unsigned -

- 워드 및 더블워드를 입력합니다.
- SET VALUE의 값은 1이 10ms를 의미합니다. 즉, 100은 1초입니다.
- SET VALUE에 단위를 기입하여 사용할 수 있습니다. Ex) 20ms , 2S, 3M, 1H
- SET VALUE의 시간은 10ms 미만을 사용할 수 없습니다.

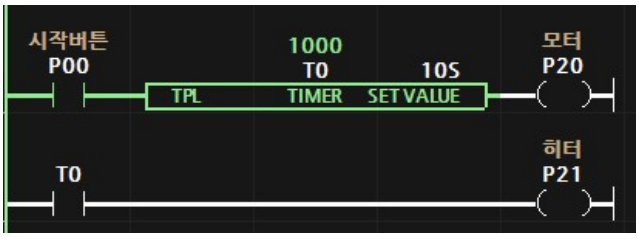
< 보조비트 메모리 >

- 입력상태가 ON될 때, 같이 ON이 됩니다. 이후, 입력상태가 OFF되거나 SET VALUE 시간에 도달하면 OFF됩니다.
- T0의 타이머 보조비트 메모리는 T0이고, DT0의 타이머 보조비트 메모리는 DT0입니다.

☐ **Example**

- 시작버튼:P00이 ON되면, 모터:P20 및 히터:P21이 ON됩니다. 이후, SET VALUE:10S에 도달하면 시작버튼:P00가 ON이 되고 있어도 모터:P20 및 히터:P21가 OFF됩니다.

<그림5-4-7 TPL 타이머 진행중 모니터링>



<그림5-4-8 TPL 타이머 진행완료 모니터링>

7.5.5. TMR (Accumulative Timer)

▢ TIMER의 값이 SET VALUE 값보다 크거나 같아지면 실행출력이 ON되고 TIMER의 보조접점 메모리가 ON됩니다.

▢ 실행조건이 ON되면 TIMER의 값이 10ms마다 1씩 증가합니다. TON 타이머와 다르게 실행조건이 OFF되어도 TIMER의 값은 0으로 초기화되지 않고 유지됩니다. 이후, 다시 실행조건이 ON되면 TIMER의 값은 10ms마다 1씩 증가합니다.

▢ 리셋 코일접점을 사용하여 TIMER값을 0으로 초기화할 수 있습니다.

< TIMER > -Unsigned-

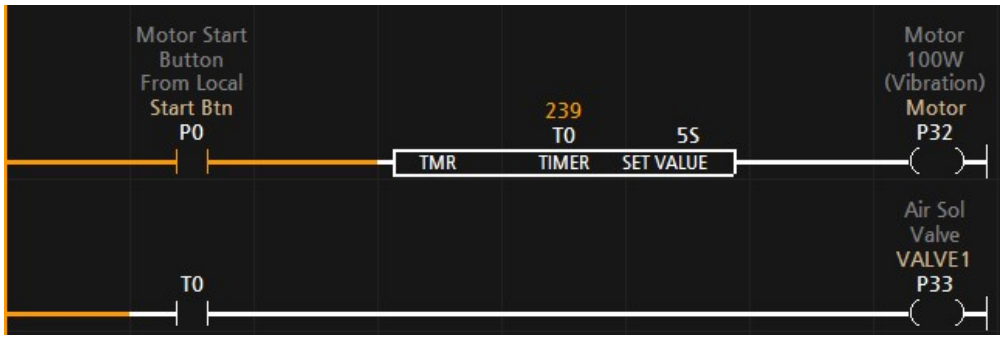
- 타이머 메모리를 입력합니다.

< SET VALUE > -Unsigned-

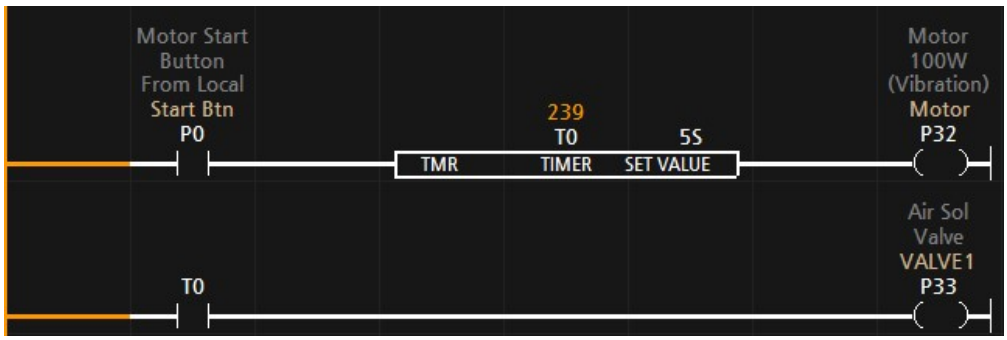
- 설정하고자 하는 시간 또는 메모리를 입력합니다.

▢ Example

- P0:Start Btn이 ON되고 있는 동안 TMR 타이머가 10ms마다 1씩 증가합니다.

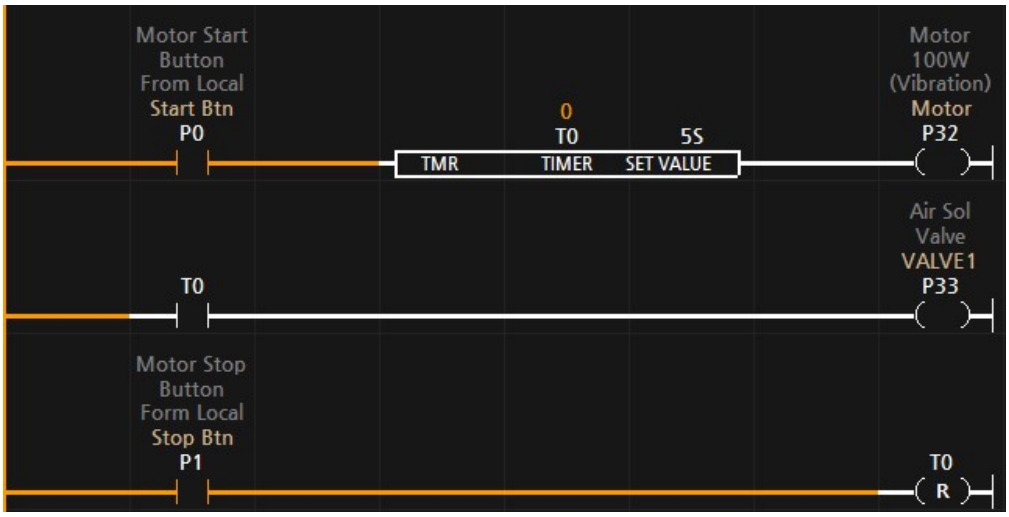


- P0:Start Btn이 OFF되어도 TIMER의 값은 0으로 초기화 되지 않으며 더 이상 증가하지 않습니다.



◦ P0:Start Btn이 다시 ON되면 TIMER의 값은 현재 값에서 다시 10ms마다 1씩 증가하여 SET VALUE의 값보다 크거나 같아지면 실행출력이 ON되고 보조접점도 ON됩니다. 이때, TMR 타이머 펑션블럭의 실행조건이 OFF라도 실행출력 및 보조점점은 계속 ON됩니다.

◦ P1:Stop Btn이 ON되면 리셋코일:T0에 의해 TIMER의 값은 0으로 초기화되고 TMR 타이머의 실행출력 및 보조접점은 OFF됩니다.



7.6. 로직 컨트롤 펑션블럭

- 조건에 따른 프로그램의 실행을 컨트롤할 수 있는 펑션블럭입니다.

- 로직 컨트롤 펑션블럭 종류

펑션블럭	설 명
JMP	입력상태가 ON일때, LABEL로 점프하여 프로그램을 실행합니다.
JMPN	입력상태가 OFF일때, LABEL로 점프하여 프로그램을 실행합니다.
LABEL	JMP와 JMPN에서 점프할 대상인 LABEL입니다.

7.6.1. JMP

- 입력상태가 ON일 때, LABEL로 점프하여 프로그램을 실행합니다.
- LABEL 펑션블럭과 함께 사용해야 합니다.

< LABEL >

- 점프할 LABEL을 입력합니다.

☐ **Example**

- 자동버튼:P00가 OFF일 경우, 자동동작조건:P01과 관계없이 수동모터ON:M0에 의해 모터:P20이 ON/OFF됩니다.
- 자동버튼:P00가 ON일 경우, 자동동작조건:P01에 의해 모터:P20이 ON/OFF됩니다. 3번째줄은 JMP펑션블럭에 의해 점프되기 때문에 동작하지 않습니다.

〈그림5-6-1 JMP 입력상태 OFF 모니터링〉

〈그림5-6-1 JMP 입력상태 ON 모니터링〉

- JMP 펑션블럭의 입력상태와 관계없이, 펑션블럭의 뒤쪽 프로그램은 실행되지 않습니다.

7.6.2. JMPN

- 입력상태가 OFF일 때, LABEL로 점프하여 프로그램을 실행합니다.
- LABEL 펑션블럭과 함께 사용해야 합니다.

< LABEL >

- 점프할 LABEL을 입력합니다.

☐ **Example**

- 수동버튼:P00가 ON일 경우, JMPN 펑션블럭의 입력상태가 ON이므로 3번째 줄을 점프하지 않습니다. 때문에, 자동동작조건:P01과 관계없이 수동모터ON:M0에 의해 모터:P20이 ON/OFF됩니다.

- 자동버튼:P00가 OFF일 경우, JMPN펑션블럭의 입력상태가 OFF되므로 4번째 줄로 점프하여 3번째 줄을 실행하지 않습니다. 때문에, 자동동작조건:P01에 의해 모터:P20이 ON/OFF됩니다.

<그림5-6-3 JMPN 입력상태 ON 모니터링>

<그림5-6-4 JMPN 입력상태 OFF 모니터링>

- JMPN 펑션블럭의 입력상태가 ON일 경우, 펑션블럭의 뒤쪽 프로그램은 실행됩니다.
- JMPN 펑션블럭의 입력상태가 OFF일 경우, 펑션블럭의 뒤쪽 프로그램은 점프되기 때문에 실행되지 않습니다.
- LABEL 펑션블럭의 입력상태가 출력상태가 됩니다. 즉, 입력상태가 ON일 경우 펑션블럭 뒤쪽 프로그램은 실행됩니다.

7.7. 데이터 컨트롤 펑션블럭 (Data Control)

- 여러 데이터를 대상으로 연산을 수행하기 위한 펑션블럭

- 데이터 컨트롤 펑션블럭 종류

펑션블럭	설 명
MAX	여러 데이터에서 최대값을 찾습니다.
MIN	여러 데이터에서 최소값을 찾습니다.
SUM	여러 데이터의 합을 구합니다.
AVE	여러 데이터의 평균값을 구합니다.
MUX	여러 데이터에서 한 개의 메모리를 선택하여 찾습니다.

7.7.1. MAX

● IN으로부터 N개의 데이터에서 최대값을 OUT에 저장합니다.

< IN > - Singed -

● 최대값을 찾을 데이터의 시작 메모리를 입력합니다.

< OUT > - Signed -

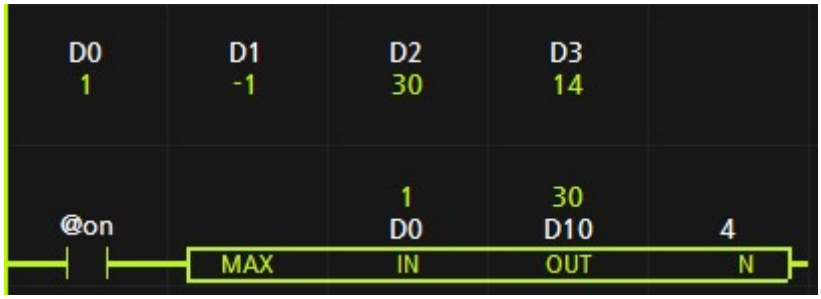
● 찾은 최대값이 저장될 메모리를 입력합니다.

< N > - Unsigned -

● 찾을 메모리의 개수를 입력합니다.

▢ **Example**

● IN:D0 워드메모리를 포함한 4개의 메모리인 D0, D1, D2, D3에서 최대값인 30이 OUT:D10에 저장됩니다.



<그림5-7-1-1 MAX 펑션블럭 모니터링>

7.7.2. MIN

● IN으로부터 N개의 데이터에서 최소값을 OUT에 저장합니다.

< IN > - Singed -

● 최소값을 찾을 데이터의 시작 메모리를 입력합니다.

< OUT > - Signed -

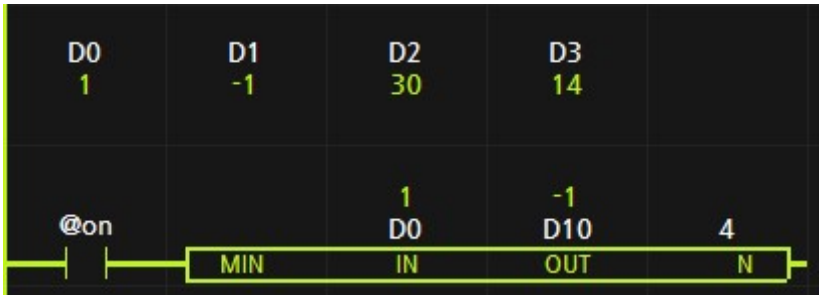
● 찾은 최소값이 저장될 메모리를 입력합니다.

< N > - Unsigned -

● 찾을 메모리의 개수를 입력합니다.

▢ **Example**

● IN:D0 워드메모리를 포함한 4개의 메모리인 D0, D1, D2, D3에서 최소값인 -1이 OUT:D10에 저장됩니다.



<그림5-7-2-1 MIN 펑션블럭 모니터링>

7.7.3. SUM

● IN으로부터 N개의 데이터의 합을 OUT에 저장합니다.

< IN > - Singed -

● 합을 계산할 데이터의 시작 메모리를 입력합니다.

< OUT > - Signed -

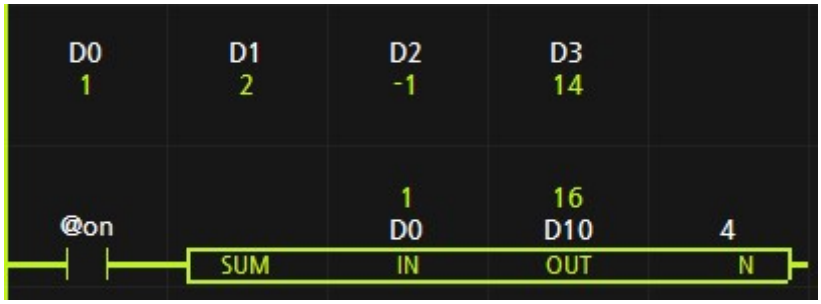
● 계산된 합이 저장될 메모리를 입력합니다.

< N > - Unsigned -

● 합을 계산할 메모리의 개수를 입력합니다.

▢ **Example**

● IN:D0 워드메모리를 포함한 4개의 메모리인 D0, D1, D2, D3의 합인 16이 OUT:D10에 저장됩니다.



<그림5-7-3-1 SUM 펑션블럭 모니터링>

7.7.4. AVE

● IN으로부터 N개의 데이터의 평균값을 OUT에 저장합니다.

< IN > - Singed -

● 평균값을 계산할 데이터의 시작 메모리를 입력합니다.

< OUT > - Signed -

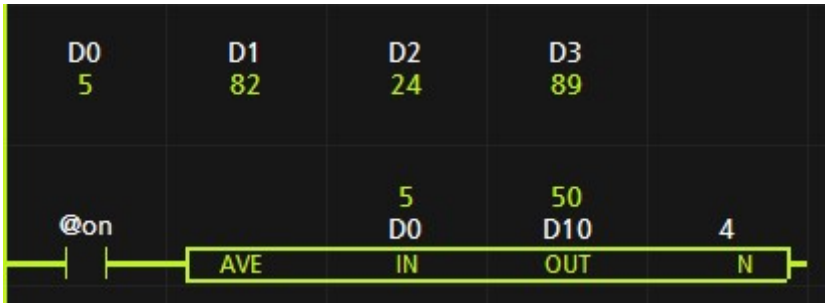
● 계산된 평균값이 저장될 메모리를 입력합니다.

< N > - Unsigned -

● 평균을 계산할 메모리의 개수를 입력합니다.

▢ **Example**

● IN:D0 워드메모리를 포함한 4개의 메모리인 D0, D1, D2, D3의 평균값인 50이 OUT:D10에 저장됩니다.



<그림5-7-4-1 AVE 펑션블럭 모니터링>

7.7.5. MUX

- IN으로부터 N번째의 데이터 값을 OUT에 저장합니다.

< IN > - Singed -

- 선택할 메모리의 시작 메모리를 입력합니다.

< OUT > - Signed -

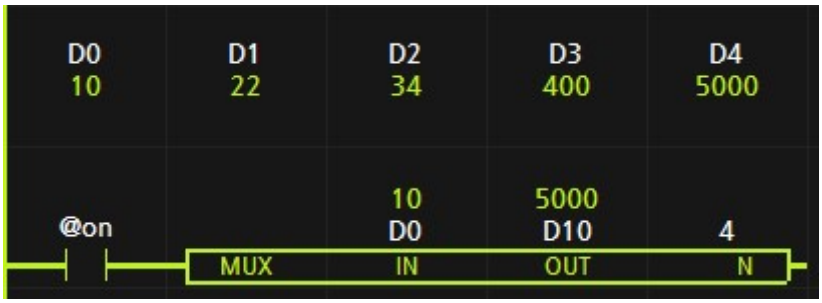
- 선택한 메모리 값이 저장될 메모리를 입력합니다.

< N > - Unsigned -

- 선택할 메모리의 위치를 입력합니다.

☐ **Example**

- IN:D0 워드메모리로부터 4번째 메모리인 D4의 값인 5000이 OUT:D10에 저장됩니다.



<그림5-7-5-1 MUX 펑션블럭 모니터링>

7.8. 프로그램 컨트롤 펑션블럭

- 여러줄의 프로그램을 컨트롤하기 위한 펑션블럭

- 워드로직 펑션블럭 종류

펑션블럭	설 명
MCS	마스터 컨트롤을 실행합니다/
MCSCLR	마스터 컨트롤을 종료합니다.

7.8.1. MCS

- 입력상태가 ON일 때, 마스터 컨트롤 영역안의 프로그램을 실행합니다.
- 입력상태가 OFF일 때, 마스터 컨트롤 영역안의 프로그램을 실행하지 않습니다. 단, 마스터 컨트롤 영역안의 코일은 OFF됩니다.

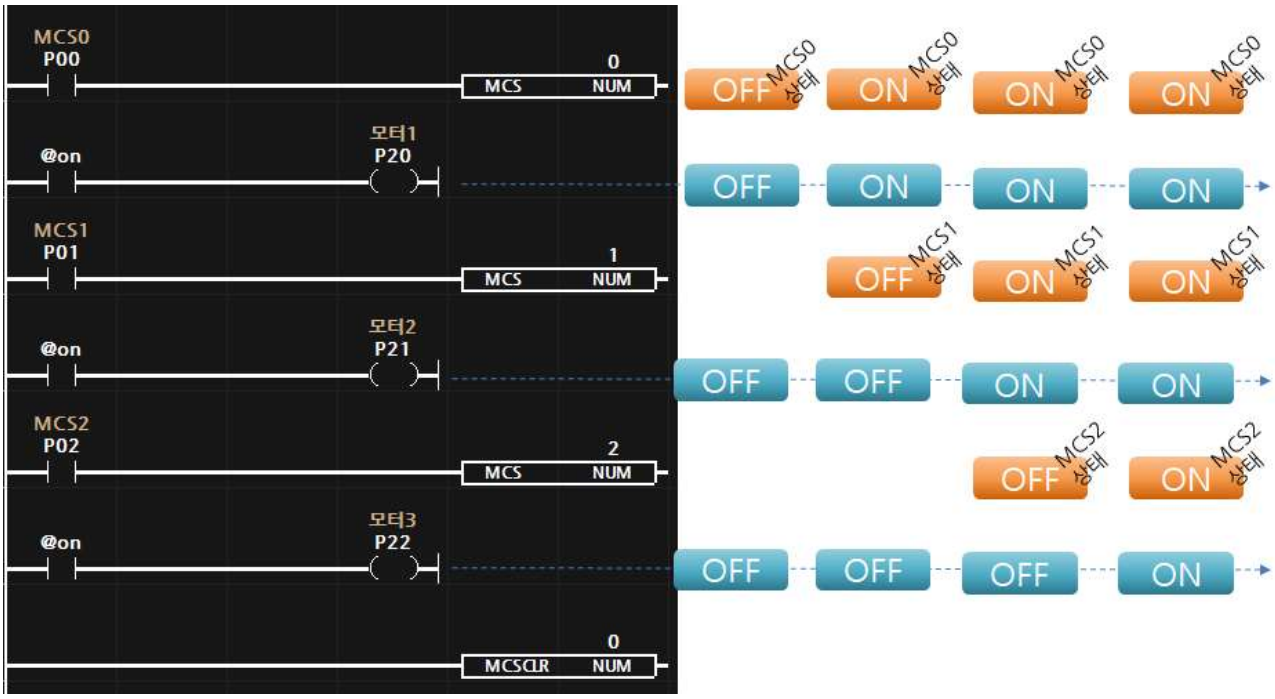
< NUM>

- 마스터 컨트롤 번호를 상수로 0 ~ 7까지 사용할 수 있습니다.
- NUM의 상수값은 0부터 1씩 증가해야 합니다.

- MCS는 MCS와 MCSCLR안에 있는 프로그램을 컨트롤합니다. MCS 0 ~ MCSCLR 0를 사용할 경우, 영역안에 있는 프로그램은 MCS 펑션블럭의 입력상태가 ON일 때, 동작합니다. 단, 코일은 입력상태가 OFF일 때 OFF됩니다.

▢ **Example**

- MCS0:P00가 OFF일 경우, 모터1:P20, 모터2:P21, 모터3:,P22는 무조건 모두 OFF됩니다.
- MCS0:P00가 ON일 경우, 모터1:P20은 ON됩니다. 이때, MCS1:P01이 OFF일 경우, 모터2:P21, 모터3:P22는 무조건 모두 OFF됩니다.
- MCS0:P00 및 MCS1:P01이 ON일 경우, 모터1:P20과 모터2:P21은 ON됩니다. 이때, MCS2:P02이 OFF일 경우, 모터3:P22는 OFF됩니다.
- MCS0:P00, MCS1:P01, MCS2:P02 모두 ON일 경우, 모든 코일은 ON됩니다.



<그림5-6-5 MCS 상태에 따른 컨트롤 동작>

7.8.2. MCSCLR

- NUM번의 마스터 컨트롤을 종료합니다.

< NUM>

- MCS 펑션블럭의 NUM과 동일해야 하며, 해당 NUM번의 마스터 컨트롤을 종료합니다.
- 마스터 컨트롤 번호를 상수로 0 ~ 7까지 사용할 수 있습니다.
- MCS 0 ~7을 사용했을 때, MCSCLR 7은 7번의 마스터 컨트롤만 종료시킵니다.
- MCS 0 ~7을 사용했을 때, MCSCLR 0은 0~7번의 마스터 컨트롤만 종료시킵니다. 즉, 낮은 수치의 NUM은 높은 수치의 NUM까지 모두 종료시킵니다.

7.9. 워드로직 펑션블럭

- 워드 및 더블워드 메모리의 논리연산을 실행하는 펑션블럭입니다.

- 워드로직 펑션블럭 종류

펑션블럭	설 명
WAND	AND 논리연산을 취합니다.
WOR	OR 논리연산을 취합니다.
WXOR	XOR(EXCLUSIVE OR) 논리연산을 취합니다.

7.9.1. WAND

- IN1과 IN2를 논리적 AND를 취하여 OUT에 저장합니다.

< IN1 >

- 워드 및 더블워드를 입력합니다.

< IN2 >

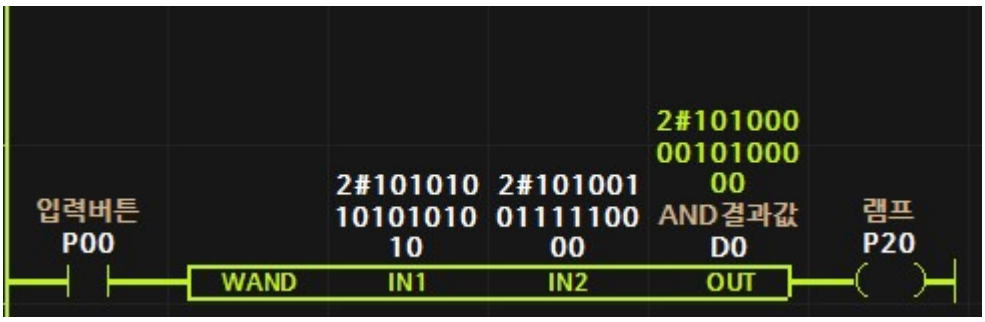
- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

☐ **Example**

- 입력버튼:P00가 ON일 경우, IN1:2#1010101010101010과 IN2:2#1010010111110000을 논리적 AND를 취하여 AND출력값:D0에 저장합니다. D0는 2#1010000010100000의 값이 저장됩니다. 이후, 출력상태가 ON되어 램프:P20도 ON됩니다.



<그림5-7-1 WAND 펑션블럭 모니터링>

- 논리적 AND는 두 수중 0이 하나라도 있으면 0이 됩니다. 즉, 두 수 모두 1일 경우에만 1이 되는 논리식입니다.
- AND Masking 기법 : 선택적 리셋 기법으로, 대상물에 마스크 워드를 AND시켜 대상물에서 특정 비트들을 리셋 시키는 방법입니다.

```
      0b01010001
AND   0b11001100
      ----------
      0b01000000
```

7.9.2. WOR

- IN1과 IN2를 논리적 OR를 취하여 OUT에 저장합니다.

< IN1 >

- 워드 및 더블워드를 입력합니다.

< IN2 >

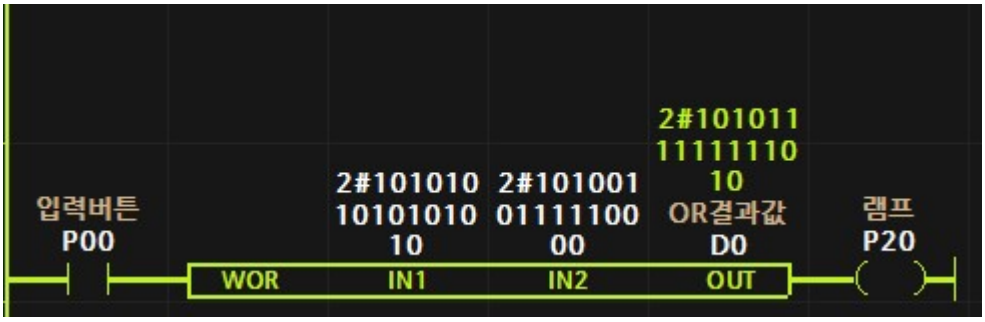
- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

▢ **Example**

- 입력버튼:P00가 ON일 경우, IN1:2#1010101010101010과 IN2:2#1010010111110000을 논리적 OR를 취하여 OR출력값:D0에 저장합니다. D0는 2#1010111111111010의 값이 저장됩니다. 이후, 출력상태가 ON되어 램프:P20도 ON됩니다.



<그림5-7-2 WOR 펑션블럭 모니터링>

- 논리적 OR는 두 수중 1이 하나라도 있으면 1이 됩니다. 즉, 두 수 모두 0일 경우에만 0이 되는 논리식입니다.
- OR Masking 기법 : 선택적 세트 기법으로, 대상물에 마스크 워드를 OR시켜 대상물에서 특정 비트들을 세트 시키는 방법입니다.

```
    0b01010001
OR  0b11001100
    ----------
    0b11011101
```

7.9.3. WXOR

- IN1과 IN2를 논리적 XOR를 취하여 OUT에 저장합니다.

< IN1 >

- 워드 및 더블워드를 입력합니다.

< IN2 >

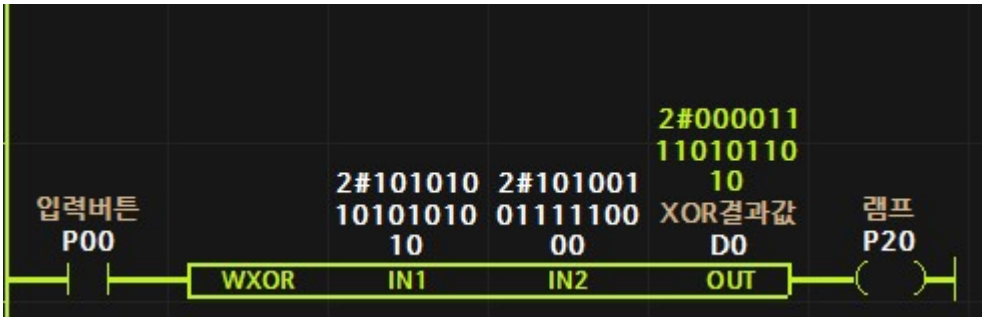
- 워드 및 더블워드를 입력합니다.

< OUT >

- 워드 및 더블워드 메모리를 입력합니다.

▢ **Example**

- 입력버튼:P00가 ON일 경우, IN1:2#1010101010101010과 IN2:2#1010010111110000을 논리적 XOR를 취하여 XOR출력값:D0에 저장합니다. D0는 2#0000111101011010의 값이 저장됩니다. 이후, 출력상태가 ON되어 램프:P20도 ON됩니다.



<그림5-7-3 WXOR 펑션블럭 모니터링>

- XOR Masking 기법 : 선택적 반전 기법으로, 대상물에 마스크 워드를 XOR시켜 대상물에서 특정 비트들을 반전 시키는 방법입니다.

```
     0b01010001
EOR  0b11001100
     ----------
     0b10011101
```

7.10. 아날로그 펑션블럭

● 제품의 아날로그 입력 및 출력과 고속펄스출력를 제어하기 위한 펑션블럭입니다.

● 아날로그 펑션블럭 종류

펑션블럭	설 명
ADC	아날로그 입력포트에 입력된 아날로그 전기신호(DC 0~5V, DC 0~10V, 0~20mA)를 디지털신호로 변경합니다.
ADC2	아날로그 입력포트에 입력된 아날로그 전기신호(DC 1~5V, 4~20mA)를 디지털신호로 변경합니다.
DAC	아날로그 출력포트로 DAC 펑션블럭의 OUT값을 전기신호(DC 0~5V, DC 0~10V, 0~20mA)로 출력합니다.
NTEMP	온도센서 입력포트에 연결된 NTC 써미스터 β-3950 (10kΩ/20℃) 온도센서의 온도값을 읽어옵니다.
PTEMP	온도센서 입력포트에 연결된 PT100Ω 온도센서(2선식,3선식)의 온도값을 읽어옵니다.
PWM	고속펄스출력(PWM) 포트에 일정 폭의 펄스를 출력합니다.
FPWM	고속펄스출력(PWM) 포트에 일정 주파수로 펄스를 출력합니다.
NPWM	고속펄스출력(PWM) 포트에 일정 주파수로 일정 펄스 개수를 출력합니다.
POFF	고속펄스출력(PWM)을 정지시킵니다.
PSR	고속펄스출력의 상태값을 출력합니다.
PID	PID 알고리즘을 실행하여 MV값을 출력합니다.
PID_PWM	PID 알고리즘을 실행하여 고속펄스출력을 출력합니다.

7.10.1. ADC (아날로그 입력 : DC 0~5V, DC 0~10V, 0~20mA)

- 아날로그 입력포트에 입력된 아날로그 전기신호를 디지털 수치로 변환합니다. (Ex. DC 0~5V, DC 0~10V, 0~20mA)
- 아날로그 입력포트 CHANNEL에 입력된 아날로그 전기신호를 MIN VALUE ~ MAX VALUE의 범위의 디지털 수치로 변환합니다.

< CHANNEL > -Unsigned-

- 아날로그 입력포트 번호를 상수 또는 메모리로 입력합니다.

< MAX VALUE > -Signed-

- 변환되어 저장되는 디지털 수치의 최대값을 상수 또는 메모리로 입력합니다.

< MIN VALUE > -Signed-

- 변환되어 저장되는 디지털 수치의 최소값을 상수 또는 메모리로 입력합니다.

< OUT > -Signed-

- 디지털 수치로 변환된 값을 저장할 메모리를 입력합니다.

☐ **Example**

- 입력버튼:P00가 ON일 때, 아날로그 입력포트 0채널에 입력된 아날로그 전기신호를 0~1000의 디지털 수치로 변경하여 아날로그값:D0에 저장합니다. 입력버튼:P00의 상태에 따라 ADC펑션블럭의 실행출력이 ON/OFF되어 램프:P20도 같이 ON/OFF 됩니다.

<ADC 펑션블럭>

- 0~10V D.C의 아날로그 입력포트일 경우, 아날로그값:D0는 4.78V가 아날로그 입력포트 0번 채널에 입력되었다는 의미입니다.

7.10.2. ADC2 (아날로그 입력 : 4~20mA, DC 1~5V)

- 아날로그 입력포트에 입력된 아날로그 전기신호를 디지털 수치로 변환합니다. (Ex. 4~20mA, DC 1~5V)
- 아날로그 입력포트 CHANNEL에 입력된 아날로그 전기신호를 MIN VALUE ~ MAX VALUE의 범위의 디지털 수치로 변환합니다.

< CHANNEL > -Unsigned-

- 아날로그 입력포트 번호를 상수 또는 메모리로 입력합니다.

< MAX VALUE > -Signed-

- 변환되어 저장되는 디지털 수치의 최대값을 상수 또는 메모리로 입력합니다.

< MIN VALUE > -Signed-

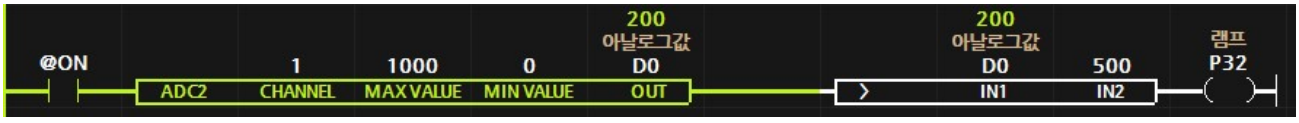
- 변환되어 저장되는 디지털 수치의 최소값을 상수 또는 메모리로 입력합니다.

< OUT > -Signed-

- 디지털 수치로 변환된 값을 저장할 메모리를 입력합니다.
- 아날로그 입력포트에 연결된 케이블이 단선일 경우에 32,767의 값이 출력됩니다.

☐ **Example**

- 입아날로그 입력포트 1채널에 입력된 아날로그 전기신호(4~20mA or 1~5V)를 0~1000의 디지털 수치로 변경하여 아날로그값:D0에 저장합니다. 이후, ADC2 펑션블럭의 실행출력이 ON되어 비교펑션블럭에 의해 아날로그값:D0가 500보다 크면 램프:P32가 ON이되고 500보다 작거나 같으면 램프:P32는 OFF가 됩니다.



<ADC2 펑션블럭>

- 4~20mA 아날로그 입력포트일 경우, 아날로그값:D0가 0일 경우 4mA 이하의 아날로그 신호가 입력되었다는 것입니다.

- MIN VALUE가 100일 경우, 아날로그 신호가 4mA 이하일 경우 100이 아날로그값:D0에 저장됩니다.

7.10.3. DAC (아날로그 출력)

- 디지털 수치를 아날로그 출력포트에 아날로그 전기신호로 출력합니다.
- 아날로그 출력포트 CHANNEL에 MIN VALUE ~ MAX VALUE의 범위에서 OUT값 만큼의 전기적신호를 출력합니다.

< CHANNEL > -Unsigned-

- 아날로그 출력포트를 상수 또는 메모리로 입력합니다.

< MAX VALUE > -Floating Point, Signed-

- 출력할 디지털 수치의 최대값을 상수 또는 메모리로 입력합니다.

< MIN VALUE > -Floating Point, Signed-

- 출력할 디지털 수치의 최소값을 상수 또는 메모리로 입력합니다..

< OUT >

- 디지털 수치의 최소값과 최대값 범위에서 출력할 디지털값을 상수 또는 메모리로 입력합니다.

☐ **Example**

- 입력버튼:P00가 ON일 때, 아날로그 출력포트 0번 채널에 0 ~ 1000 범위에서 500만큼의 전기적인 신호를 출력합니다. 이때, 램프:P20도 ON됩니다.

<DAC 펑션블럭>

- 0~10V D.C의 아날로그 출력포트일 경우, 5V의 전기신호를 출력한다는 의미입니다.

7.10.4. NTEMP (NTC 써미스터 온도)

● 온도센서 입력포트에 연결된 NTC 써미스터 β-3950 (10kΩ/20℃) 온도센서의 온도값을 읽어옵니다.

< CHANNEL > -Unsigned-

● 온도센서 입력채널을 상수 또는 메모리로 입력합니다.

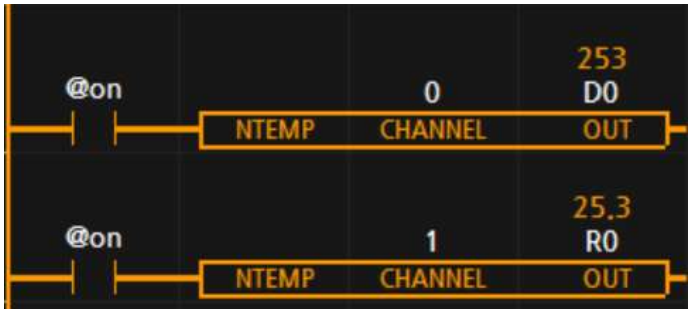
< OUT > -Signed-

● 온도값을 저장시킬 워드 메모리(D메모리) 또는 실수 메모리(R메모리)를 입력합니다.

▢ **Example**

● 온도센서 입력채널 0번에 연결된 NTC 써미스터 온도센서 값을 D0에 저장합니다. 253은 25.3℃를 의미합니다.

● 온도센서 입력채널 1번에 연결된 NTC 써미스터 온도센서 값을 R0에 저장합니다. 25.3은 25.3℃를 의미합니다.



<NTEMP 펑션블럭>

7.10.5. PTEMP (PT100Ω 온도센서)

- PT100Ω 온도센서의 온도값을 읽어옵니다.
- MPA 제품군에서 F(PT100Ω 온도센서 입력) 모듈에서만 사용할 수 있습니다.

< CHANNEL > -Unsigned-

- 온도센서 입력채널을 상수 또는 메모리로 입력합니다.

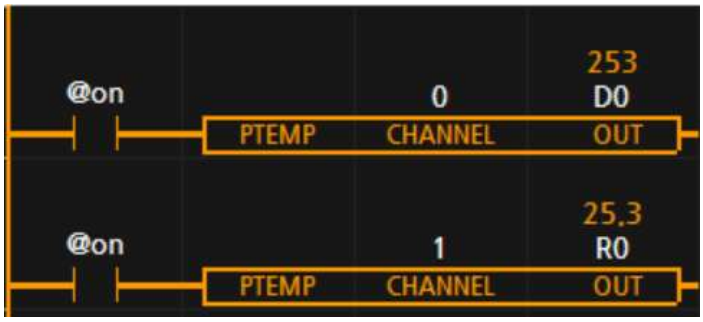
< OUT > -Signed-

- 온도값을 저장시킬 워드 메모리(D메모리) 또는 실수 메모리(R메모리)를 입력합니다.

▢ **Example**

- PT100Ω 온도센서 입력채널 0번에 연결된 PT100Ω 온도센서 값을 D0에 저장합니다. 253은 25.3℃를 의미합니다.

- PT100Ω 온도센서 입력채널 1번에 연결된 PT100Ω 온도센서 값을 R0에 저장합니다. 25.3은 25.3℃를 의미합니다.



<PTEMP 펑션블럭>

7.10.6. PWM (펄스 출력)

- 일정한 폭을 가진 고속펄스를 출력합니다.
- 고속펄스포트(PWM)에 WIDTH길이에서 DUTY의 폭만큼 HIGH(ON)인 펄스를 연속적으로 출력합니다.

< PORT > -Unsigned-

- 펄스를 출력할 포트번호를 상수 또는 메모리로 입력합니다.

< DUTY > -Floating Point, Unsigned-

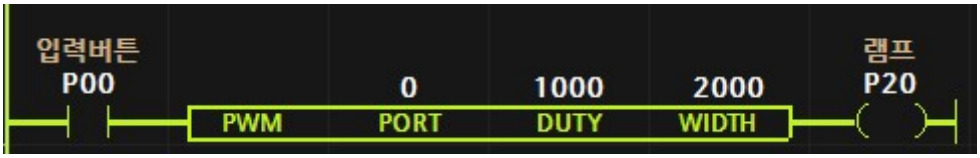
- 펄스의 HIGH(ON) 길이를 상수 또는 메모리로 입력합니다.
- 0 ~ WIDTH 범위에서 값을 입력할 수 있습니다.

< WIDTH > -Floating Point, Unsigned-

- 펄스의 전체 길이를 상수 또는 메모리로 입력합니다.
- 0 ~ 65535 범위에서 값을 입력할 수 있습니다.

☐ **Example**

- 입력버튼:P00가 ON이 되면, 고속펄스 0번포트에 길이가 2000이고 HIGH가 1000인 펄스를 연속적으로 출력합니다. 이때, 램프:P20도 ON됩니다. 펄스의 출력을 중지시키기 위해서는 PWMOFF 펑션블럭을 사용해야 합니다.



<PWM 펑션블럭>

☐ **Caution**

- PWM0~2 는 하나의 펄스발생 채널을 사용하고 있습니다.
- PWM3~5 는 하나의 펄스발생 채널을 사용하고 있습니다.
- FPWM 또는 NPWM을 쓰고 있을 경우, 동채널에 PWM 펑션블럭을 사용하면 안됩니다.

7.10.7. FPWM (주파수 펄스출력)

- 일정한 주파수를 가진 고속펄스를 출력합니다.
- 고속펄스포트(PWM)에 FREQ 주파수로 Duty비(펄스 폭)가 50%인 펄스를 연속적으로 출력합니다.

< PORT > -Unsigned-

- 펄스를 출력할 포트번호를 상수 또는 메모리로 입력합니다.

< FREQ > -Unsigned-

- 출력할 펄스의 주파수를 상수 또는 메모리로 입력합니다.
- 주파수는 Hz 단위로 1은 1Hz, 1000은 1kHz, 1000000은 1Mhz입니다.
- 0 ~ 1Mhz의 주파수를 사용할 수 있습니다.

▢ **Example**

- 입력버튼:P00가 ON이 되면, 고속펄스 0번포트에 Duty비(펄스 폭)가 50%인 1000Hz 펄스를 연속적으로 출력합니다. 이때, 램프:P20도 ON됩니다. 펄스의 출력을 중지시키기 위해서는 PWMOFF 펑션블럭을 사용해야 합니다.

〈FPWM 펑션블럭〉

▢ **Caution**

- PWM0~2 는 하나의 펄스발생 채널을 사용하고 있습니다.
- PWM3~5 는 하나의 펄스발생 채널을 사용하고 있습니다.
- NPWM을 쓰고 있을 경우, 동채널의 다른 PWM포트에 FPWM 펑션블럭을 사용하면 안됩니다.
- FPWM을 쓰고 있을 경우, 동채널의 다른 PWM포트에 FPWM 펑션블럭은 같은 주파수일 경우에만 사용할 수 있습니다. 만약, 사용할 경우 처음에 실행한 FPWM의 주파수는 후자에 실행한 FPWM 주파수로 출력됩니다.
- PWM을 사용하고 있을 경우, 동채널의 다른 PWM포트에 FPWM 펑션블럭을 사용하면 동 채널의 PWM포트도 FPWM의 주파수로 출력되고 Duty비(펄스 폭)은 유지됩니다.

7.10.8. FDPWM (주파수 및 듀티비 펄스출력)

- 일정한 주파수 및 듀티비를 가진 펄스를 출력합니다.
- 고속펄스포트(PWM)에 FREQ 주파수로 Duty비(펄스 폭)를 0~100%로 설정하여 펄스를 연속적으로 출력합니다.

< PORT > -Unsigned-

- 펄스를 출력할 포트번호를 상수 또는 메모리로 입력합니다.

< FREQ > -Unsigned-

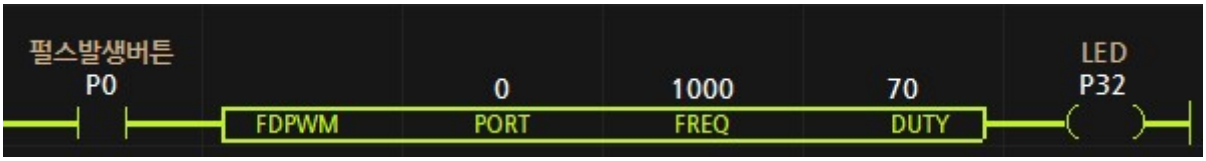
- 출력할 펄스의 주파수를 상수 또는 메모리로 입력합니다.
- 주파수는 Hz 단위로 1은 1Hz, 1000은 1kHz, 1000000은 1Mhz입니다.
- 0 ~ 1Mhz의 주파수를 사용할 수 있습니다.

< DUTY > -Unsigned-

- 출력할 펄스의 듀티비를 0~100으로 0~100%로 설정할 수 있습니다.

☐ **Example**

- 펄스발생버튼:P0가 ON이 되면, 고속펄스 0번포트에 Duty비(펄스 폭)가 70%인 1kHz 펄스를 연속적으로 출력합니다. 이때, LED:P32도 ON됩니다. 펄스의 출력을 중지시키기 위해서는 PWMOFF 펑션블럭을 사용해야 합니다.



<FDPWM 펑션블럭>

☐ **Caution**

- PWM0~2 는 하나의 펄스발생 채널을 사용하고 있습니다. 각 포트의 듀티비는 다르게 출력이 가능하나 주파수는 마지막에 실행된 명령어의 주파수로 동일하게 출력됩니다.
- PWM3~5 는 하나의 펄스발생 채널을 사용하고 있습니다. 각 포트의 듀티비는 다르게 출력이 가능하나 주파수는 마지막에 실행된 명령어의 주파수로 동일하게 출력됩니다.
- NPWM을 쓰고 있을 경우, 동채널에 사용할 수 없습니다.
- PWM을 사용하고 있을 경우, 동채널의 다른 PWM포트에 FDPWM 펑션블럭을 사용하면 동 채널의 PWM포트도 FDPWM의 주파수로 출력됩니다.

7.10.9. NPWM (제한된 펄스개수 출력)

- 일정한 주파수를 가진 고속펄스를 제한된 수만큼 출력합니다.
- 고속펄스포트(PWM)에 FREQ 주파수로 Duty비(펄스 폭)가 50%인 펄스를 N개의 수만큼 출력합니다.

< PORT > -Unsigned-

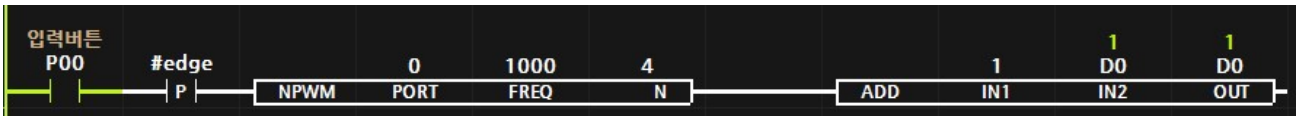
- 펄스를 출력할 포트번호를 상수 또는 메모리로 입력합니다.

< FREQ > -Unsigned-

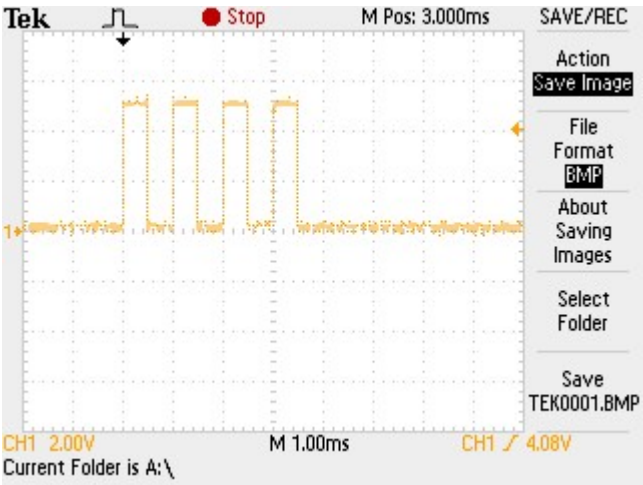
- 출력할 펄스의 주파수를 상수 또는 메모리로 입력합니다.
- 주파수는 Hz 단위로 1은 1Hz, 1000은 1kHz입니다.
- 0 ~ 60khz의 주파수를 사용할 수 있습니다. 30kHz를 넘지 않는 것을 권장드립니다.

☐ **Example**

- 입력버튼:P00가 ON이 되면, 고속펄스 0번포트에 Duty비(펄스 폭)가 50%인 1000Hz 펄스를 4개 출력합니다. 이때, ADD 펑션블럭에 의해 OUT:D0는 1이 합산되어 1이 됩니다. NPWM 펑션블럭 앞에 상승엣지접점:#edge에 의해 NPWM펑션블럭은 단 한번 실행됩니다. 펄스의 출력을 중지시키기 위해서는 PWMOFF 펑션블럭을 사용해야 합니다.



<NPWM 펑션블럭>



<그림5-8-6 NPWM 출력된 오실로스코프 디스플레이>

▢ **Caution**

- 상승엣지접점을 사용해서 동작조건이 되었을 때, 한번만 실행되도록 해야 합니다.

그렇지 않을 경우, NPWM펑션블럭은 계속실행되어 N 개수를 초과한 펄스가 출력되게 됩니다.

- PWM0~2 는 하나의 펄스발생 채널을 사용하고 있습니다.
- PWM3~5 는 하나의 펄스발생 채널을 사용하고 있습니다.
- NPWM을 쓰고 있을 경우, 동채널에 다른 PWM, FPWM, NPWM을 사용할 수 없습니다.
  단, NPWM의 펄스출력이 모두 완료되었을 경우 다른포트를 사용할 수 있습니다.

Ex) NPWM 펑션블럭에 PWM1포트를 사용할 경우, 0번과 2번은 사용금지 포트가 됩니다.

7.10.10. POFF (펄스출력 정지)

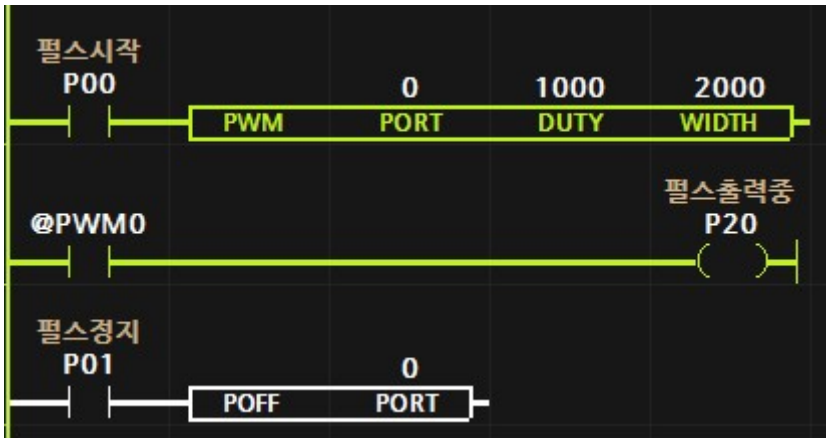
- PORT의 고속펄스포트(PWM)을 정지시킵니다.

< PORT > -Unsigned-

- 펄스출력을 정지할 포트번호를 상수 또는 메모리로 입력합니다.

▢ **Example**

- 펄스시작:P00가 ON이 되면, 고속펄스 0번포트에 Duty비가 50%인 펄스가 출력됩니다. 펄스가 출력되는동안 @PWM0비트가 ON되어 펄스출력중:P20은 ON됩니다. 이후, 펄스정지:P01가 ON이 되면, 고속펄스 0번포트의 펄스출력은 정지됩니다.



<PWM과 POFF 펑션블럭>

- FPWM과 NPWM 펑션블럭에서 생성한 펄스도 정지됩니다.

7.10.11. PSR (펄스출력 상태)

- PORT의 고속펄스포트(PWM)의 출력상태를 읽어옵니다.

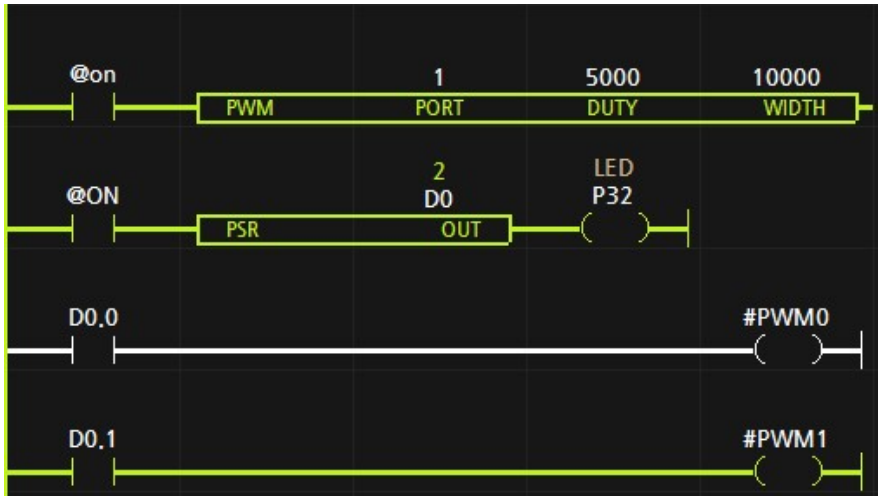
< OUT > -Unsigned-

- 펄스출력상태를 저장할 메모리를 입력합니다.
- 아래 표와 같이 PWM0포트는 0번째 비트가 PWM5는 5번째 비트가 ON되어 있으면 PWM0과 PWM5포트로 펄스가 출력되고 있다는 것입니다.

비트	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM 채널	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

☐ **Example**

- 1번줄 : PWM펑션블럭이 실행되어 1번 고속펄스포트에 펄스가 발생되었다.
- 2번줄 : PSR 펑션블럭이 실행되어 고속펄스 상태값이 D0에 저당되고 LED:P32 출력을 ON 시킨다.
- 3번줄 : 고속펄스 상태값이 저장된 D0에서 0번째 비트는 PWM0의 상태값이므로 D0.0은 OFF되어 사용자 비트메모리인 #PWM0은 OFF된다.
- 3번줄 : 고속펄스 상태값이 저장된 D0에서 1번째 비트는 PWM1의 상태값이므로 D0.1은 ON되어 사용자 비트메모리인 #PWM1은 ON된다.



< PSR 펑션블럭 >

- D0.0 및 D0.1는 매뉴얼의 "워드 메모리의 비트 접점 사용"을 참조해 주세요.

7.10.12. PID (Proportional-Integral-Differential)

- SV(설정값)에 도달하기 위하여 PV(현재값)을 참조하여 비례,적분,미분을 계산하여 출력값 MV값을 출력합니다.

< PV : Present Value > -Floating Point, Signed -

- 현재값으로 참조할 메모리를 입력합니다.

< SV : Setting Vale > -Floating Point, Signed -

- 설정값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KP : Proportional Value > -Floating Point, Signed -

- 비례매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KI : Integral Value > -Floating Point, Signed -

- 적분매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KD : Differential Value > -Floating Point, Signed -

- 미분매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< OUT_MAX : Output Max Value > -Floating Point, Signed -

- MV 출력의 최대값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< OUT_MIN : Output Min Value > -Floating Point, Signed -

- MV 출력의 최소값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< MV : Manipulated Value > -Floating Point, Signed -

- PID 연산 결과값인 제어값을 저장할 메모리를 입력합니다.

▢ 비례, 적분, 미분항 계산값을 확인하는 방법

- PID 펑션블럭을 사용하면 펑션블럭당 7개의 더블워드 사용자 메모리를 사용합니다. 메모리값은 실수이며, 모니터링은 모니터링 표현방식을 Floating Point 방식으로 설정해야 합니다.
- 비례항 : ###__PID_PTERM_x (x는 PID번호)
- 적분항 : ###__PID_ITERM_x (x는 PID번호)
- 미분항 : ###__PID_DTERM_x (x는 PID번호)
- 스캔타임 : ###__PID_MSCAN_x (x는 PID번호)
- 에러값 : ###__PID_ERROR_x (x는 PID번호, 에러는 SV-PV)
- 이전 에러값 : ###__PID_PREERR_x (x는 PID번호)
- PID번호 x는 프로그램의 0번줄부터 아래쪽으로 PID, PID_PWM 펑션블럭에 순차적으로 0부터 상승 카운트로 자동 부여됩니다.
- 적분항 값의 증가는 적분항 최대값(OUT_MAX – OUT_MIN)을 초과하지 않습니다.
- 적분항 값의 감소는 적분항 최소값(- (OUT_MAX – OUT_MIN))을 초과하지 않습니다.

▢ Example

- 온도센서 0채널의 온도값을 R0 실수 메모리에 저장한다.
- 현재값 R0, 설정값 30도, P값 1000, I값 0.1, D값 1.5, 출력 최대값 60,000, 출력 최소값 0으로 PID 알고리즘을 실행하고 계산된 MV값을 R1 실수 메모리에 저장한다.
- 아날로그 출력 0채널에 MV값(R1)을 출력한다.
- 만약, PID 펑션블럭의 실행조건이 OFF가 되면, 적분항 및 MV의 값은 0으로 초기화 됩니다.

< PID 펑션블럭 사용 예제 >

7.10.13. PID_PWM (Proportional-Integral-Differential for PWM)

- SV(설정값)에 도달하기 위하여 PV(현재값)을 참조하여 비례,적분,미분을 계산하여 출력값을 PWM 포트로 펄스를 출력합니다.

< PORT : PWM Port > - Unsigned -

- MV 출력값을 참조하여 펄스를 출력할 PWM 포트번호를 입력합니다.

< PV : Present Value > -Floating Point, Signed -

- 현재값으로 참조할 메모리를 입력합니다.

< SV : Setting Vale > -Floating Point, Signed -

- 설정값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KP : Proportional Value > -Floating Point, Signed -

- 비례매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KI : Integral Value > -Floating Point, Signed -

- 적분매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KD : Differential Value > -Floating Point, Signed -

- 미분매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< KD : Differential Value > -Floating Point, Signed -

- 미분매개변수값으로 참조할 실수값, 정수값, 메모리를 입력합니다.

< MV : Manipulated Value > -Floating Point, Signed -

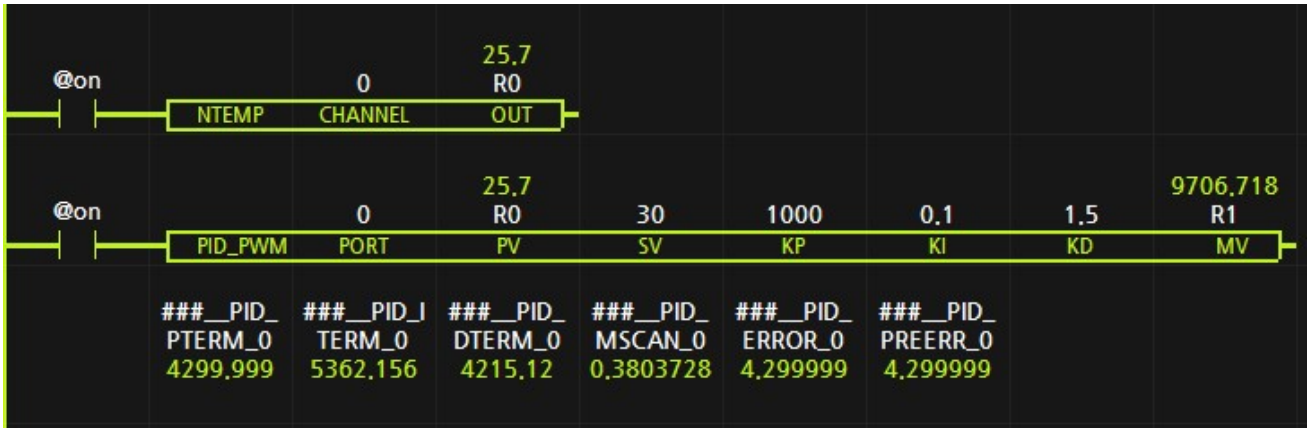
- PID 연산 결과값인 제어값을 저장할 메모리를 입력합니다. 해당값(0~ 65,535)이 펄스의 Duty이며, PORT에 입력한 PWM 포트번호로 펄스가 출력됩니다.

▢ 비례, 적분, 미분항 계산값을 확인하는 방법

- PID 펑션블럭을 사용하면 펑션블럭당 7개의 더블워드 사용자 메모리를 사용합니다. 메모리값은 실수이며, 모니터링은 모니터링 표현방식을 Floating Point 방식으로 설정해야 합니다.
- 비례항 : ###__PID_PTERM_x (x는 PID번호)
- 적분항 : ###__PID_ITERM_x (x는 PID번호)
- 미분항 : ###__PID_DTERM_x (x는 PID번호)
- 스캔타임 : ###__PID_MSCAN_x (x는 PID번호)
- 에러값 : ###__PID_ERROR_x (x는 PID번호, 에러는 SV-PV)
- 이전 에러값 : ###__PID_PREERR_x (x는 PID번호)
- PID번호 x는 프로그램의 0번줄부터 아래쪽으로 PID, PID_PWM 펑션블럭에 순차적으로 0부터 상승 카운트로 자동 부여됩니다.
- 적분항 값의 증가는 적분항 최대값(65535.0)을 초과하지 않습니다.
- 적분항 값의 감소는 적분항 최소값(-65535.0)을 초과하지 않습니다.

▢ Example

- 온도센서 0채널의 온도값을 R0 실수 메모리에 저장한다.
- 현재값 R0, 설정값 30도, P값 1000, I값 0.1, D값 1.5으로 PID 알고리즘을 실행하고 계산된 MV값(0~ 65,535)을 R1 실수 메모리에 저장하고 MV(9706.718)의 값이 Duty인 펄스를 PORT에 입력한 PWM 포트로 출력합니다.
- 만약, PID_PWM 펑션블럭의 실행조건이 OFF가 되면, 적분항 및 MV의 값은 0으로 초기화 되며 PWM 펄스출력도 정지됩니다.



< PID_PWM 펑션블럭 사용 예제 >

7.11. 이동 펑션블럭 (MOVE)

- 메모리 값을 복사하기 위한 펑션블럭입니다.

- 아날로그 펑션블럭 종류

펑션블럭	설 명
MOV	한 개의 값 또는 메모리를 한 개의 메모리에 복사합니다.
GMOV	여러개의 메모리를 여러개의 메모리에 복사합니다.
FMOV	한 개의 값 또는 메모리를 여러개의 메모리에 복사합니다.

7.11.1. MOV

- IN의 값을 OUT에 복사합니다.

< IN >

- 복사할 상수 및 메모리를 입력합니다.

< OUT >

- 복사될 메모리를 입력합니다.

▢ **Example**

- 입력버튼:P00가 ON일 경우, IN:1234를 OUT:D0에 복사하여 D0값이 1234가 됩니다. 이때, 펄스출력중:P20도 ON됩니다.

〈그림5-7-4 MOV 펑션블럭 모니터링〉

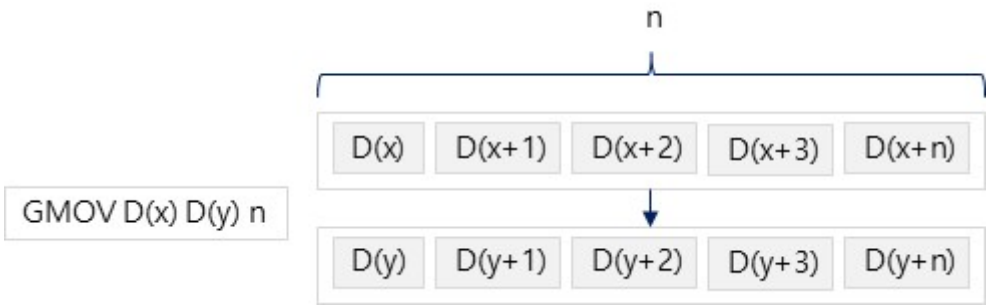
7.11.2. GMOV

- IN 파라미터 메모리부터 N개의 메모리 값을 OUT 파라미터 메모리로 N개의 메모리에 복사합니다.

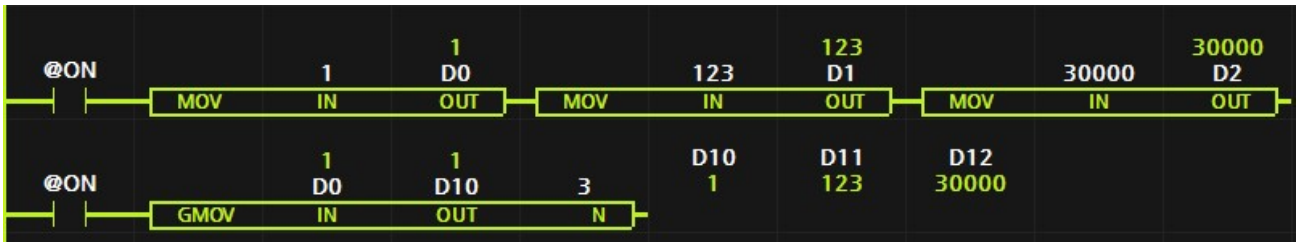
< IN >
- 복사할 메모리를 입력합니다.

< OUT >
- 복사될 메모리를 입력합니다.

< N > -Unsigned-
- 복사할 양을 상수 또는 메모리를 입력합니다.
- OUT 파라미터 메모리의 값이 바이트일 경우, IN 파라미터를 바이트 단위로 읽어서 복사.
- OUT 파라미터 메모리의 값이 워드일 경우, IN 파라미터를 워드 단위로 읽어서 복사.
- OUT 파라미터 메모리의 값이 더블워드일 경우, IN 파라미터를 거블워드 단위로 읽어서 복사 합니다.



☐ **Example**
- D0, D1, D2 워드 메모리에 각각 1, 123, 30000을 입력하고 GMOV 펑션블럭으로 D0부터 D10으로 3개의 워드를 복사했을 때 D10, D11, D12 워드 메모리는 각각 D0~D2와 동일하게 1, 123, 30000이 됩니다.



<그림5-7-2-1 GMOV 펑션블럭 모니터링>

7.11.3. FMOV

● IN 파라미터 메모리의 값을 OUT 파라미터 메모리로부터 N개의 메모리에 복사합니다.

< IN >

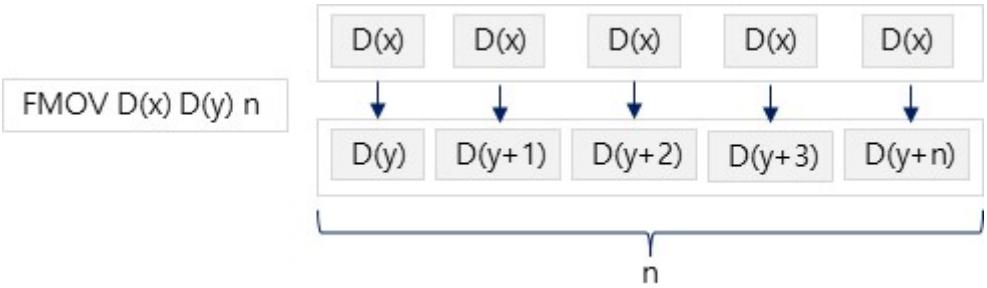
● 복사할 상수값 또는 메모리를 입력합니다.

< OUT >

● 복사될 메모리를 입력합니다.

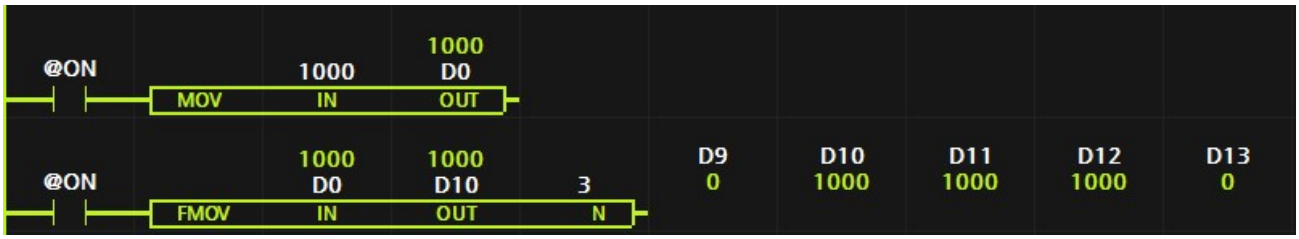
< N > -Unsigned-

● 복사될 메모리의 양을 입력합니다. (상수 또는 메모리)



☐ **Example**

● D0 워드 메모리에 각각 1000을 입력하고 FMOV 펑션블럭으로 D0을 D10으로부터 3개의 워드에 복사했을 때 D10, D11, D12 워드 메모리는 모두 1000이 됩니다.



<그림5-7-3-1 FMOV 펑션블럭 모니터링>

7.12. 워드로직 펑션블럭

- 워드 및 더블워드 메모리의 논리연산을 실행하는 펑션블럭입니다.

- 워드로직 펑션블럭 종류

펑션블럭	설 명
SHL	메모리의 비트를 왼쪽으로 이동시킵니다.
SHR	메모리의 비트를 오른쪽으로 이동시킵니다.

7.12.1. SHL

● IN의 비트를 N개만큼 왼쪽으로 이동시킵니다.

< IN >

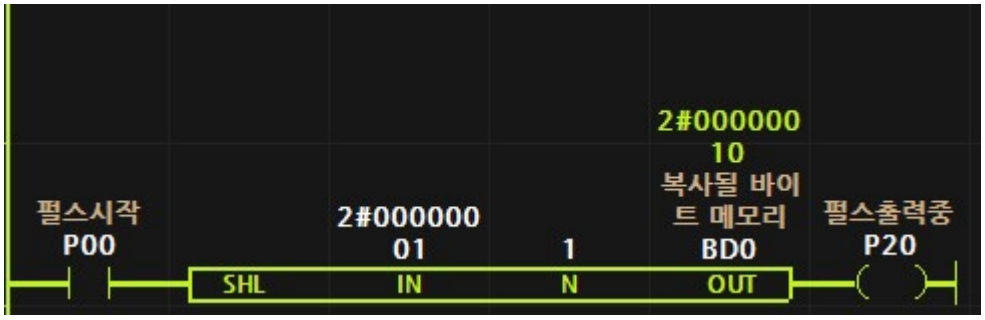
● 이동시킬 상수 또는 메모리를 입력합니다.

< OUT >

● 이동시킨 값이 저장될 메모리를 입력합니다.

☐ **Example**

● 입력버튼:P00가 ON일 경우, IN:2#00000001를 1비트 왼쪽으로 이동시킨 2#00000010을 OUT:BD0에 저장합니다. 이때, 펄스출력중:P20도 ON됩니다.



<그림5-7-5 SHL 펑션블럭 모니터링>

7.12.2. SHR

● IN의 비트를 N개만큼 오른쪽으로 이동시킵니다.

< IN >

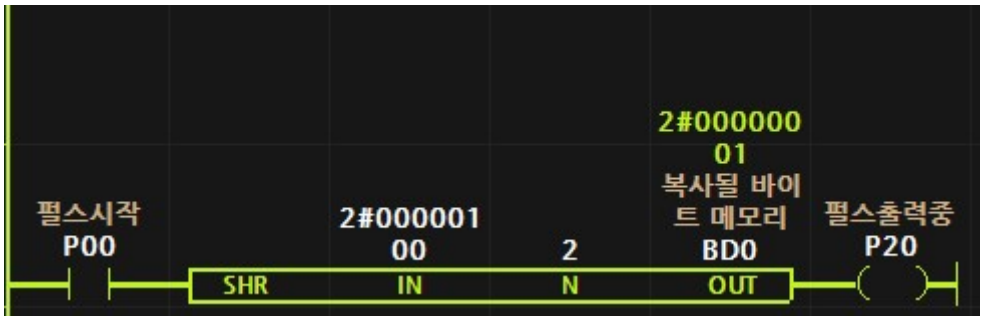
● 이동시킬 상수 또는 메모리를 입력합니다.

< OUT >

● 이동시킨 값이 저장될 메모리를 입력합니다.

☐ **Example**

● 입력버튼:P00가 ON일 경우, IN:2#00000100를 2비트 오른쪽으로 이동시킨 2#00000001을 OUT:BD0에 저장합니다. 이때, 펄스출력중:P20도 ON됩니다.



<그림5-7-6 SHR 펑션블럭 모니터링>

7.13. 시리얼통신 펑션블럭

7.13.1. 시리얼 통신박식의 종류

- 시리얼통신은 제품과 제품간의 인터페이스를 위하여 가장 보편적으로 사용되고 있습니다.
- 시리얼통신은 UART, RS232, RS422, RS485 통신방식이 있습니다.

Specification	UART	RS232	RS422	RS485
최대 Driver 최대 Receiver	1 Driver 1 Receiver	1 Driver 1 Receiver	1 Driver 32 Receiver	32 Driver 32 Receiver
최대 통신거리	10m 이내	15m 이내	1km 이내	1km 이내
지원 전송방식	Full Duplex	Full Duplex	Full Duplex	Half Duplex
최대 출력전압	0 ~ VCC(+5V)	±25V	-0.25V ~ +6V	-7V ~ +12V
최대 입력전압	0 ~ VCC(+5V)	±15V	-7V ~ +7V	-7V ~ +12V

- UART는 TTL레벨의 통신방식으로 시리얼통신 방식중 가장 기본이 되는 통신입니다. 통신 전압이 가장 낮기 때문에, 각종 노이즈에 의해 통신의 문제가 발생할 수 있어서 산업에서는 많이 쓰지 않지만, UART 통신만을 제공하는 저렴한 센서 등의 제품들을 연결하기 위해서 필요한 통신방식입니다. UART 통신을 하는 제품과의 거리를 짧게하고 통신속도를 낮게 설정함으로 각종 노이즈 등의 영향을 줄이는 해결책이 있습니다.

- RS232는 UART통신보다 통신전압을 높게하여 산업에서 발생하는 각종 노이즈 등에 의한 문제를 줄인 통신방식입니다. RS232는 제품과 1:1 통신을 해야하는 경우, 산업에서 보편적으로 가장 많이 사용하는 시리얼통신입니다.

- RS485는 차동입력방식을 사용하여 산업에서 발생하는 각종 노이즈를 상쇄시켜 입력받는 방식입니다. 때문에, 통신거리가 길며 제품과 1:N 통신을 해야하는 경우, 산업에서 보편적으로 가장 많이 사용하는 시리얼통신입니다. 통신선이 2가닥으로 간단한 연결이 가능한 장점이 있습니다.

- RS422는 RS485의 차동입력방식을 송신과 수신에 각각 사용함으로서, RS485에서 사용하지 못하는 전이중 방식(Full Duplex)을 사용할 수 있습니다. 하지만. 통신선이 4가닥으로 RS485보다 연결선이 복잡한 단점이 있습니다.

7.13.2. 시리얼 통신의 설정

< BaudRate >

- 시리얼통신의 속도를 말합니다. 1초당 송/수신하는 비트수를 의미합니다. 보편적으로 9600, 19200, 38400, 57600, 115200을 사용하며, 속도가 느릴수로 보다 안정적이고 먼 거리의 통신이 가능합니다.

< Data Bit >

- 송/수신의 전송 패킷의 비트 수를 말합니다. 보편적으로 8 비트를 사용하지만, 경우에 따라 7비트와 9비로 설정하여 사용할 수 있습니다.

< Stop Bit >

- 송신 패킷의 뒤에 Stop 비트를 전송하며, 1비트 혹은 2비트로 설정할 수 있습니다. 보편적으로 1비트를 사용하며, 예전에는 수신측에서 Stop 비트를 놓칠 수 있어 2비트로 사용하기도 하였습니다.

< Parity Bit >

- 송신 패킷의 문제가 없는지 확인하기 위한 비트입니다. 홀수(Odd), 짝수(Even), 없음(None)으로 설정할 수 있습니다. 보편적으로 없음(None)을 선택합니다. 페리티 비트를 사용할 경우, 송신 패킷의 HIGH레벨 신호를 카운트하여 홀수(Odd) 또는 짝수(Even)일 경우 페리티 비트를 HIGH로 송신합니다. 보편적으로 페리티 비트로 오류를 검출할 수 있는

7.13.3. 시리얼 통신 모드

● 통신모드는 아래와 같이 3가지를 제공합니다.
① Normal : 정의되지 않은 통신으로 PUT, GET 펑션블럭으로 통신을 구성해야 합니다.
② Modbus RTU : 산업에서 보편적으로 사용하는 통신방식입니다.
③ Cubloc Modbus RTU : HMI(터치디스플레이)와 연결하기 위해 사용되는 통신방식입니다.

< Normal >

● 사용자가 통신 프로토콜을 정의하여 PUT, GET 펑션블럭으로 프로그램할 수 있습니다.

< Modbus RTU >

● 산업에서 보편적으로 사용하고 있는 범용 프로토콜입니다. 대부분의 시리얼 통신을 지원하는 센서, 인디게이터 등은 Modbus RTU Slave를 지원하고 있습니다.
● Modbus는 Master 1대와 Slave 여러대를 연결할 수 있으며, Master는 보편적으로 HMI(터치디스플레이) 또는 PC가 됩니다.
● Master에서 1대 또는 여러대의 Slave 제품에 쓰기 또는 읽기 명령을 내립니다. Slave에서 Master에게 먼저 송신하지 않고, 무조건 Master에서 SLave에 요청하면 Slave는 이에 응답을 하는 방식입니다.

< Cubloc Modbus RTU >

● 저희가 판매하고 있는 HMI에서 사용하는 통신 프로토콜입니다. 기본은 Modbus와 같지만, Modbus는 D0 메모리를 D0라고 하지 않고 저희 컨트롤러 제품에서 정의한 2000번지라고 합니다. 따라서, HMI에서 쉽게 D0의 데이터를 읽고 쓰게 하기 위해 2000번지라고 안하고 D0라고 정의할 수 있도록 미리 프로토콜을 맞추어 놓았습니다. 저희가 판매하는 ComfileHMI 제품을 구입하신 경우라면, Cubloc Modbus RTU를 선택해 주시기 바랍니다.

시리얼통신 펑션블럭의 종류

PUT : 시리얼통신의 송신버퍼에 데이터를 집어넣어 데이터를 전송합니다.
GET : 시리얼통신의 수신버퍼에서 데이터를 가져 옵니다.
PUT_LEN : 시리얼통신의 송신버퍼에 쓰여지 데이터의 길이를 가져 옵니다.
GET_LEN : 시리얼통신의 수신버퍼에 쓰여진 데이터의 길이를 가져 옵니다.

7.13.4. PUT (송신)

● 시리얼통신 CHANNEL에 DATA의 LEN 바이트 개수를 송신합니다.

< CHANNEL > -Unsigned-

● 송신 채널의 상수 또는 메모리로 입력합니다.

	8A4R(T)-R2	16A8R(T)-R2	8A4R(T)	8A8R	16A8R	16A8R8T
CHANNEL 1	RS232	RS232	RS232	RS232	RS232	RS232
CHANNEL 2	x	x	x	RS485	RS485	RS485
CHANNEL 3	x	x	x	UART	UART	UART

< DATA >

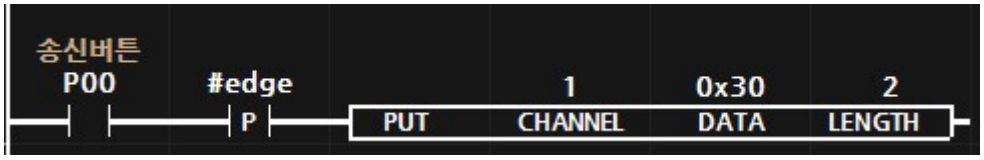
● 송신 데이터를 상수 또는 메모리로 입력합니다.

< LENGTH > -Unsigned-

● 송신할 데이터의 개수를 상수 또는 메모리로 입력합니다.

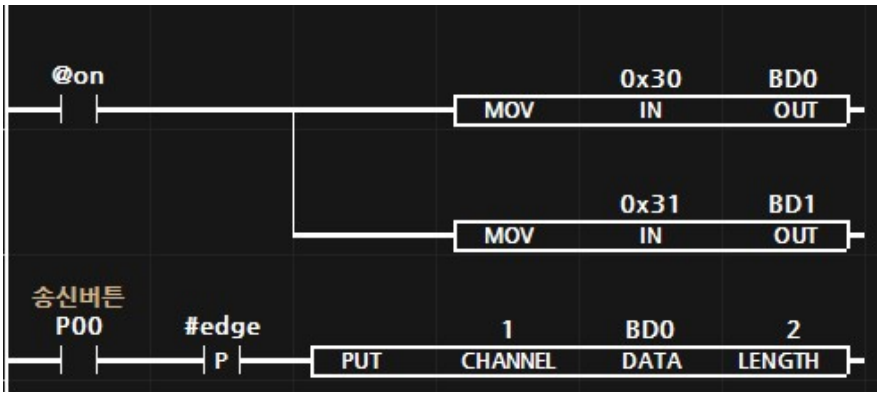
☐ **Example**

● 송신시작:P00가 ON되면, 시리얼통신 1채널에 16진수 30을 2번 전송합니다.
전송패킷 : 0x30, 0x30



<그림5-9-1 PUT 펑션블럭>

● 송신시작:P00가 ON되면, 시리얼통신 1채널에 BD0 및 BD1을 전송합니다.
전송패킷 : 0x30, 0x31



<그림5-9-2 PUT 펑션블럭>

7.13.5. GET (수신)

● 시리얼통신 CHANNEL에 DATA의 LEN 바이트 개수를 송신합니다.

< CHANNEL > -Unsigned-

● 수신 채널의 상수 또는 메모리로 입력합니다.

	8A4R(T)-R2	16A8R(T)-R2	8A4R(T)	8A8R	16A8R	16A8R8T
CHANNEL 1	RS232	RS232	RS232	RS232	RS232	RS232
CHANNEL 2	x	x	x	RS485	RS485	RS485
CHANNEL 3	x	x	x	UART	UART	UART

< LENGTH >

● 수신한 데이터의 개수를 상수 또는 메모리로 입력합니다.

< OUT > -Unsigned-

● 수신한 데이터를 저장할 메모리를 입력합니다.

☐ **Example**

● 시리얼통신 1채널에 GETLEN 펑션블럭으로 수신한 패킷의 수를 D10에 저장합니다. 이후, 수신패킷:D10이 1개 이상 있으면, 1바이트를 D영역 바이트 메모리:BD0에 저장합니다.

<그림5-9-3 GET 펑션블럭 미수신 상태 모니터링>

● 외부기기에서 시리얼통신 1채널에 0x30을 송신하면 아래와 같이 D영역 바이트 메모리:BD0에 0x30이 저장됩니다.

<그림5-9-4 GET 펑션블럭 수신 상태 모니터링>

7.13.6. PUTLEN (송신버퍼개수)

● 시리얼통신 CHANNEL의 송신버퍼에 남아있는 송신패킷의 수를 바이트 단위로 읽어옵니다.

< CHANNEL > -Unsigned-

● 송신버퍼의 채널을 상수 또는 메모리로 입력합니다.

	8A4R(T)-R2	16A8R(T)-R2	8A4R(T)	8A8R	16A8R	16A8R8T
CHANNEL 1	RS232	RS232	RS232	RS232	RS232	RS232
CHANNEL 2	x	x	x	RS485	RS485	RS485
CHANNEL 3	x	x	x	UART	UART	UART

< OUT > -Unsigned-

● 송신버퍼에 남아있는 송신패킷의 수가 저장될 메모리를 입력합니다.

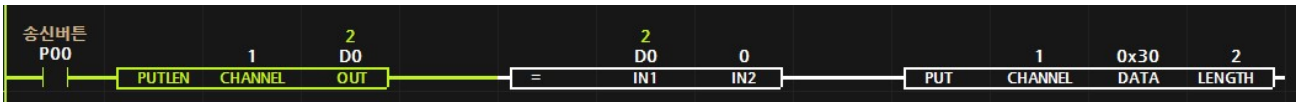
▢ **Example**

● 송신버튼:P00가 ON일 때, PUTLEN 펑션블럭을 이용하여 송신버퍼에 있는 송신패킷을 바이트 단위로 읽어와 OUT:D0에 저장합니다. 이후, 송신버퍼의 패킷수인 D0가 0일 때, 즉, 송신버퍼가 비어있을때에만 PUT 펑션블럭을 이용하여 16진수 30을 시리얼통신 1채널로 2바이트 전송합니다.

<그림5-9-5 PUTLEN 펑션블럭을 이용하여 송신버퍼가 비어 있일때 PUT 펑션블럭 실행 >

● 만약, PUTLEN 펑션블럭을 이용하지 않고 PUT 펑션블럭만을 이용하면 시리얼통신 1채널로 송신할 패킷이 송신버퍼에 계속 쌓입니다. 때문에, 향후 송신버퍼의 용량을 초과하여 송신한 패킷의 오류가 발생할 수 있습니다. 송신버퍼가 쌓이는 이유는 시리얼통신으로 보내는 속도보다 컨트롤러의 스캔타임(프로그램을 한번 실행하는 속도)이 더욱 빠르기 때문입니다.

● 송신버튼:P00가 ON이고 송신버퍼에 아직 2개의 송신패킷이 남아 있을 경우, 아래와 같이 비교펑션블럭에 의해 PUT 펑션블럭을 실행하지 않습니다.



<그림5-9-6 PUTLEN 펑션블럭을 이용하여 송신버퍼가 비어 있지 않을 때 PUT 펑션블럭 중단 >

7.13.7. GETLEN (수신버퍼개수)

● 시리얼통신 CHANNEL의 수신버퍼에 남아있는 수신패킷의 수를 읽어옵니다.

< CHANNEL > -Unsigned-

● 수신버퍼의 채널을 상수 또는 메모리로 입력합니다.

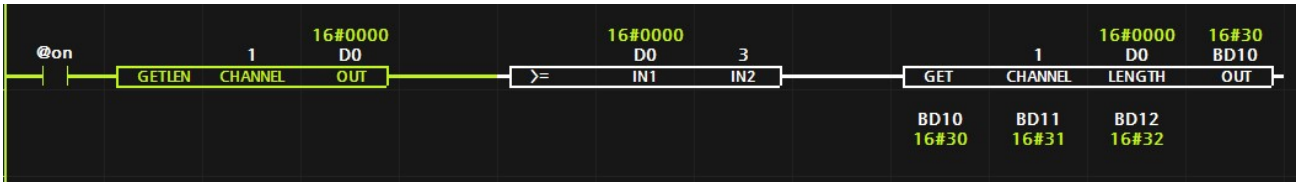
	8A4R(T)-R2	16A8R(T)-R2	8A4R(T)	8A8R	16A8R	16A8R8T
CHANNEL 1	RS232	RS232	RS232	RS232	RS232	RS232
CHANNEL 2	x	x	x	RS485	RS485	RS485
CHANNEL 3	x	x	x	UART	UART	UART

< OUT > -Unsigned-

● 수신버퍼에 남아있는 수신패킷의 수가 정자될 메모리를 입력합니다.

☐ **Example**

● GETLEN 펑션블럭을 실행하여 시리얼통신 1채널의 수신버퍼에 남아있는 수신패킷의 수를 OUT:D0에 저장합니다. 이후, 수신패킷수:D0가 3패킷 이상이면 GET 펑션블럭을 실행하여 시리얼통신 1채널의 수신버퍼의 패킷을 D영역 바이트 메모리인 BD10부터.. BD11, BD12에 저장합니다. 만약, 수신버퍼에 남은 패킷이 4개일 경우, BD13까지 사용됩니다.



<그림5-9-7 GETLEN 펑션블럭을 이용하여 3패킷(바이트)을 GET 펑션블럭을 이용하여 수신 >

8. HMI 연결 / 외부기기 연결

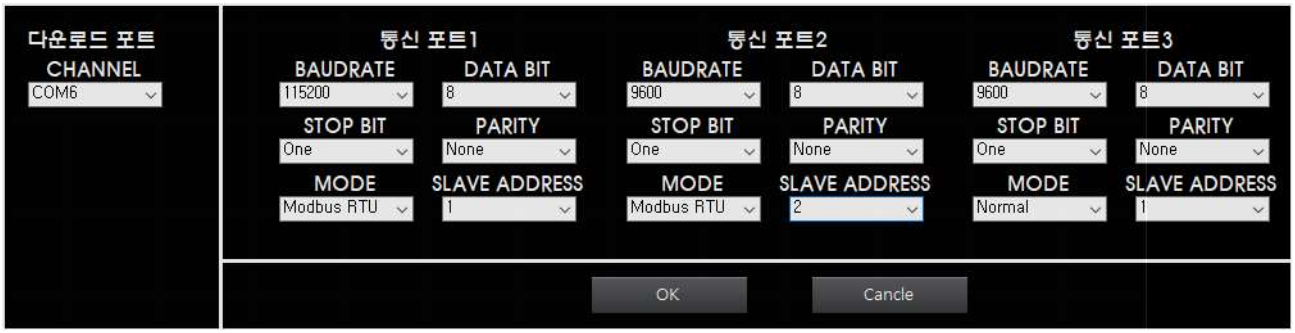
MP STUDIO(MPS제품군)은 시리얼 통신의 전채널에 하기와 같은 프로토콜을 지원합니다.

- MODBUS RTU SLAVE
  산업범용 프로토콜로서, MODBUS RTU MASTER를 지원하는 여러 산업장비와 연결이 가능합니다.
- CUBLOC MODBUS RTU SLAVE
  저희가 취급하고 있는 ComfileHMI 제품과 연결이 가능합니다.
- LS산전 CNET (XBC/LINK)
  LS산전의 MASTER-K PLC의 통신 프로토콜로서, CNET 프로토콜을 지원하는 여러 기업의 HMI 제품과 연결이 가능합니다.

8.1. MODBUS RTU

8.1.1. MODBUS 설정

MP STUDIO의 통신포트 설정에서 통신속도 등을 변경할 수 있습니다.



BAUDRATE : 통신속도 (초당 전송되는 비트로서 단위는 b/s입니다)
DATA BIT : 하나의 패킷에 전송되는 BIT의 개수이며, 보통 8 BIT를 사용합니다.
STOP BIT : 정지신호로 사용되는 BIT의 개수이며, 보통 ONE을 사용합니다.
PARITY BIT : 송/수신의 오류를 검출하기 위한 검사방법을 선택할 수 있습니다.
MODE : Modbus RTU를 선택하여 MPS 제품군을 Modbus RTU Slave 모드로 설정할 수 있습니다.
SLAVE ADDRESS : Modbus RTU Slave의 Address를 설정할 수 있습니다. Modbus Master에서 Slave에게 명령을 전송할 때, 해당 Slave Address의 MPS 제품군만 명령을 받아 동작하게 됩니다.

8.1.2. MODBUS RTU 패킷 구성

MODBUS는 MASTER에서 SLAVE에게 메모리정보를 달라고 요청하거나 메모리 값을 변경하라고 명령을 하는 형식으로서, 항상 MASTER가 SLAVE에게 명령을 전달하는 방식으로 통신을 진행합니다.

통신패킷의 구성은 슬레이브 어드레스, 펑션코드, 시작 어드레스, 길이, 데이터, CRC (에러검출)로 구성됩니다. 상세한 설명은 각각의 펑션코드 설명을 참조해주세요.

8.1.3. 펑션코드

펑션코드는 Master에서 Slave에게 송신을 할 때, 명령의 종류를 구분지어 줍니다.

펑션코드	기 능	설 명
펑션코드 01	Read Coil Status	메모리를 Bit 단위로 읽음
펑션코드 02	Read Input Status	메모리를 Bit 단위로 읽음
펑션코드 03	Read Holding Registers	메모리를 Word 단위로 읽음
펑션코드 04	Read Input Registers	메모리를 Word 단위로 읽음
펑션코드 05	Force Single Coil	하나의 Bit의 메모리를 On/Off 시킴
펑션코드 06	Preset Single Registers	하나의 Word의 메모리를 변경시킴
펑션코드 15	Force Multiple Coils	여러 Bit의 메모리를 On/Off 시킴
펑션코드 16	Preset Multiple Registers	여러 Word의 메모리를 변경시킴

※ 펑션코드 1번과 2번 그리고 3번과 4번은 동일한 펑션코드입니다.

8.1.4. CRC

CRC는 Modbus 송/수신 패킷의 마지막 두 개의 바이트로서, 송/수신한 패킷이 제대로 전달 되었는지를 알기 위한 검증코드입니다. 계산 방식이 복잡하므로 아래 링크를 참조하시기 바랍니다.
www.lammertbies.nl/comm/info/crc-calculation.html

8.1.5. START ADDRESS

Start Address는 MPS 제품군의 메모리의 주소값을 의미합니다. 각각의 접점명은 아래의 표의 범위에서 시작 어드레스를 설정하여 사용할 수 있습니다.

접점명	Address (주 소)	설 명
P	0 ~ 997	입/출력 메모리
M	1,000 ~ 1,999	내부 M 메모리
D	2,000 ~ 2,999	내부 D 메모리
C	3,000 ~ 3,999	내부 C 메모리 (카운터)
T	4,000 ~ 4,999	내부 T 메모리 (타이머)
R	5,000 ~ 5,499	내부 R 메모리 (실수)

Start Address는 아래와 같이 계산하여 사용하셔야 합니다.

① Bit

※ P32 : P메모리 시작 어드레스(0) + 32 = 32 (HEX:0020)

※ M88 : M메모리 시작 어드레스(1,000) + 88 = 1,088 (HEX:0440)

② Word

※ D10 : D메모리 시작 어드레스(2,000) + 10 = 2,010 (HEX:07DA)

※ C5 : C메모리 시작 어드레스(3,000) + 5 = 3,005 (HEX:0BBD)

※ T60 : T메모리 시작 어드레스 (4,000) + 60 = 4,060 (HEX:0FDC)

※ R1 : R메모리 시작 어드레스 (5,000) + 1 = 5,001 (HEX:1389)

※ DD1 : D메모리 시작 어드레스 (2,000) + 1 = 2,001 (HEX:07D1)

8.1.6. HMI의 시작 어드레스

HMI 또는 SCADA는 제품에 따라 1000번대, 10000번대 100000번대 주소체계를 사용하지만 MPS는 그와 상관없이 동작합니다. 그 이유는 1000번대의 제품은 어드레스에 1001을 기입했을 경우 실제로 MPS로 들어오는 어드레스 데이터값은 1001을 뺄셈한 0이 들어옵니다. HMI 제품에서 1001을 뺄셈하여 송신하게 됩니다. 2001을 기입하면 2001을 뺄셈하여 0이 송신됩니다. 3011을 기입하면 3001을 뺄셈하여 10이 송신됩니다. HMI에서 1001, 2001, 3001을 구분하는 이유는 펑션코드(비트, 워드읽기, 워드쓰기)를 구분하기 위해 사용하는 것입니다. 때문에, 사용자는 HMI에 펑션코드를 별도로 입력하지 않고 주소값만을 사용합니다. (10000번대 제품도 똑같이 10001을 기입하면 10001을 뺄셈하여 0을 송신합니다.)

예를들어 아래와 같이 사용하시면 됩니다.
(10000번대 어드레스 사용하는 기기, HEX값 어드레스를 사용하는 기기)

① Bit Read
※ P10 : 00001(HEX) + (P메모리 시작 어드레스(0) + 10)(DEC) = 00001(HEX) + A(HEX) = 0000B(HEX)
※ M23 : 00001 + (M메모리 시작 어드레스(1,000) + 23)(DEC) = 00001(HEX) + 3FF(HEX) = 400(HEX)

② Bit Write
※ P32 : 10001(HEX) + (P메모리 시작 어드레스(0) + 32)(DEC) = 10001(HEX) + 20(HEX) = 10021(HEX)
※ M77 : 10001(HEX) + (M메모리 시작 어드레스(1,000) + 77)(DEC) = 10001(HEX) + 435(HEX) = 10436(HEX)

③ Word Read
※ D11 : 30001(HEX) + (D메모리 시작 어드레스(2,000) + 11)(DEC) = 30001(HEX) + 7DB(HEX) = 307DC(HEX)
※ C36 : 30001(HEX) + (C메모리 시작 어드레스(3,000) + 36)(DEC) = 30001(HEX) + BDC(HEX) = 30BDD(HEX)

④ Bit Write
※ D15 : 40001(HEX) + (D메모리 시작 어드레스(2,000) + 15)(DEC) = 40001(HEX) + 7DF(HEX) = 407E0(HEX)
※ C92 : 40001(HEX) + (C메모리 시작 어드레스(3,000) + 92)(DEC) = 40001(HEX) + C14(HEX) = 40C15(HEX)

8.1.7. M2I의 HMI에서 사용할 경우

M2I에서 통신설정을 MODBUS RTU로 설정해야 하며 M2I에서 제공하고 있는 모드버스 주소는 아래와 같습니다.

TOP에서 사용 가능한 디바이스는 아래와 같습니다.
CPU 모듈 시리즈/타입에 따라 디바이스 범위(어드레스) 차이가 있을 수 있습니다. TOP 시리즈는 외부 장치 시리즈가 사용하는 최대 어드레스 범위를 지원합니다. 사용하고자 하는 장치가 지원하는 어드레스 범위를 벗어 나지 않도록 각 CPU 모듈 사용자 매뉴얼을 참조/주의 하십시오.

	Bit Address	Word Address	32 bits	Remarks
Coil	000001 – 065536	000001 – 065521	L/H	
Discrete Input	100001 – 165536	100001 – 165521		*주1)
Input Register	300001.00 – 365536.15	300001 – 365536		*주1)
Holding Register	400001.00 – 465536.15	400001 – 465536		

*주1) 쓰기 불가능(읽기 전용)

Coil과 Discrete Input은 BIT 메모리로서 두 개를 구분 없이 사용하시면 됩니다.
Input Register와 Holding Register는 WORD 메모리로서 두 개를 구분 없이 사용하시면 됩니다.

Coil을 사용할 경우
M2I에서 기입한 어드레스에서 1을 뺀 값을 Start Address로 사용합니다.

Discrete Input을 사용할 경우
M2I에서 기입한 어드레스에서 100001을 뺀 값을 Start Address로 사용합니다.

Input Register를 사용할 경우
M2I에서 기입한 어드레스에서 300001을 뺀 값을 Start Address로 사용합니다.

Holding Register를 사용할 경우
M2I에서 기입한 어드레스에서 400001을 뺀 값을 Start Address로 사용합니다.

Start Address로 사용한다는 것은 실제 통신 패킷값을 의미합니다.

※ D9 : Input Register(400001) + D메모리 시작 어드레스(2000) + Byte(9) = 402010을 M2I에 기입해야 합니다.

8.1.8. MODBUS RTU Protocol Packet

펑션코드 01 : Read Coil Status
펑션코드 02 : Read Input Status

- 메모리를 BIT단위로 읽는 펑션코드입니다.
- 아래는 2번 슬레이브 제품에서 M15~M33을 읽어오는 예제입니다.

- Master에서 송신

패 킷 명	구 분	패 킷 (16진수)
슬레이브 어드레스		02
펑션코드		01
시작 어드레스	HIGH BYTE	03
시작 어드레스	LOW BYTE	F7
길이	HIGH BYTE	00
길이	LOW BYTE	0F
CRC	LOW BYTE	CD
CRC	HIGH BYTE	8B

* 길이는 16#07F8(10#2040)을 초과할 수 없습니다.

- 어드레스 : M메모리 시작 어드레스(1000) + M15 = 1,015= 0x3F7
- 길이 : 읽고자 하는 비트의 개수를 의미합니다.

- Slave로 부터 수신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		01
바이트 카운트		02
데이터	M15 ~ M22	01
데이터	M23 ~ M28	02
CRC	LOW BYTE	7D
CRC	HIGH BYTE	AD

- Master에서 15개의 Bit를 요청하였기 때문에, Slave의 바이트 카운트는 2 Byte가 되며, 데이터 패킷의 응답이 M15와 M24가 High이고나머지는 Low의 응답입니다.
- 아래에 수신된 데이터의 패킷을 Binary와 Hex로 표시하였습니다. 참조해주세요.

BIT	M22	M21	M20	M19	M18	M17	M16	M15
BIN	0	0	0	0	0	0	0	1
HEX	0				1			

BIT	Dummy	M29	M28	M27	M26	M25	M24	M23
BIN	0	0	0	0	0	0	1	0
HEX	0				2			

펑션코드 03 : Read Holding Registers
펑션코드 04 : Read Input Regigsters

- 메모리를 Word단위로 읽는 펑션코드입니다.
- 아래는 2번의 슬레이브 제품의 D32~D33를 읽어오는 예제입니다.

- Master에서 송신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		03
어드레스	HIGH BYTE	07
어드레스	LOW BYTE	F0
길이	HIGH BYTE	00
길이	LOW BYTE	02
CRC	LOW BYTE	C5
CRC	HIGH BYTE	7F

*길이는 16#007F(10#127)을 초과할 수 없습니다.

- 어드레스 : D메모리 시작 어드레스(2000) + D32 = 2032 = 0x07F0
- 길이 : 읽고자 하는 데이터의 워드 개수를 의미합니다.

- Slave로 부터 수신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		03
바이트 카운트		04
데이터1 (DW32)	HIGH BYTE	12
데이터1 (DW32)	LOW BYTE	34
데이터2 (DW34)	HIGH BYTE	56
데이터2 (DW34	LOW BYTE	78
CRC	LOW BYTE	B2
CRC	HIGH BYTE	07

- Master에서 2개의 Word를 요청하였기 때문에, Slave의 바이트 카운트는 4 Byte가 되며, 데이터는 D32부터 D34까지 순차적으로 16진수로 1234,5678의 값으로 응답되었습니다.
- 아래는 데이터 패킷을 Binary와 Hex로 표시한 것입니다.

WORD	D32															
HEX	1(MSB)				2				3				4(LSB)			
BIN	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

WORD	D33															
HEX	5(MSB)				6				7				8(LSB)			
BIN	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0	0

펑션코드 05 : Force Single Coil

- 메모리중 하나의 Bit를 변경하기 위한 펑션코드입니다.
- 아래는 2번의 슬레이브 제품의 P32 출력접점을 On시키는 예제입니다.

- Master에서 송신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		05
어드레스	HIGH BYTE	00
어드레스	LOW BYTE	20
데이터	HIGH BYTE	FF
데이터	LOW BYTE	00
CRC	LOW BYTE	8D
CRC	HIGH BYTE	C3

- 어드레스 : P메모리 시작 어드레스(0) + P32 = 32 = 0x0020
- 데이터 : 0xFF00는 HIGH로 변경하고 0x0000는 LOW로 변경시킵니다.

- Slave로 부터 수신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		05
어드레스	HIGH BYTE	00
어드레스	LOW BYTE	20
데이터	HIGH BYTE	FF
데이터	LOW BYTE	00
CRC	LOW BYTE	8D
CRC	HIGH BYTE	C3

- Master에서P32(16#0020) 메모리에 HIGH(16#FF00) 데이터를 Slave에게 송신을 하였습니다. Slave에서는 데이터가 16#FF00일 경우, Bit를 High로 16#0000은 Low로 인식합니다. Slave는 Master에서 받은 명령을 처리하고 받은 패킷과 똑같은 패킷을 응답합니다.

펑션코드 06 : Preset Single Registers

- 메모리중 하나의 Word를 변경하기 위한 펑션코드입니다.
- 아래는 2번의 슬레이브 제품의 D16 메모리 값을 16#8520으로 변경하는 예제입니다.

- Master에서 송신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		06
어드레스	HIGH BYTE	07
어드레스	LOW BYTE	E0
데이터	HIGH BYTE	85
데이터	LOW BYTE	20
CRC	LOW BYTE	EA
CRC	HIGH BYTE	33

- 어드레스 : D메모리 시작 어드레스(2,000) + D16 = 2,016 = 0x07E0 입니다.

- Slave로 부터 수신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		06
어드레스	HIGH BYTE	07
어드레스	LOW BYTE	E0
데이터	HIGH BYTE	85
데이터	LOW BYTE	20
CRC	LOW BYTE	EA
CRC	HIGH BYTE	33

- Master에서 16#07E0(10#2016) 주소값에 16#8520 값을 쓰도록 Slave에게 송신을 하였습니다. Slave는 Master에서 송신한 패킷을 처리하고 받은 패킷과 동일한 패킷을 Master에게 송신합니다.

펑션코드 15 : Force Multiple Coils

- PLC의 내부메모리중 여러개의 Bit를 변경하기 위한 펑션코드입니다.
- 아래는 02번의 슬레이브의 M27~M40의 BIT 값을 변경하는 예제입니다.

- Master에서 송신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		0F
어드레스	HIGH BYTE	04
어드레스	LOW BYTE	03
길이	HIGH BYTE	00
길이	LOW BYTE	10
바이트 카운트		02
데이터1	FIRST DATA	35
데이터2	SECOND DATA	8B
CRC	LOW BYTE	E4
CRC	HIGH BYTE	14

- 어드레스: M메모리 시작 어드레스(1,000) + M27 = 1,027 = 0x0403
- 길이: M27~M40의 비트개수 = 16 = 0x10
- 바이트 카운트: 16비트의 바이트개수 = 2
- 아래는 변경할 패킷을 Binary와 Hex로 표시한 것입니다.

Bit	M40	M39	M38	M37	M36	M35	M34	M33	M32	M31	M30	M29	M28	M27
Bin	1	1	0	1	0	1	1	0	0	0	1	0	1	1
Hex	3		5				8				B			
	Second Data						First Data							

- Slave로 부터 수신

패킷명	구분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		0F
어드레스	HIGH BYTE	04
어드레스	LOW BYTE	03
길이	HIGH BYTE	00
길이	LOW BYTE	10
CRC	LOW BYTE	A5
CRC	HIGH BYTE	04

- Master에서 16#0403(10#1023) 주소값 부터 16#358B 값을 쓰도록 Slave에게 송신을 하였습니다 Slave는 Master에서 받은 패킷을 처리하고 받은 패킷중 일부를 회신합니다.

펑션코드 16 : Preset Multiple Registers

- PLC의 내부메모리중 여러개의 Word를 변경하기 위한 펑션코드입니다.
- 아래는 2번의 슬레이브 제품의 D22 ~ D24를 16진수 1234,5678,1245로 변경하는 예제입니다.

- Master에서 송신

패킷명	구 분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		10
어드레스	HIGH BYTE	07
어드레스	LOW BYTE	E6
길이	HIGH BYTE	00
길이	LOW BYTE	03
바이트 카운트		06
데이터1	HIGH BYTE	12
데이터1	LOW BYTE	34
데이터2	HIGH BYTE	56
데이터2	LOW BYTE	78
데이터3	HIGH BYTE	12
데이터3	LOW BYTE	45
CRC	LOW BYTE	F3
CRC	HIGH BYTE	0A

- 어드레스: D메모리 시작 어드레스(2000) + D22 = 2022 = 0x07E6
- 길이: D메모리 개수 = 3
- 바이트 카운트 : 수신받을 바이트 개수 = 6

- Slave로 부터 수신

패킷명	구 분	패킷 (16진수)
슬레이브 어드레스		02
펑션코드		10
어드레스	HIGH BYTE	07
어드레스	LOW BYTE	E6
길이	HIGH BYTE	00
길이	LOW BYTE	03
CRC	LOW BYTE	60
CRC	HIGH BYTE	B8

- Master에서 16#07E6(10#2022) 주소값을 시작으로 3개의 Word에 순차적으로 16#1234,16#5678, 16#1245 값을 쓰도록 Slave에게 송신을 하였습니다 Slave는 Master에서 받은 패킷을 처리하고 받은 패킷중 일부를 회신합니다.

8.2. COMFILE HMI

ComfileHMI는 저희 아이로직스 쇼핑몰에서 HMI 카테고리에서 판매하고 있는 HMI 제품입니다.

8.2.1. MP STUDIO 설정

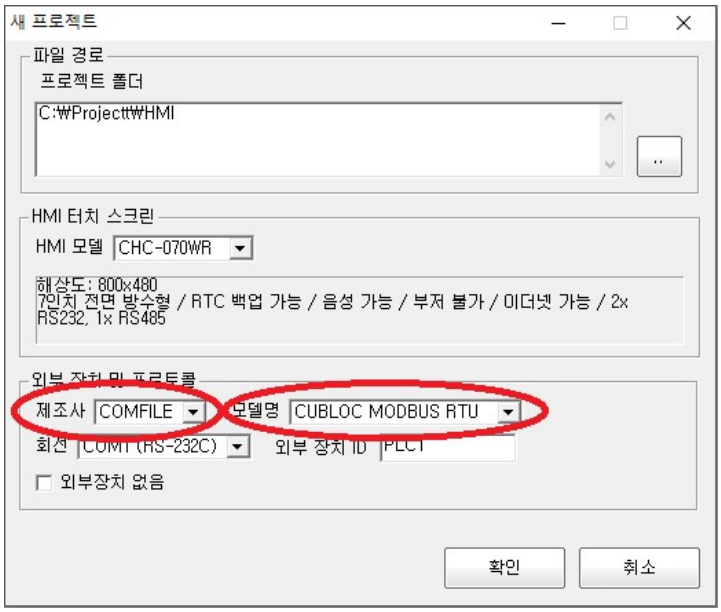
MP STUDIO의 통신포트 설정에서 ComfileHMI와 연결하고자 하는 통신포트의 MODE를 Cubloc Modbus RTU로 설정해야 합니다.

Baudrate : 통신속도입니다. ComfileHMI의 통신속도와 똑같이 설정해야 합니다.
Data Bit : 통신패킷의 비트 구성 수입니다. 기본은 8을 사용합니다.
Stop Bit : 통신패킷의 정지 비트 수입니다. 기본은 1을 사용합니다.
Parity : 통신패킷의 에러율검증을 위한 비트입니다. 기본은 사용하지 않음으로 None입니다.
Mode : Cubloc Modbus RTU를 선택합니다.
Slave Address : Comfile HMI에서 통신으로 요청할 MPS제품의 고유넘버입니다.

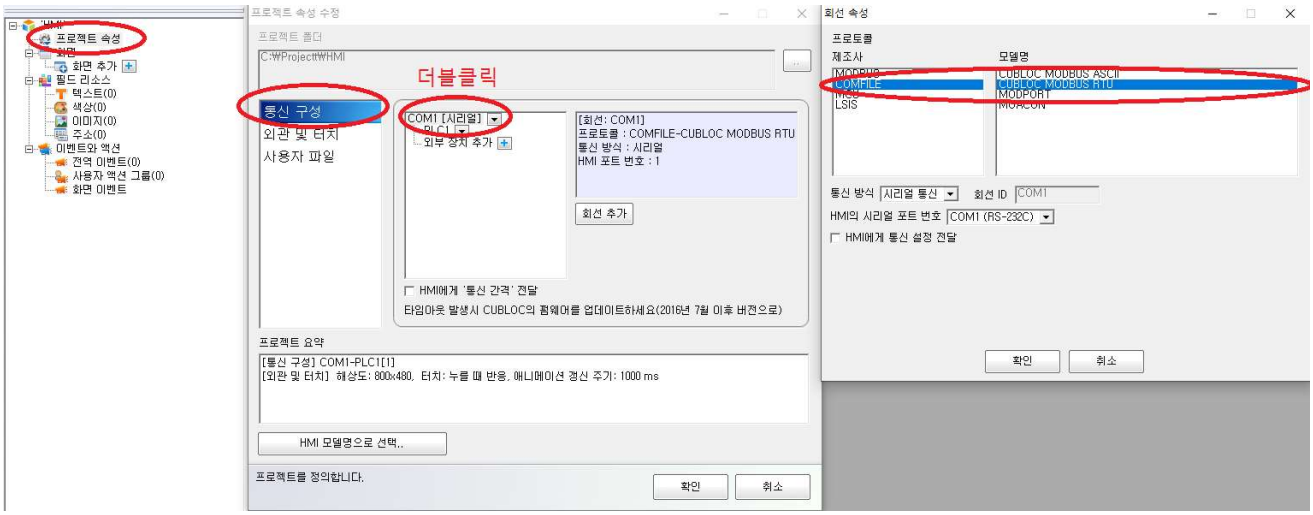
통신포트 설정 후에는 프로그램 다운로드를 실행해야 MPS제품의 통신포트 설정이 변경됩니다.

8.2.2. COMFILE HMI 설정

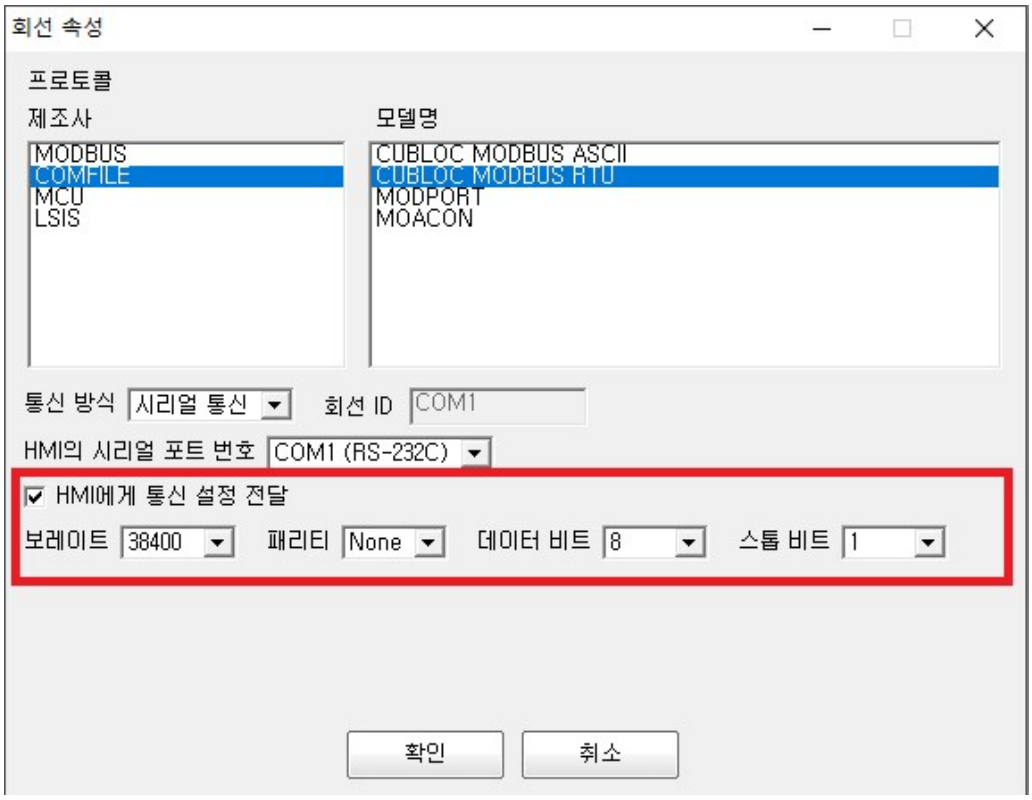
ComfileHMI에서 새 프로젝트를 생성할 때 외부장치 및 프로토콜을 제조사는 COMFILE, 모델명은 CUBLOC MODBUS RTU를 선택하여 생성합니다.



프로젝트를 생성한 이후라면 ComfileHMI에서 왼쪽에 위치해 있는 프로젝트 트리의 프로젝트 속성을 실행하여 아래와 같이 COMFILE 제조사의 CUBLOC MODBUS RTU로 변경하시기 바랍니다.



MP STUDIO에서 설정한 Baudrate, Data Bit, Stop Bit, Parity를 위에서 표기했던 화면속성 창에서 하단의 HMI에게 통신 설정 전달을 체크하면 나타나는 속성박스에 적용합니다.

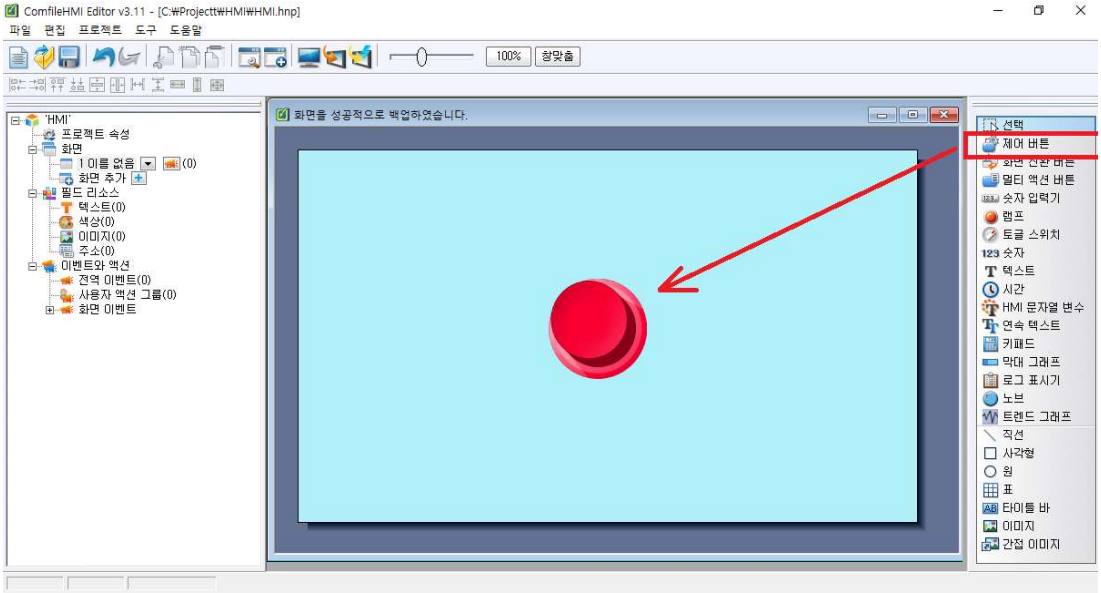


통신설정을 마치면 화면을 추가하여 HMI의 화면구성 및 동작을 프로그램할 수 있습니다. ComfileHMI 사용설명서를 참조해 주시기 바랍니다.

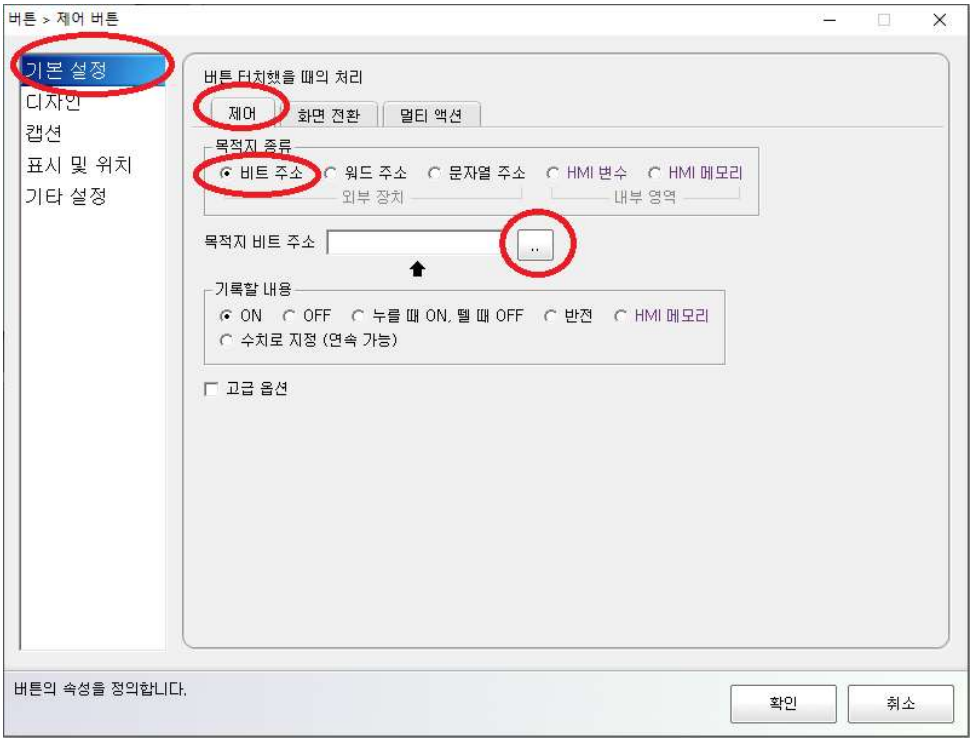
8.2.3. Example

Ⅰ. 제어버튼

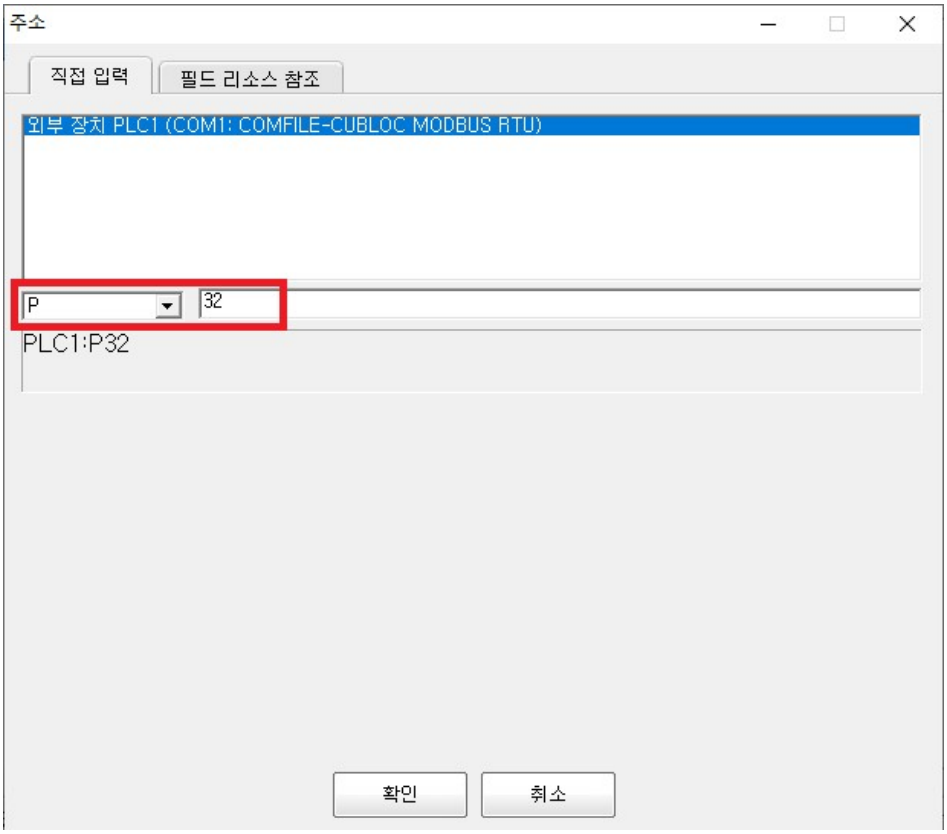
우측의 제어버튼을 선택하고 그리고자 하는 화면의 위치에 Drag&Drop하여 제어버튼을 생성합니다.



제어버튼을 더블클릭하면 나타나는 제어버튼의 속성창의 기본설정에서 아래와 같이 목적지 종류를 비트주소를 선택하고 목적지 비트 주소 우측의 ..버튼을 클릭합니다. 하단의 기록할 내용에서 버튼을 눌렀을 때 MPS의 비트를 어떻게 변경할지 정할 수 있습니다. (On, Off, On->Off, Reverse)



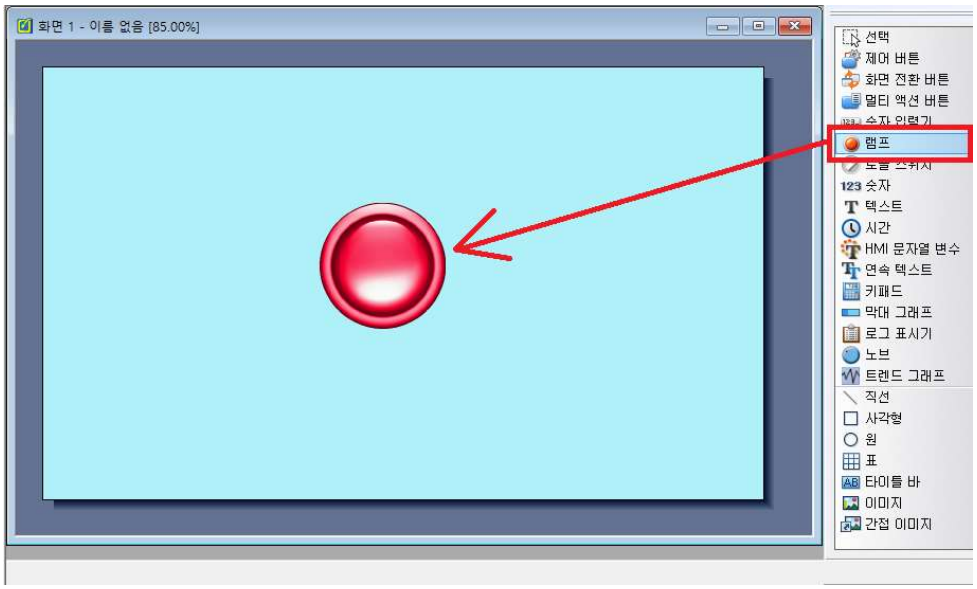
제어하고자 하는 MPS의 비트 메모리를 입력합니다.



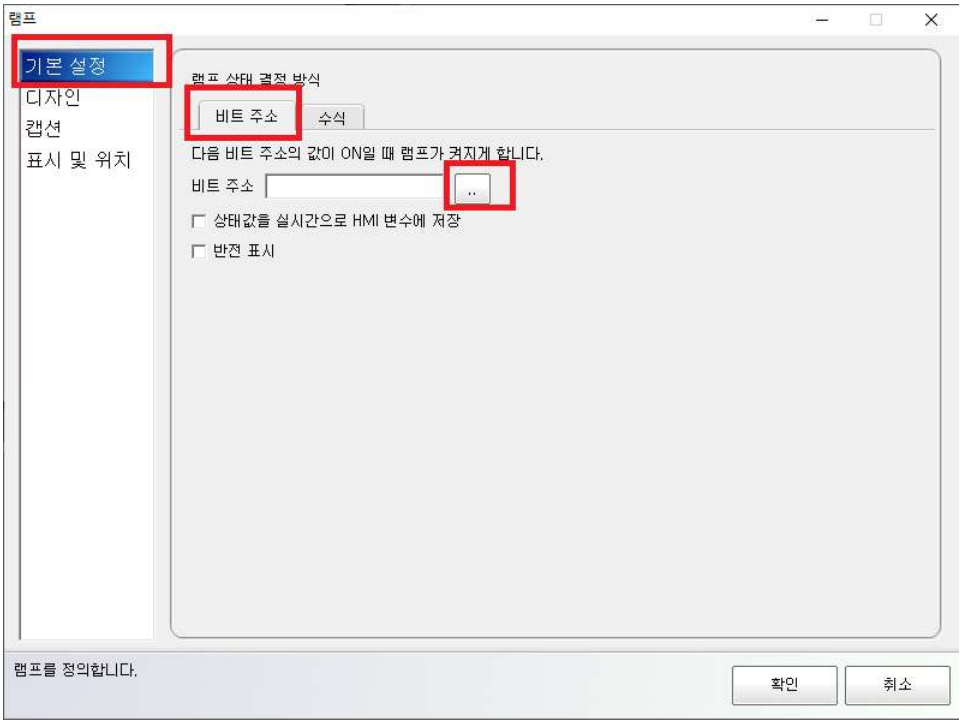
위와 같이 설정하면 제어버튼을 눌렀을 때 MPS제품의 P32 비트를 ON 시킵니다.

Ⅱ. 램프

comfileHMI에서 아래와 같이 우측의 램프를 선택하고 생성하고자 하는 위치에서 마우스로 Drag&Drop하여 램프를 생성합니다.

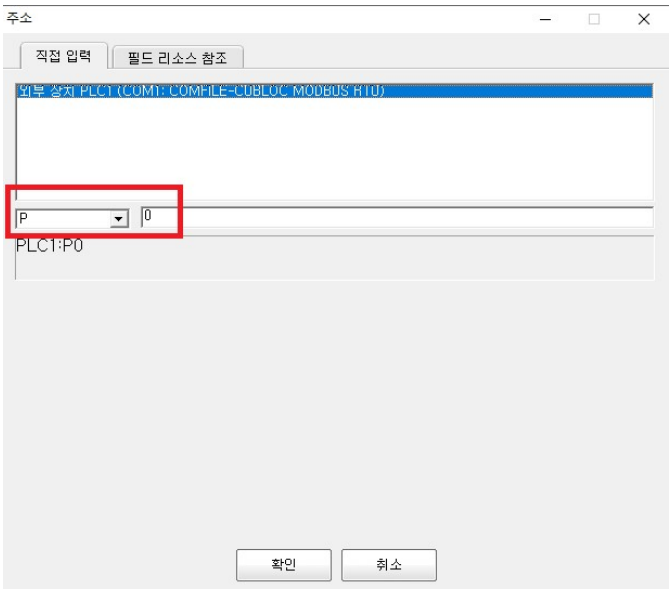


램프를 두 번 클릭하여 램프속성에서 아래와 같이 비트주소의 ... 버튼을 클릭합니다.



MPS 제품의 비트주소가 ON이 되면 램프가 점등되게 하고자하는 비트를 기입합니다.

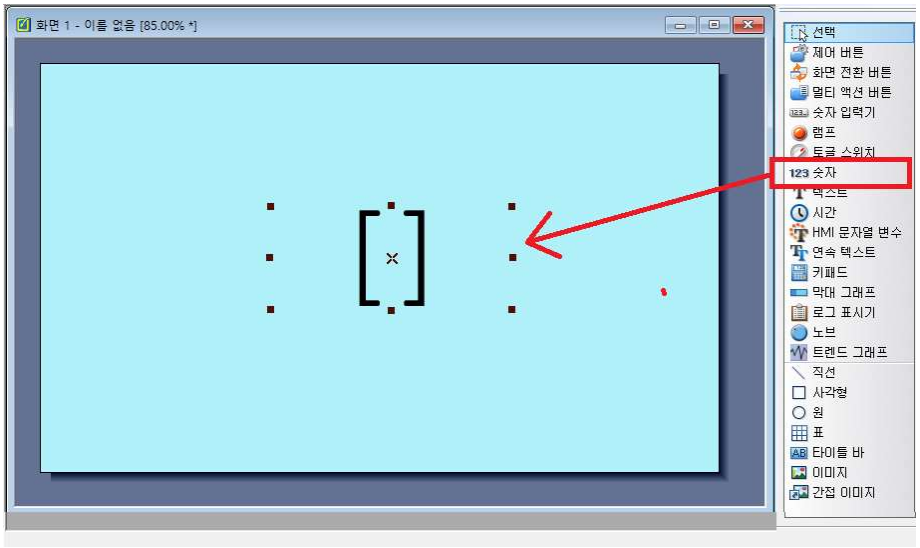
아래에서는 디지털입력 P0가 ON이 되면 램프가 켜집니다.



위와 같이 램프를 사용하여 MPS의 디지털입력 또는 특정비트의 On/Off를 표시할 수 있습니다.

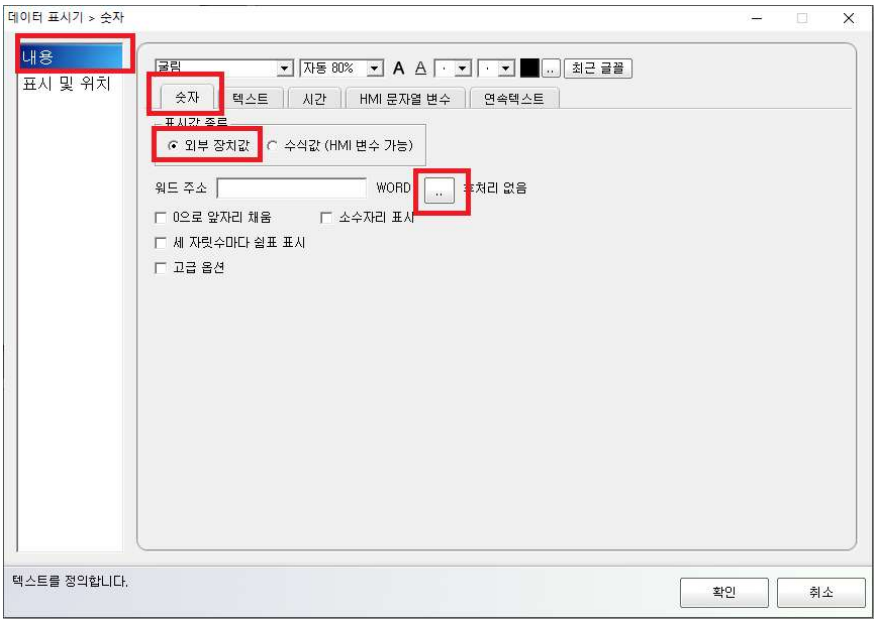
Ⅲ. 숫자 표시기

ComfileHMI에서 아래와 같이 우측의 숫자를 클릭하고 숫자 표시기를 생성하고 하는 위치에서 마우스로 Drag&Drop합니다.

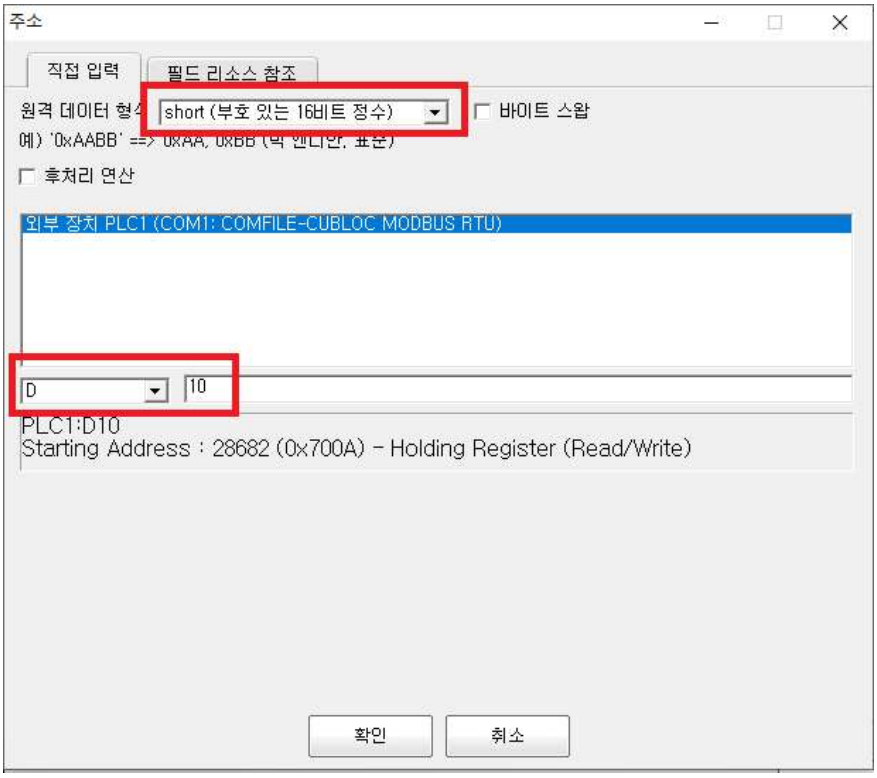


숫자 표시기를 더블클릭하여 나타나는 속성창에서 아래와 같이 선택하고 워드주소의 ..버튼을

클릭합니다.



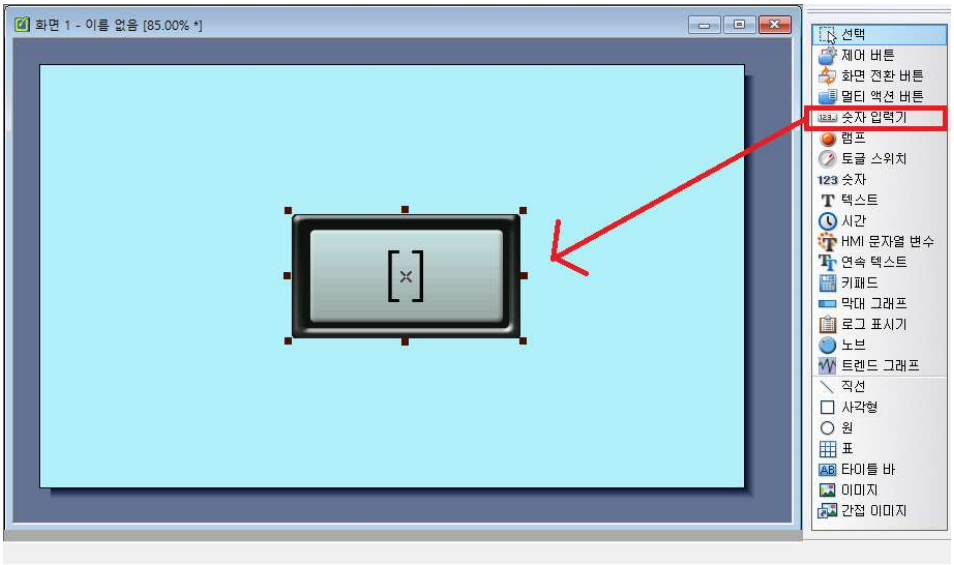
아래와 같이 원격 데이터 형식을 short(부호 있는 16비트 정수)로 선택하고 표시하고자 하는 워드 어드레스를 입력합니다. 그림에서는 D10을 표시하고 있습니다.



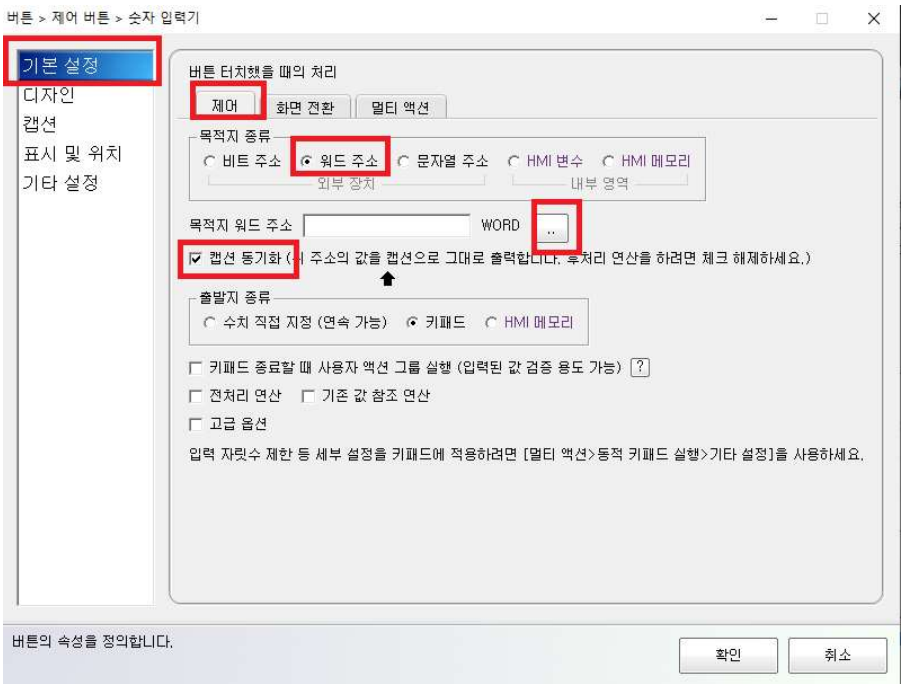
위와 같이 MPS의 워드 메모리값의 숫자를 표시할 수 있습니다. MPS에서 워드값을 읽어 소수점 처리하거나 수학적 연산이 필요할 경우에는 속성창에서 다양한 옵션을 제공하고 있으니 이를 활용해 주시기 바랍니다. 옵션 사용법은 ComfileHMI 사용설명서를 참조해 주시기 바랍니다.

IV. 숫자 입력기

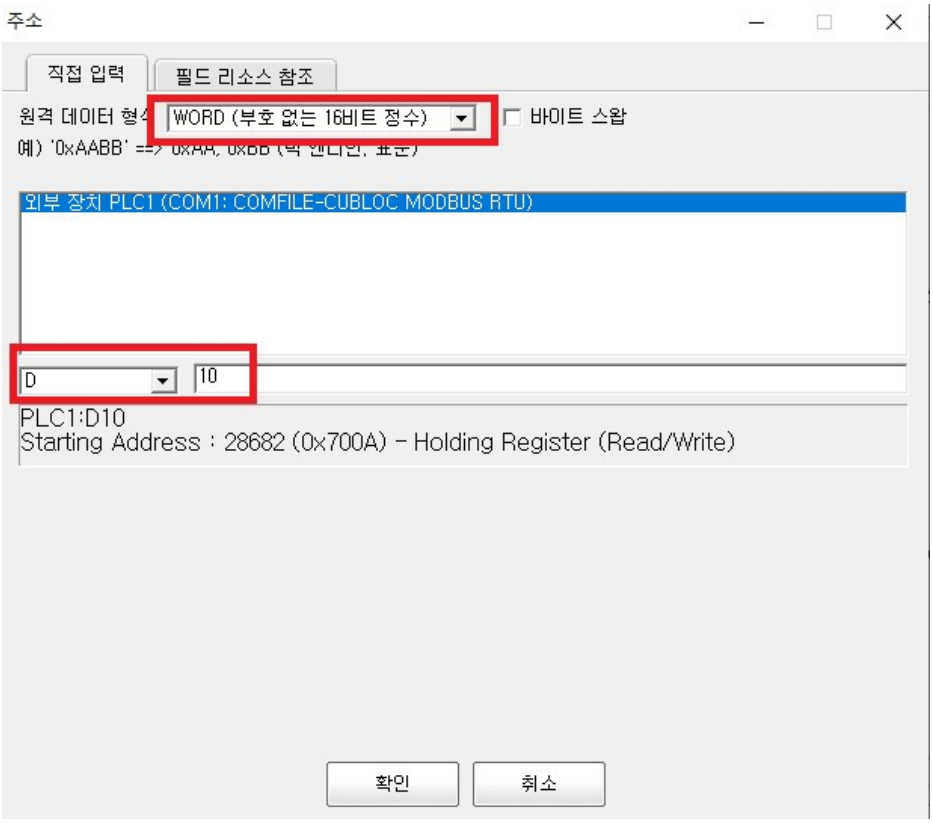
ComfileHMI에서 아래와 같이 우측의 숫자 입력기를 선택하고 숫자 입력기를 생성하고자 하는 위치에서 마우스를 Drag&Drop하여 숫자 입력기를 생성합니다.



숫자 입력기를 더블클릭하면 나타나는 속성창에서 아래와 같이 선택하고 목적지 워드 주소의 우측에 있는 ..버튼을 클릭합니다.



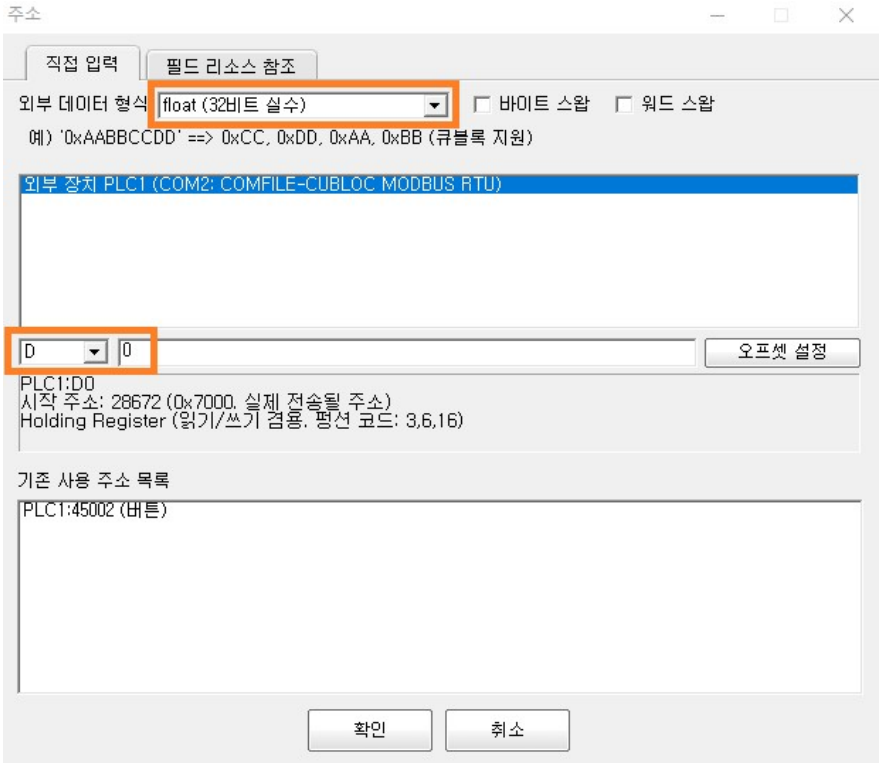
입력하고 하는 MPS 제품의 메모리 정보를 입력합니다. 아래는 원격 데이터 형식을 WORD(부호 없는 16비트 정수)로 하고 메모리값을 D10일때를 표현하였습니다.



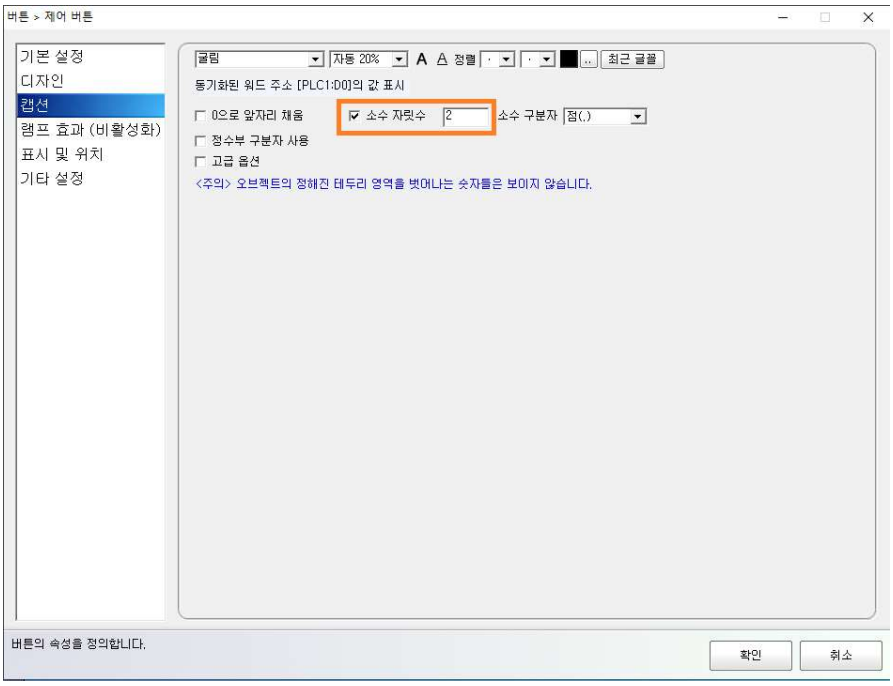
위와 같이 MPS 제품의 메모리값에다 특정 수치를 HMI를 통하여 입력할 수 있고 그 정보를 캡션 동기화 하여 HMI에 표시시킬 수 있습니다.

IV. 실수 숫자 입력기

ComfileHMI에서 숫자 입력기를 화면에 생성하고, 속성창에서 아래와 같이 설정합니다.
IV. 숫자 입력기에서 설명한 내용과 같지만 외부 데이터 형식을 float (32비트 실수)로 선택한 것이 다릅니다.



소수점의 자리수는 아래와 같이 캡션의 "소수 자리수"를 조정할 수 있습니다.



MP STUDIO에서는 아래와 같이 NTEMP 펑션블럭으로 0번 채널의 온도값을 실수 메모리 R0에 저장하고 R0 메모리를 더블워드 메모리 DD0으로 MOVE하여 DD0(D0,D1) 메모리에 실수 값을 넣어야 합니다. DD0 다음은 DD2를 사용해야 합니다. 더블워드 DDn 메모리는 짝수단위로 사용하는 것을 권장드립니다. DD2를 사용할 경우, ComfileHMI에서의 주소값은 D2입니다.

DD0의 모니터링 값은 처음에는 더블워드 값으로 보이기 때문에 "1103835955"로 보이지만 마우스 우클릭으로 Floating Point 표현법으로 선택하면 25.4로 보이게 됩니다.

지금까지 간단한 예를 나타내보았습니다. 보다 자세한 사용방법과 통신설정에 대해서는 ComfileHMI 사용설명서를 참조해 주시기 바랍니다.

8.3. CNET (XBC/LINK)

CNET은 LS산전에서 개발한 프로토콜로서 LS산전의 PLC와 HMI를 연결하기 위해 개발되었지만 여러 HMI 제조사들이 LS산전의 PLC와 연결할 수 있도록 CNET 프로토콜을 지원하고 있습니다.

MPS 및 MPA 제품군의 RS232, RS485, UART 모든 통신포트에서 CNET (XBC/LINK) 슬레이브 프로토콜을 지원합니다. 따라서, LS산전 HMI 또는 CNET을 지원하는 다양한 HMI에서 연결이 가능합니다. 아래 8.4 항목에서 M2I HMI를 이용하여 CNET통신의 설명을 참조해주시기 바랍니다.

8.3.1. XBC

LS산전은 초기에 Master-k와 Glofa PLC의 2가지 형식이 있었습니다.
Master-k는 미쯔비시 형식과 흡사하고 메모리타입이 M00, D0와 같은 방식이며,
Glofa는 지멘스 형식과 흡사하고 메모리타입이 %MX100, %DW0, %IX0.0.0과 같은 방식입니다.

현재는 Master-K PLC가 성능에 따라서 XGK/MBM/XBC PLC로 출시되었으며,
Glofa PLC는 성능에 따라서 XGI/XGR/XEC PLC로 출시되었습니다.

위의 메모리타입을 보시면 저희 MPS도 Master-K와 비슷한 방식의 메모리 타입이기 때문에 Master-K,XGK,XBM,XBC 타입의 CNET LINK를 지원합니다.

8.3.2. LINK

CNET은 LS산전의 CPU모듈에 직접 연결해서 사용하는 DIRECT와 통신채널에 연결해서 사용하는 LINK 타입이 있습니다.

DIRECT는 LS산전의 소프트웨어인 XG5000의 모니터링 용도로 개발된 통신으로서 CPU에 다이렉트로 연결할 수 있어 별도의 통신모듈 없이 사용할 수 있는 장점이 있었지만, 현재의 XGB제품들은 CPU모듈에 LINK타입의 CNET을 지원하므로 굳이 DIRECT를 사용하지 않아도 됩니다.

LINK는 LS산전의 PLC와 HMI를 연결하기 위하여 개발된 통신입니다. LS산전의 인지도가 높기 때문에 여러 HMI 브랜드 제품에서도 CNET LINK 타입을 지원하여 LS산전의 PLC 타제품의 HMI를 쉽게 연결하여 사용할 수 있습니다.

8.3.3. BCC 체크섬 사용

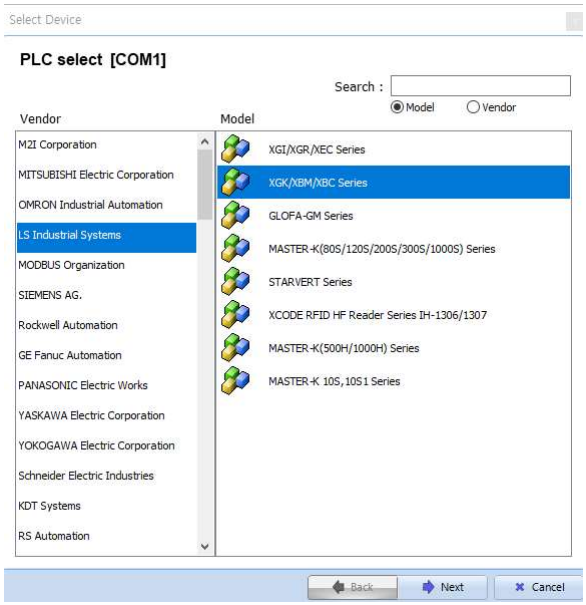
HMI에서 CNET 프로토콜에 BCC 체크섬을 사용할지 안할지를 결정할 수 있습니다. 자사의 제품은 BCC 체크섬을 사용하는 프로토콜을 지원하므로 체크섬을 사용해 주시기 바랍니다.

8.3.4. M2I를 사용할 경우

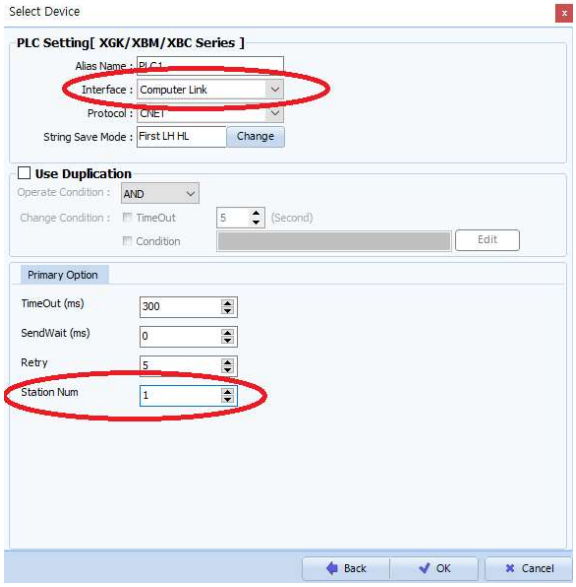
M2I 기업의 HMI를 저희 MPS제품과 CNET으로 연결하기 위하여 M2I에서 제공하는 TOP Design을 사용하는 예를 설명하도록 하겠습니다.

Ⅰ. 아래와 같이 M2I HMI와 연결할 XBC PLC를 선택

프로젝트 생성 --> Display 모델명 선택 --> COM포트 클릭 --> ADD버튼 클릭 --> LS Industrial Systems의 XGK/XBM/XBC Series 선택

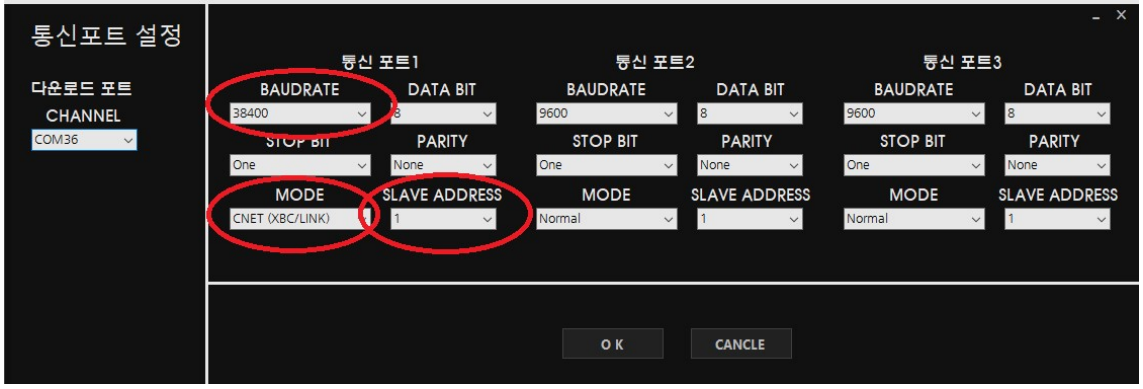


Ⅱ. Interface를 Computer Link로 선택하고 Station Num을 MPS 제품의 슬레이브 넘버(1~255)를 선택합니다. MPS의 슬레이브 넘버는 MP STUDIO 통신포트 설정에서 확인 및 설정이 가능합니다.



Ⅲ. MP STUDIO 통신포트 설정

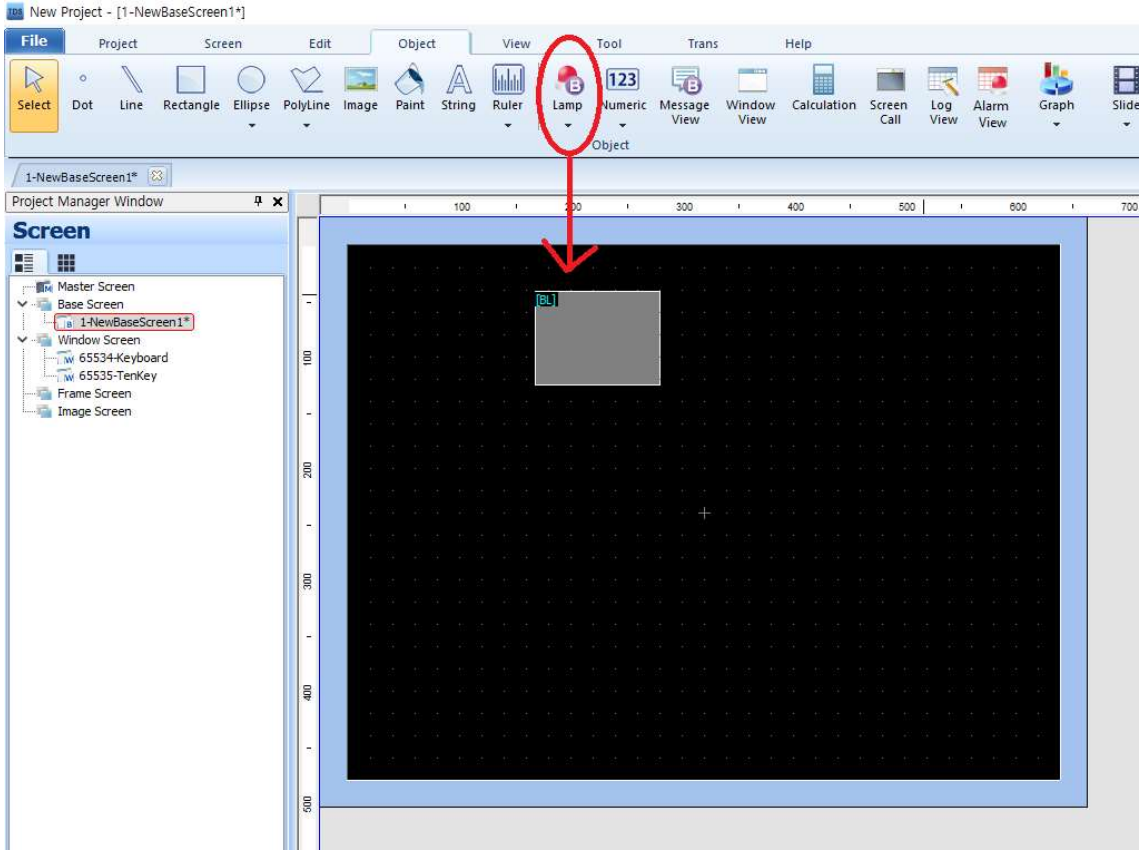
MP STUDIO 통신포트 설정창에서 HMI와 연결하고자 하는 통신포트 채널의 BAUDRATE를 38400, MODE를 CNET(XBC/LINK), SLAVE ADDRESS를 1로 변경합니다.



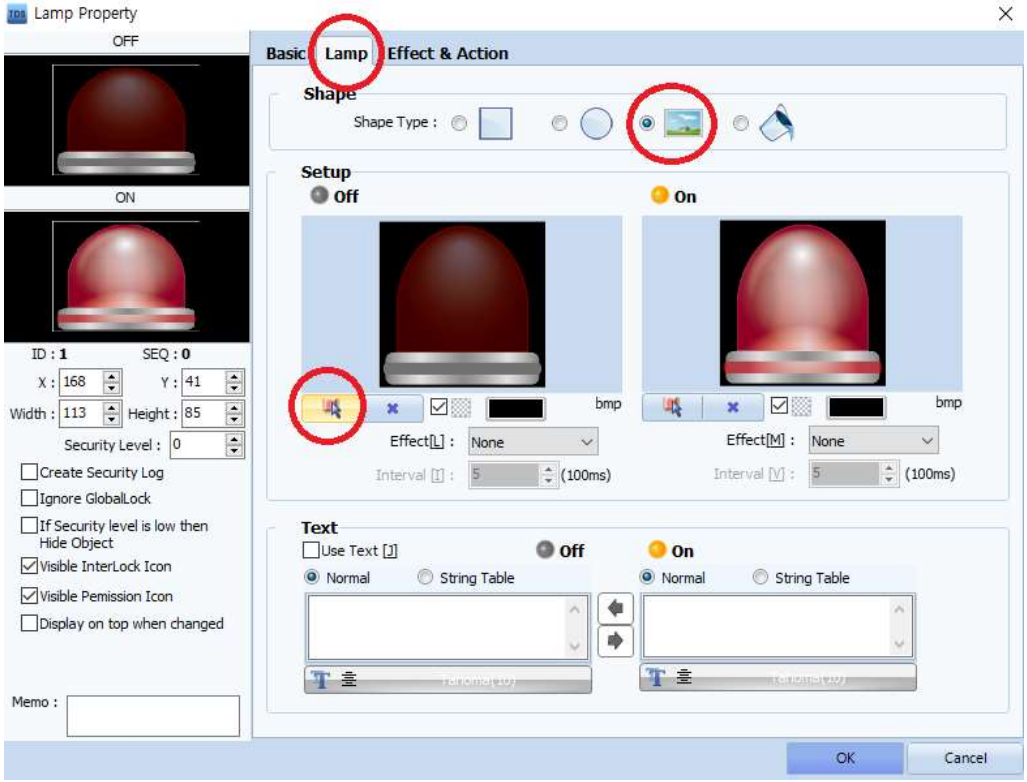
변경한 통신포트 설정은 프로그램 다운로드를 실행할 때 MPS제품에 적용됩니다.

Ⅳ. 비트 메모리 제어

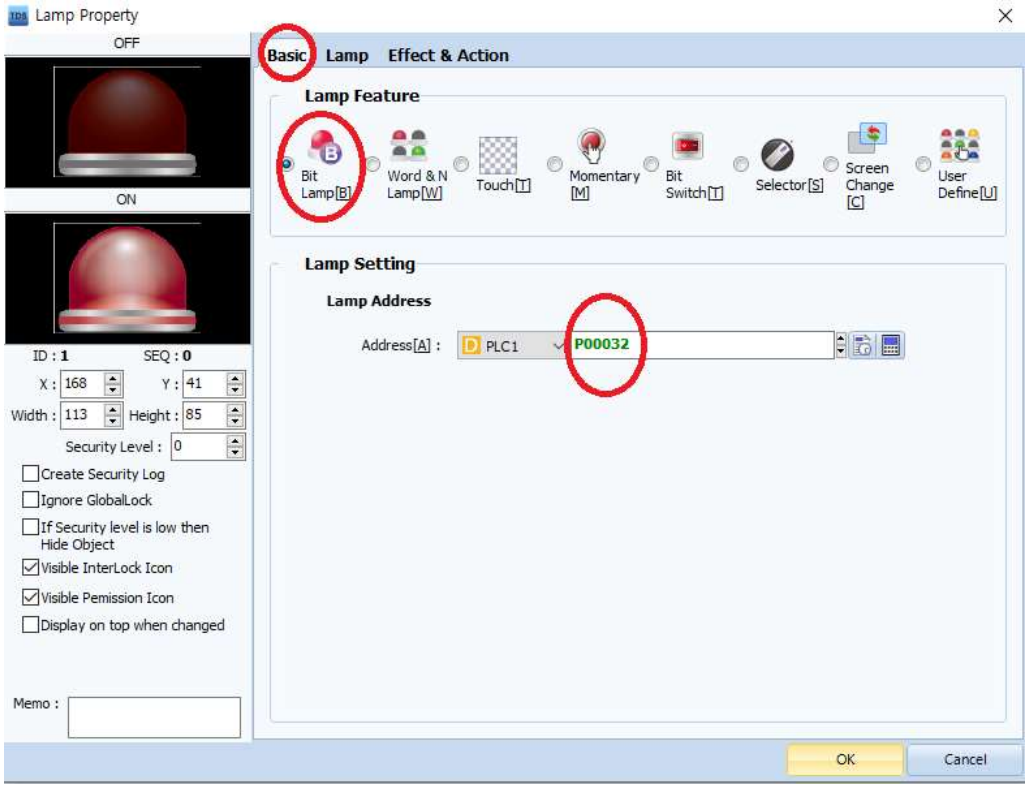
오브젝트 탭에서 램프를 선택하고 마우스로 Drag&Drop하여 작화창의 원하는 위치에 램프 오브젝트를 생성합니다.



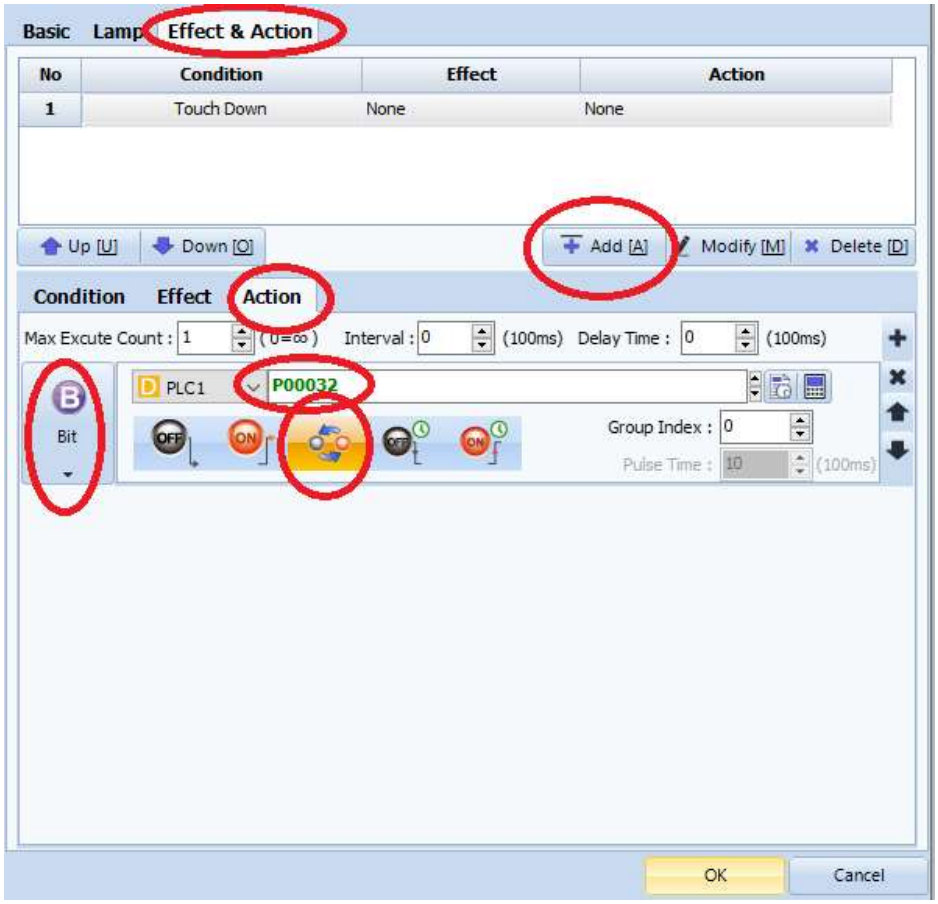
램프를 더블클릭하여 나타나는 속성창에서 아래처럼 램프의 이미지를 변경합니다.



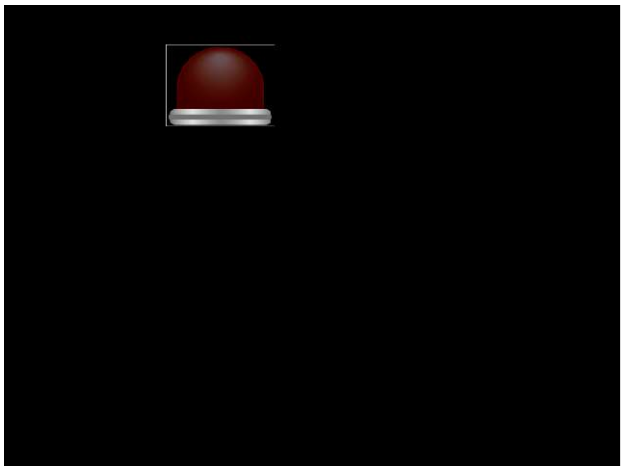
Basic탭에서 Address 입력창에 P32를 입력합니다.



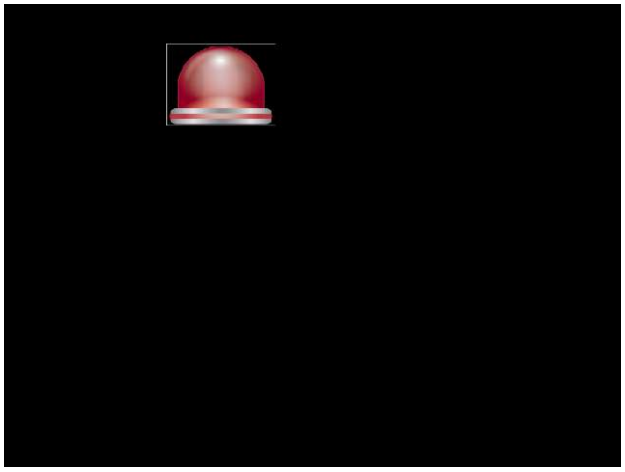
Effect&Action탭에서 ADD를 누르고 BIT를 선택하고 하위 ACTION탭에서 P32를 입력하고 Reverse를 선택합니다.



HMI를 동작시키면 최초에 램프는 OFF되어 있고, 램프를 클릭하면 램프가 점등되면서 MPS 제품의 P32(릴레이출력)가 ON됩니다. (MPS-8A4R 기준으로 P32는 릴레이 출력입니다.)

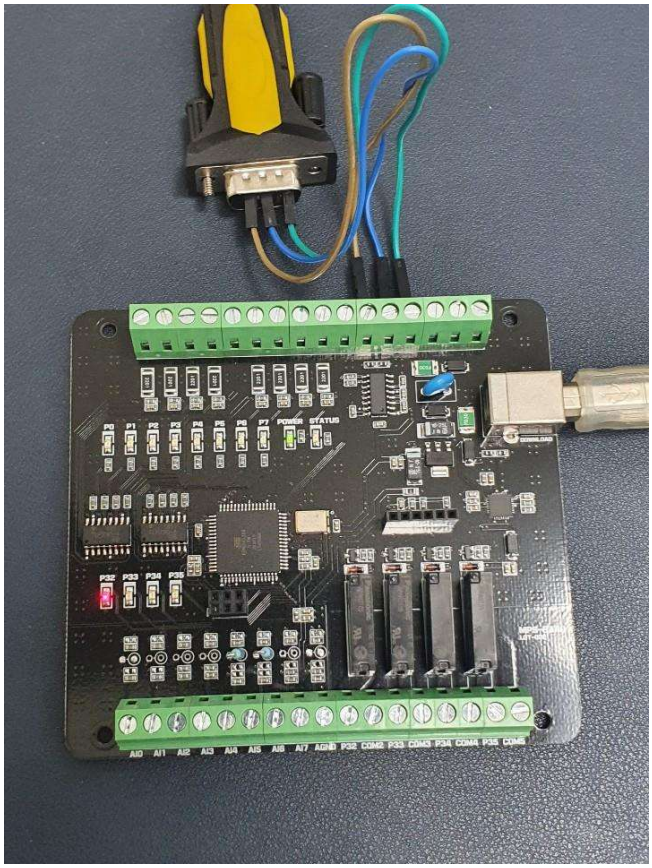


< 램프:P32가 소등된 상태 >



< 클릭하여 램프:P32가 점등된 상태 >

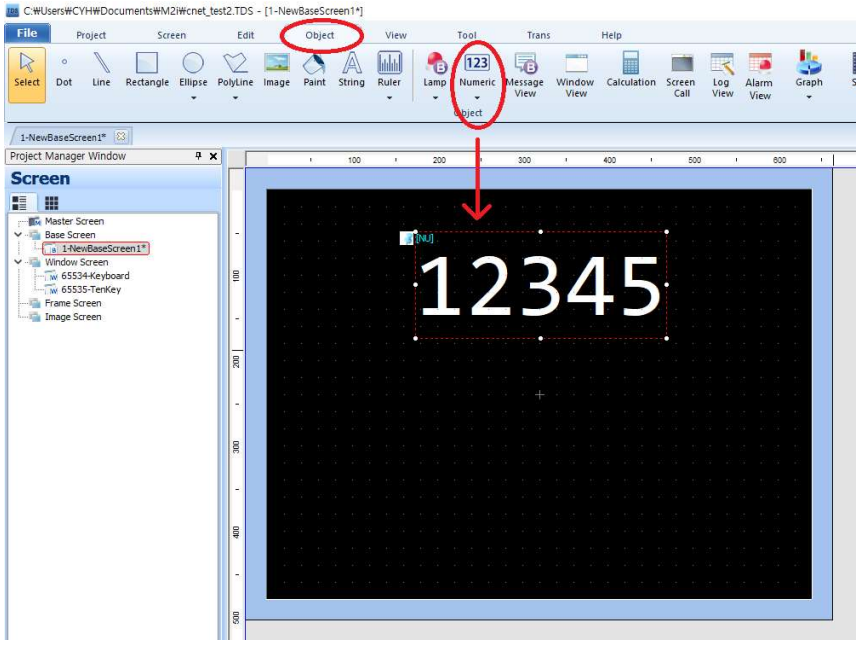
MPS-8A4R 제품은 아래와 같이 RS232를 PC와 연결하였으며 램프가 점등될 때 P32 출력 LED도 같이 점등됩니다.



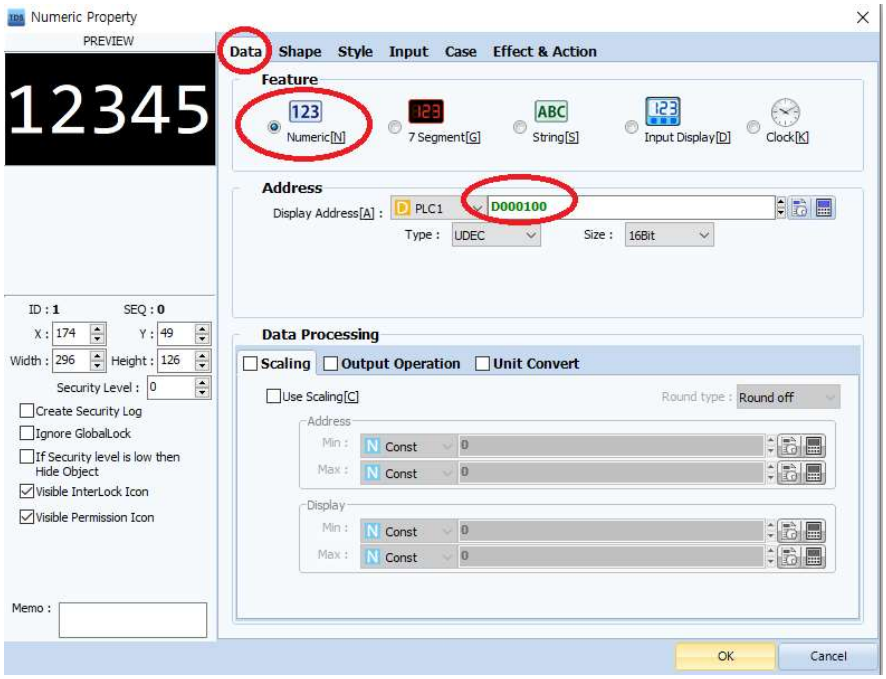
V. 워드 메모리 제어

MPS제품의 D100 워드 메모리에는 50을 입력하고 D200 워드 메모리에는 -25를 입력해 보겠습니다.

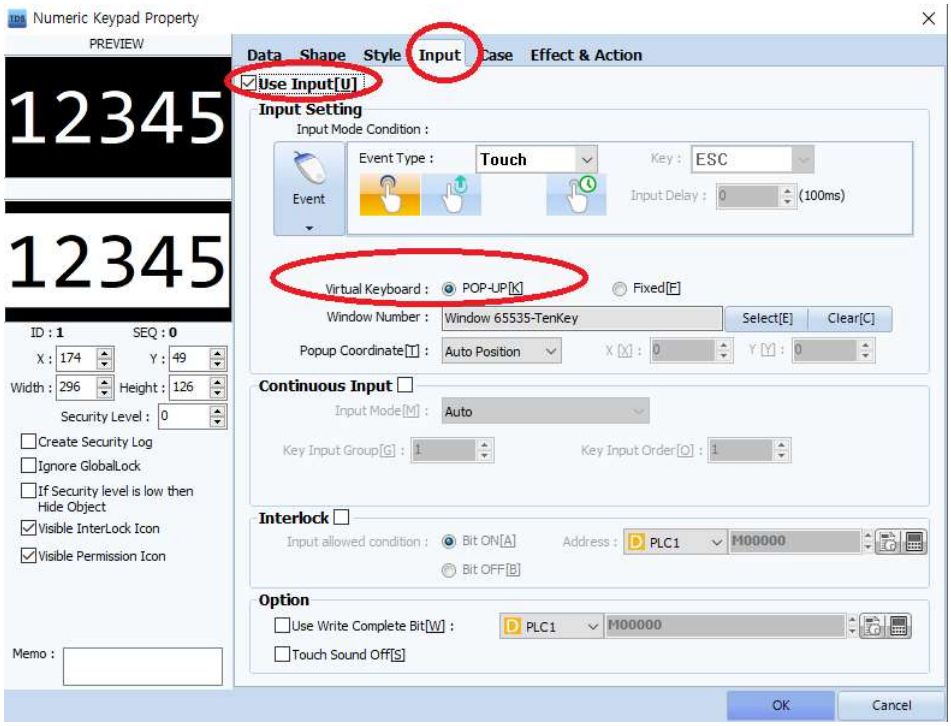
M2I의 TOP Design 소프트웨어에서 Object탭의 Numeric을 작업창의 생성하고 하는 위치에 마우스로 Drag&Drop합니다.



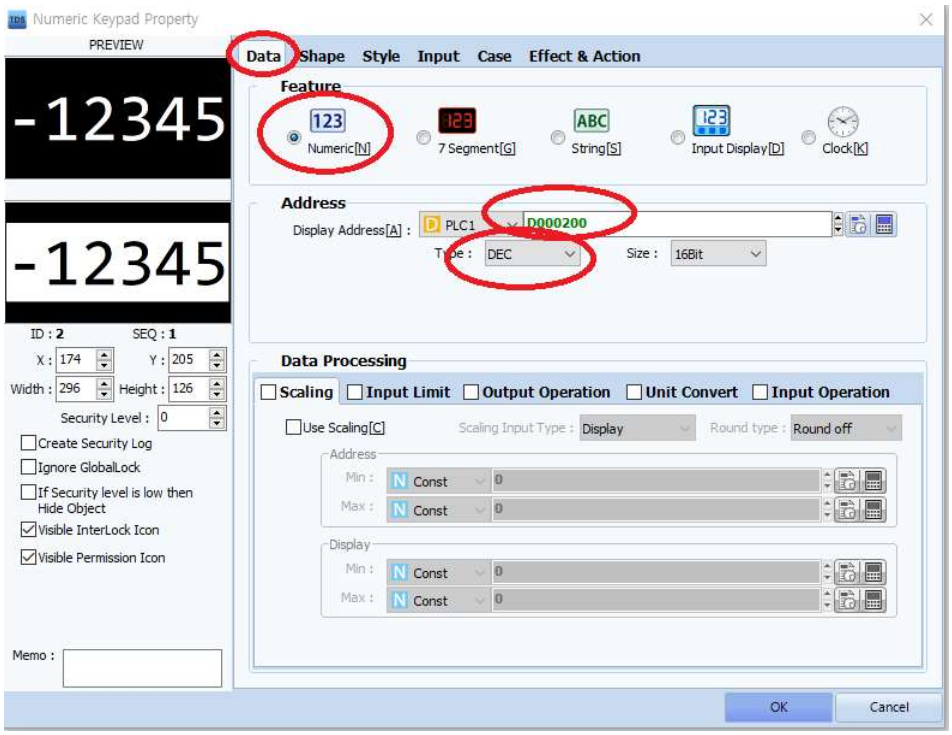
Numeric을 더블클릭하여 속성창을 띄우고 아래처럼 Address에 D100을 입력합니다.



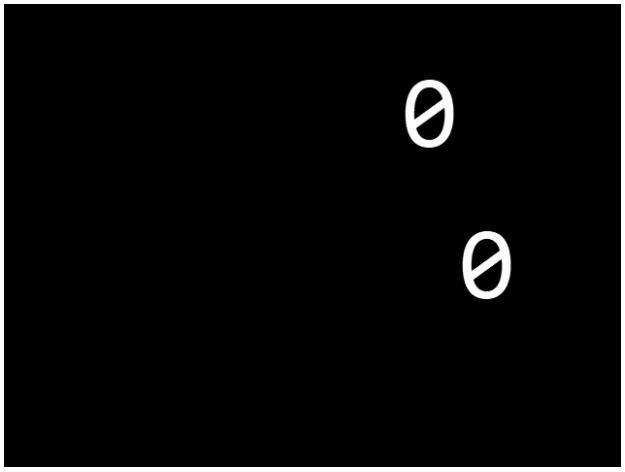
Input탭의 Use Input 체크박스를 체크하고 Virture Keyboard의 Pop-up을 선택하여 숫자를 클릭하면 키패드가 생성되도록 설정합니다.



위와 같은 방식으로 Numeric 오브젝트를 하나 더 생성시키고 Data탭에서 Address를 D200으로 Type을 DEC로 선택합니다.



HMI를 동작시키면 두 개의 Numeric 오브젝트는 0이 표시됩니다. 이후, D100의 Numeric에는 50을 입력하고 D200의 Numeric에는 -25를 클릭하여 키패드에 입력하면 두 개의 Numeric은 각각 50, -25로 표기됩니다.



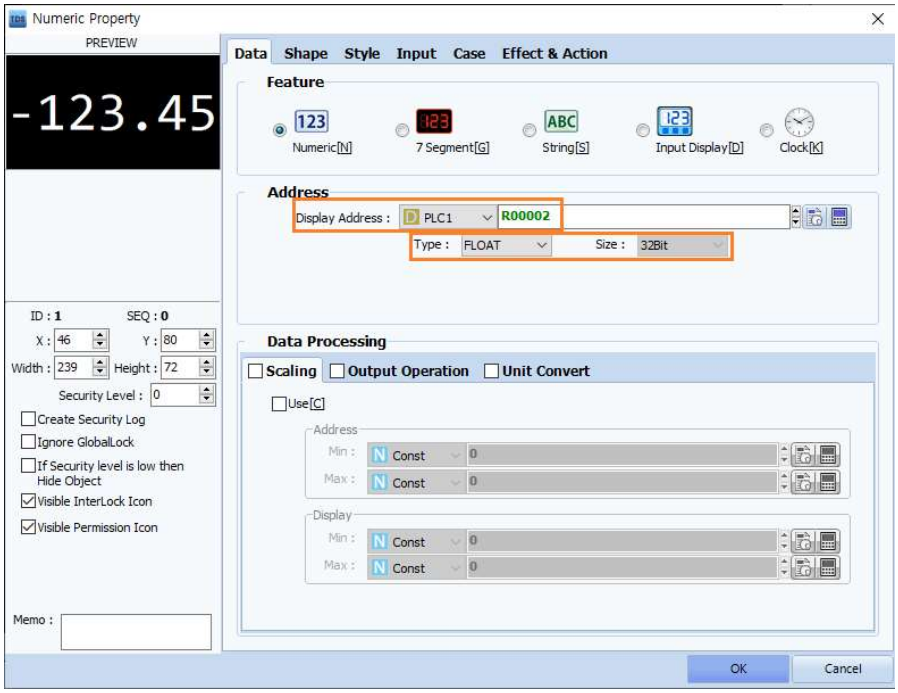
<HMI 실행시 초기 D100,D200 값>

<키패드를 입력하여 변경된 D100,D200 값>

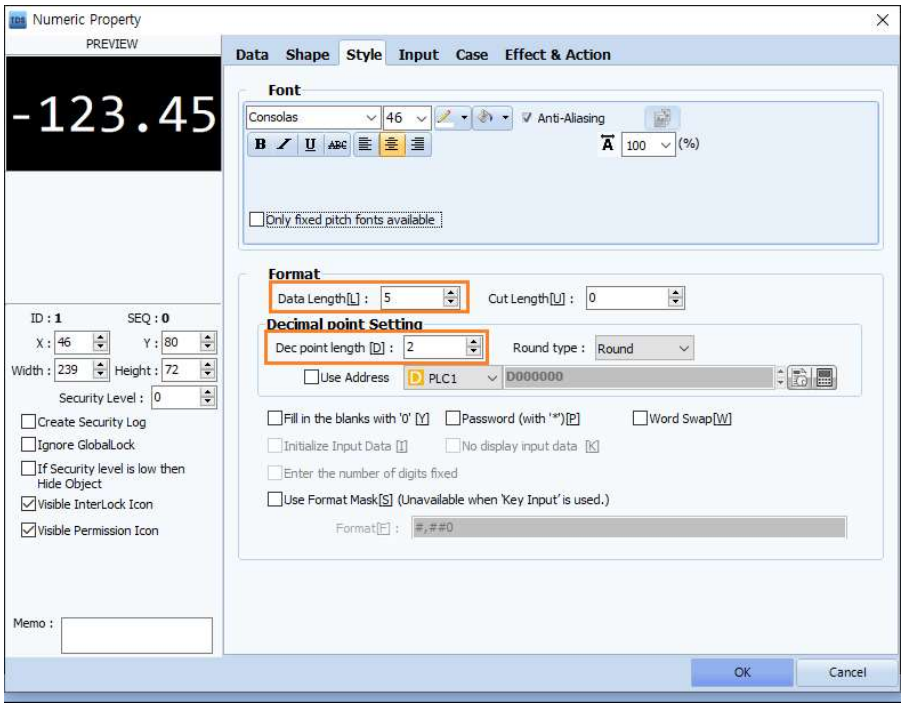
또한, MPS제품의 D100과 D200의 워드 메모리에도 각각 50, -25가 입력되어 아래와 같이 모니터링으로 확인이 가능합니다.

VI. 실수 메모리 읽기

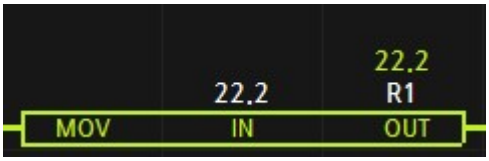
Numberic 오브젝트에서 표시하고자 하는 R 메모리에 곱하기 2를 하여 Display Address에 기입해 주시기 바랍니다. 또한, Type : Float, Size : 32bit로 설정해 주시기 바랍니다.
(아래는 R1 메모리의 값을 읽기 위한 설정입니다)



Style 탭의 Data Length(총길이)와 Dec Point Length(소수점 길이)를 조절하여 소수점 자리수를 설정할 수 있습니다.



MP STUDIO에서 R1 메모리의 값이 22.2일 때, MP STUDIO 와 M2I의 모니터링의 상태는 아래와 같습니다. (M2I의 Address는 위에서 언급했듯이 R1 *2 = R2로 설정되었습니다)

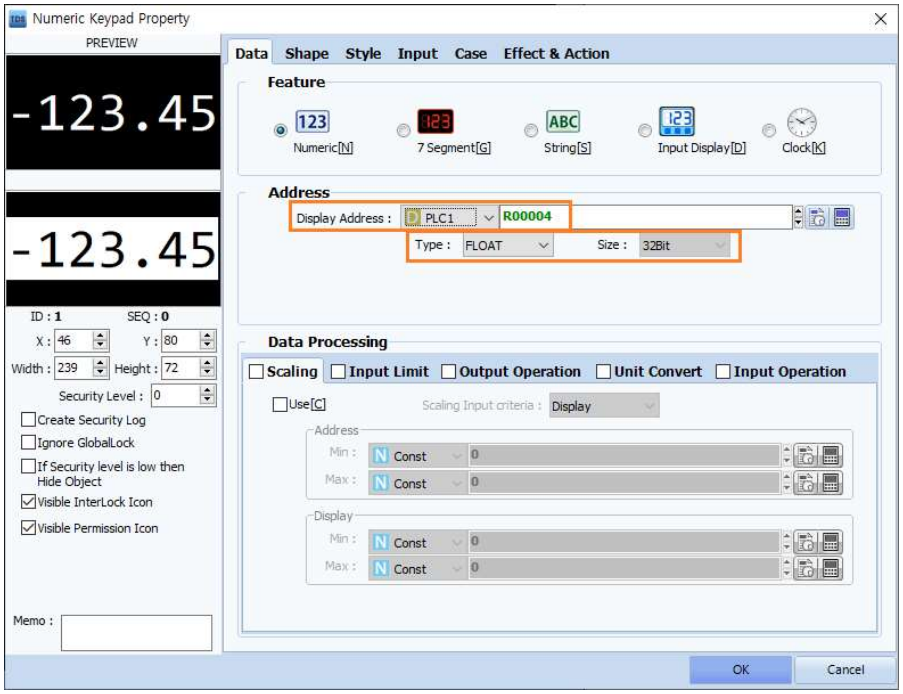


< MP STUDIO >

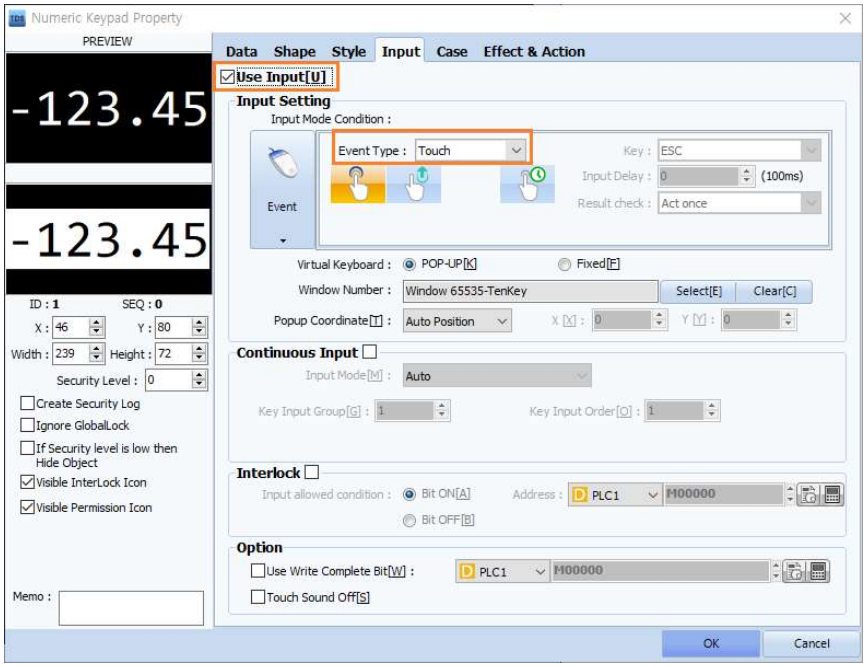
< M2I >

VI. 실수 메모리 쓰기

Numberic 오브젝트에서 쓰기하고자 하는 R 메모리에 곱하기 2를 하여 Display Address에 기입해 주시기 바랍니다. 또한, Type : Float, Size : 32bit로 설정해 주시기 바랍니다.
(아래는 R2 메모리에 값을 쓰기하기 위한 설정입니다)



키보드를 이용하여 값을 입력하고자 하면, input탭에서 use input을 체크하고 Event Type을 Touch로 설정해 주시기 바랍니다.



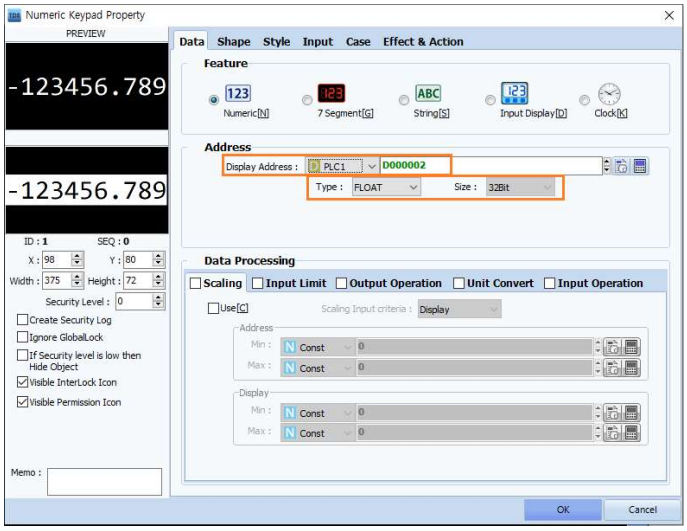
M2I에서 12.8을 입력하여 MP STUDIO에서 R2 메모리의 값이 12.8로 변경되었을 때, MP STUDIO와 M2I의 모니터링의 상태는 아래와 같습니다.
(M2I의 Address는 위에서 언급했듯이 R2 *2 = R4로 설정되었습니다)

< MP STUDIO >

< M2I >

VI. 실수 메모리 읽기 (더블워드 메모리 이용)

더블워드 메모리를 이용하여 실수값을 읽고 쓰기할 수 있습니다.
(아래는 워드메모리 DD2를 사용하여 실수값을 표시하기 위한 Numberinc 오브젝트 설정입니다)



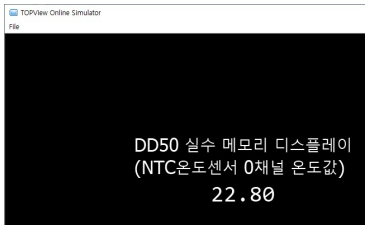
MP STUDIO에서 DD2 메모리의 값이 987654.321 일 때, MP STUDIO 와 M2I의 모니터링의 상태는 아래와 같습니다.

< MP STUDIO >

< M2I >

MP STUDIO에서 NTC온도센서 0채널의 온도값을 NTEMP 펑션블럭으로 읽어와서 실수 메모리 R0에 저장합니다. (모니터링상에서 R0는 22.8이므로 22.8도를 의미합니다) 이후, R0 값을 더블워드 메모리 DD50으로 옮깁니다. (DD50은 D50, D51 메모리 영역을 사용합니다) DD50의 값을 HMI에서 읽어와서 디스플레이하게 됩니다.

아래와 같이 M2I HMI에서 22.8도를 디스플레이할 수 있습니다.



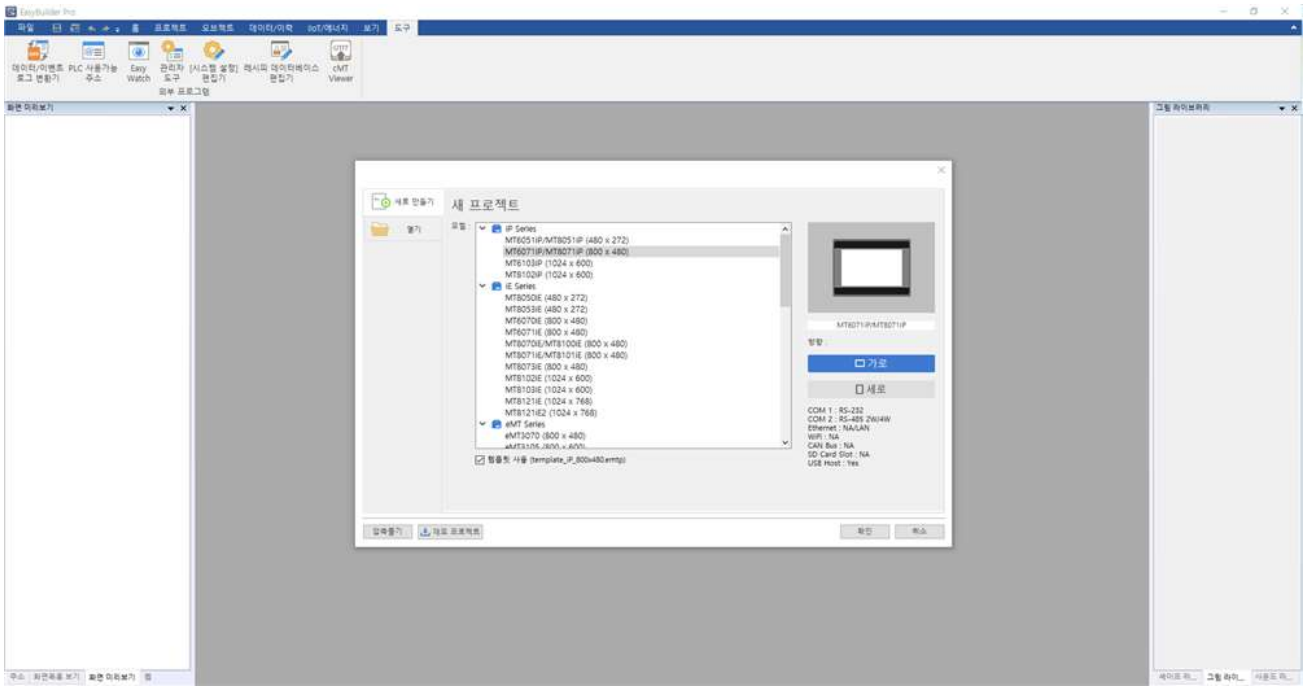
8.3.5. EasyView를 사용할 경우

Ⅰ. EasyView 소프트웨어 및 모델명

- 본 매뉴얼에서는 EasyBuilder Pro V6.04.02.552를 사용하였습니다.
- 본 매뉴얼에서는 MT6071iP/MT807iP (800x480)을 사용하였습니다.

Ⅱ. EasyBuilder Pro 새 프로젝트 생성

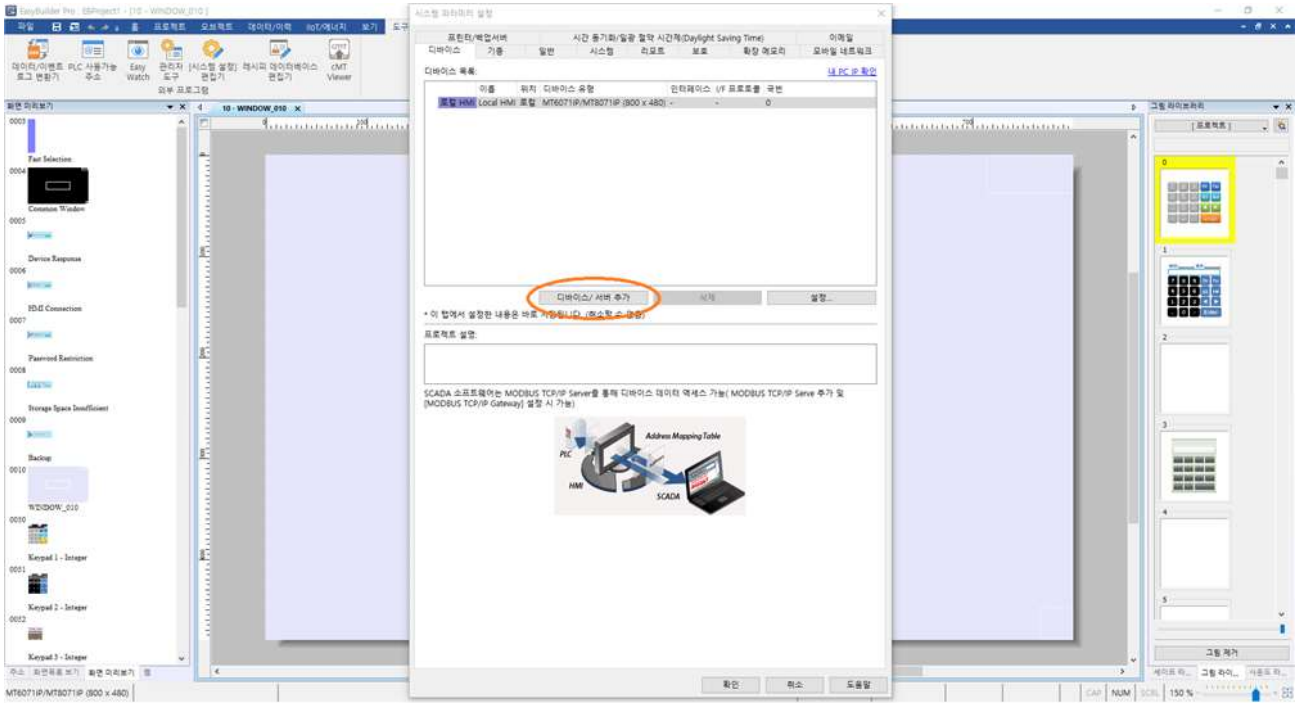
- 새 프로젝트를 실행하여 사용하고자 하는 모델명을 선택합니다. 본 사용설명서에서는 MT6071iP/MT8071iP (800x480)를 기준으로 설명하였습니다.



<그림 1 - EasyBuilder Pro 모델명 선택 >

Ⅲ. EasyBuilder Pro 시스템 파라미터 설정

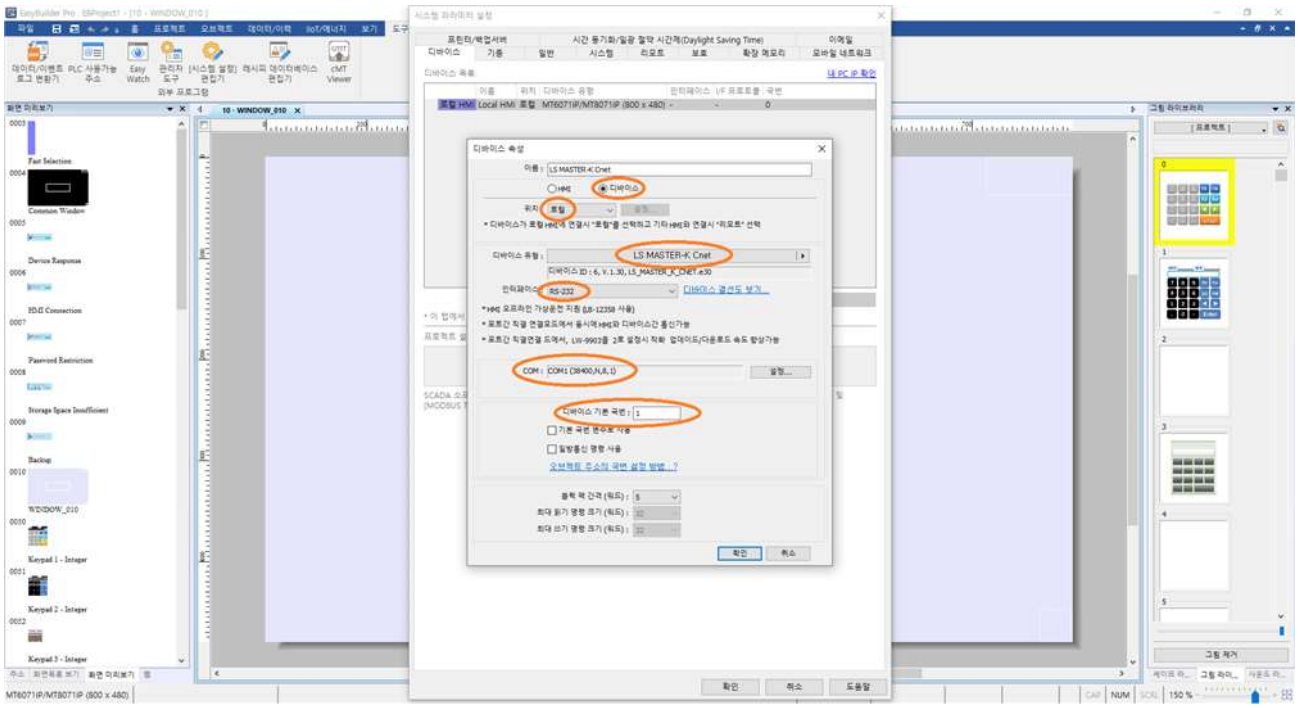
- 통신 프로토콜을 추가하기 위하여 시스템 파라미터 설정에서 "디바이스/서버 추가" 버튼을 클릭합니다.



<그림 2 - EasyBuilder Pro 디바이스 추가 >

IV. EasyBuilder Pro 프로토콜 설정

- 하기와 같이 디바이스의 속성값을 변경합니다.
- 위치 : 로컬
- 디바이스 유형 : LS MASTER-K Cnet
- 인터페이스 : 사용하고자 하는 통신 방식을 선택
- COM : 9,600~115,200에서 사용하고자 하는 통신속도를 설정합니다. 본 매뉴얼에서는 38,400을 선택합니다. (통신거리가 길 경우 통신속도를 낮추어 통신 안정성을 높일 수 있습니다.)
- 디바이스 기본 국번 : 현재 설정하고 있는 디바이스의 국번을 1~31중에서 선택합니다. 본 매뉴얼에서는 1번을 선택합니다. (RS485 통신방식을 사용하여 여러대의 PLC와 연결할 경우, Ⅲ. EasyBuilder Pro 시스템 파라미터 설정의 "디바이스/서버 추가" 버튼을 사용하여 연결하고자 하는 PLC의 개수만큼 다비이스를 생성하여 사용해야 합니다.)

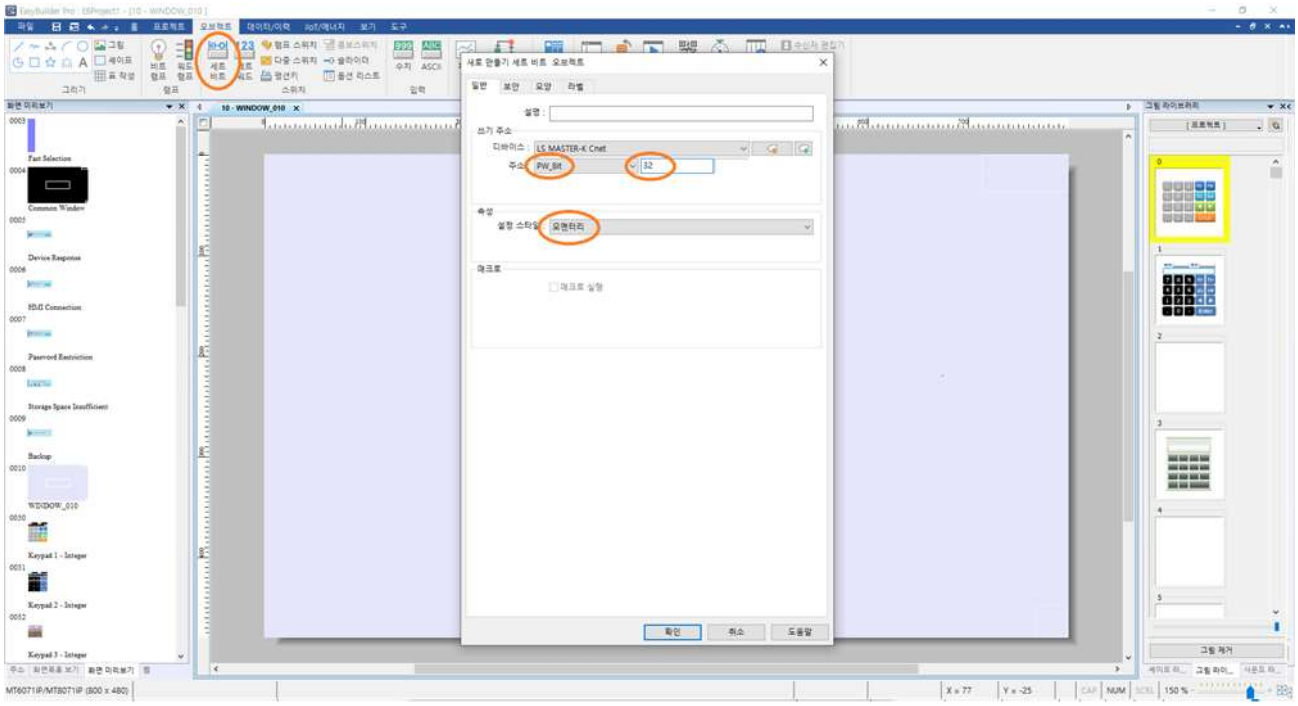


<그림 3 - EasyBuilder Pro 디바이스 속성 설정 >

V. EasyBuilder Pro 세트비트 생성

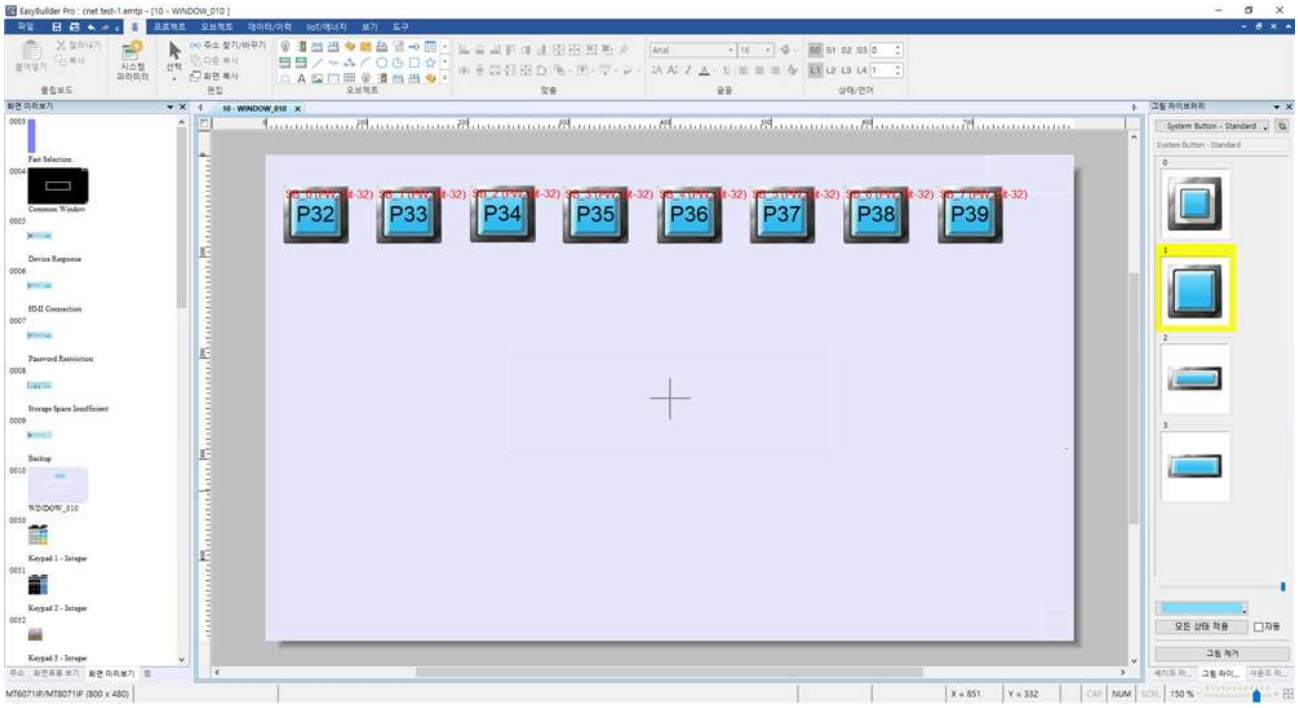
EasyBuilder Pro 작화 화면의 상단 메뉴의 오브젝트 -> 세트비트를 선택하고 읽기주소 파라미터를 변경합니다. 본 매뉴얼에서는 MPA-16A16RX 제품의 출력 릴레이 P32~P39를 제어하는 버튼을 생성하였습니다.

- 디바이스 : 위에서 생성한 디바이스를 선택합니다. RS485 통신방식을 사용하여 여러대의 PLC를 제어하기 위해서는 디바이스를 제어하고자 하는 PLC 이름을 선택해야 합니다.
- 주소 : 비트주소는 PW_Bit와 MW_Bit 만을 지원합니다. P32는 PW_Bit를 선택하고 32를 기입합니다.
- 설정 스타일 : 세트ON(터치시 PLC에 ON을 전달), 세트OFF(터치시 PLC에 OFF를 전달). 모멘터리(터치시 PLC에 ON을 전달하고 터치를 땠을 때 PLC에 OFF를 전달)



<그림 4 - EasyBuilder Pro 세트비트 속성 설정 >

- MPA-16A16RX 제품의 16개의 릴레이접점 P32~P39를 각각 제어하기 위한 버튼을 하기와 같이 생성하였습니다.



<그림 5 - EasyBuilder Pro 세트비트(스위치) 생성 >

- 하기와 같이 P34버튼을 터치시 릴레이출력 P34가 ON됩니다.

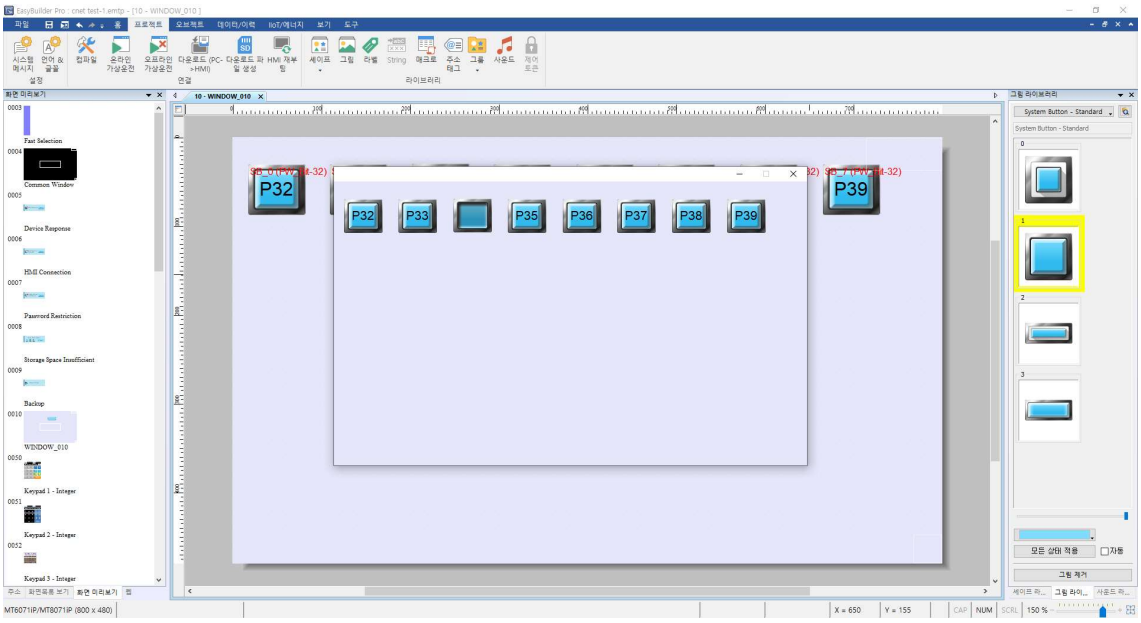


그림 6 - EasyBuilder Pro 세트비트(스위치) 제어 >

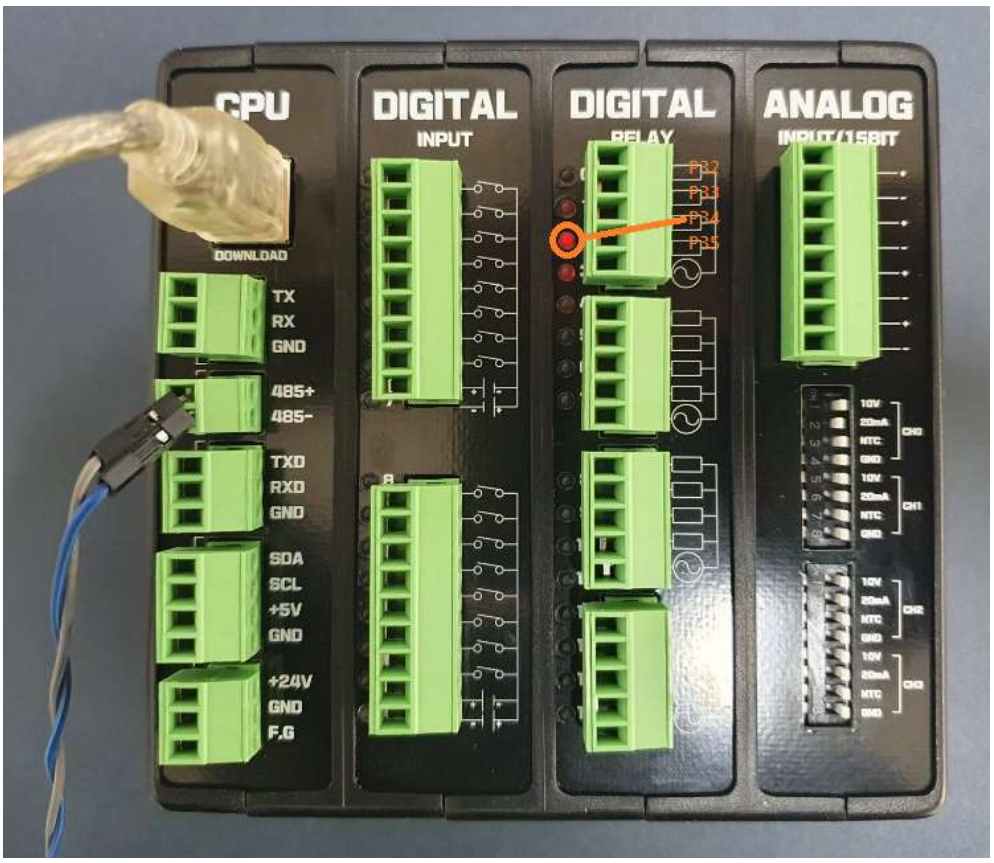
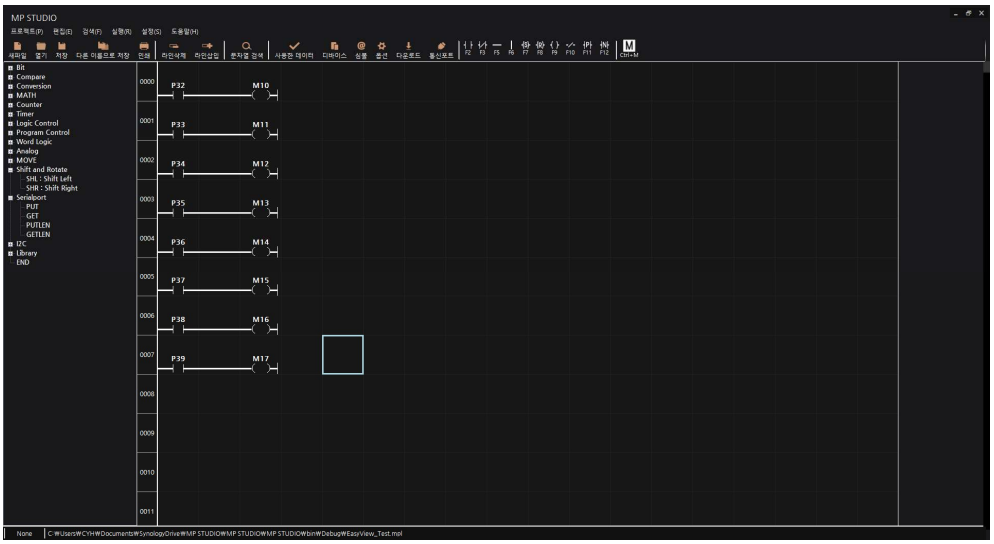


그림 6 - EasyBuilder Pro 세트비트(스위치) 제어 >

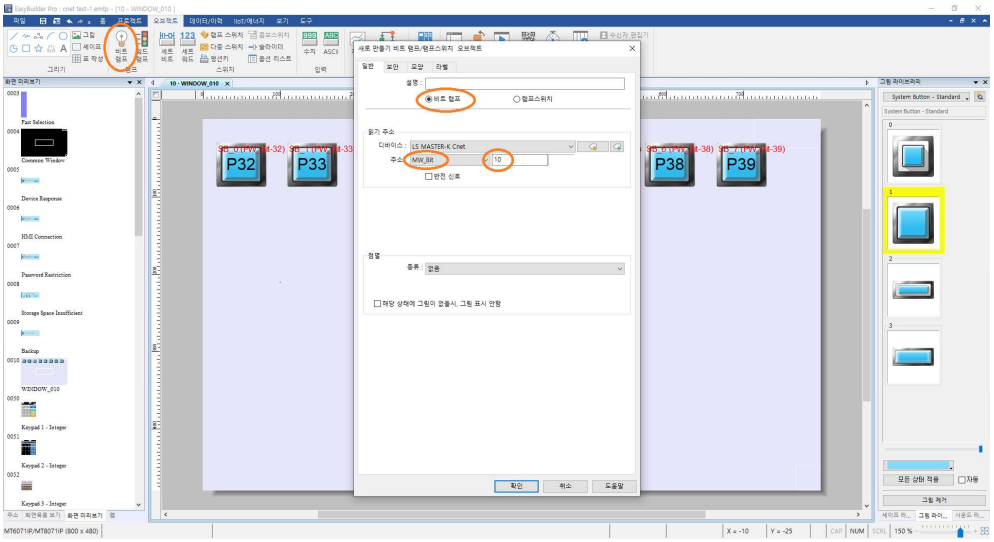
VI. EasyBuilder Pro 비트램프 생성

EasyBuilder Pro 작화 화면의 상단 메뉴의 오브젝트 -> 비트램프를 선택하고 읽기주소 파라미터를 변경합니다. 본 매뉴얼에서는 MPA-16A16RX 제품의 출력 릴레이 P32~P39가 ON되면 각각 M10~M17이 ON되도록 MP STUDIO 프로그램을 하고 M0~M17의 상태를 표시하는 램프를 생성하였습니다.

- 디바이스 : 위에서 생성한 디바이스를 선택합니다. RS485 통신방식을 사용하여 여러대의 PLC를 제어하기 위해서는 디바이스를 제어하고자 하는 PLC 이름을 선택해야 합니다.
- 주소 : 비트주소는 PW_Bit와 MW_Bit 만을 지원합니다. M10은 MW_Bit를 선택하고 10을 기입합니다.

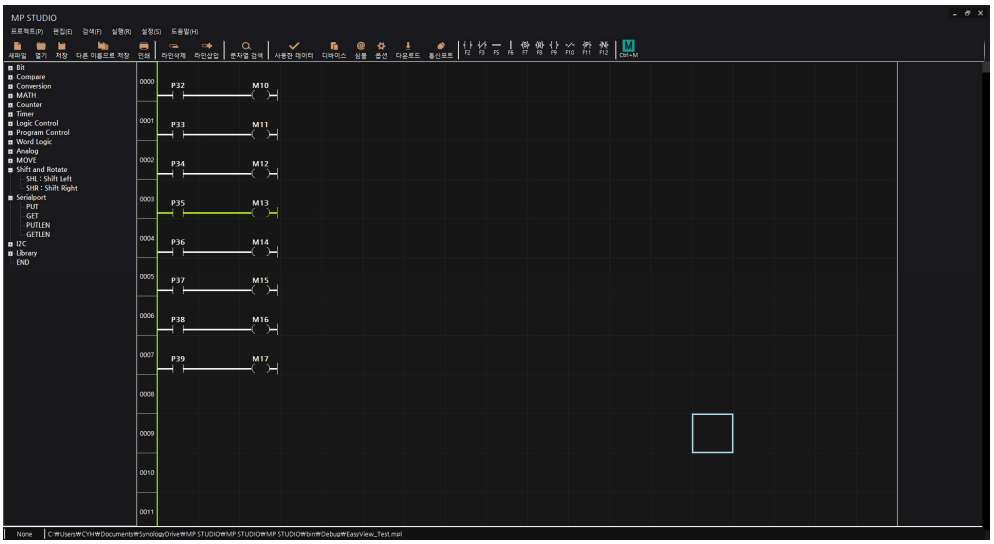


<그림 7 - MP STUDIO 프로그램 작성 및 EasyBuilder Pro 비트램프 생성 >

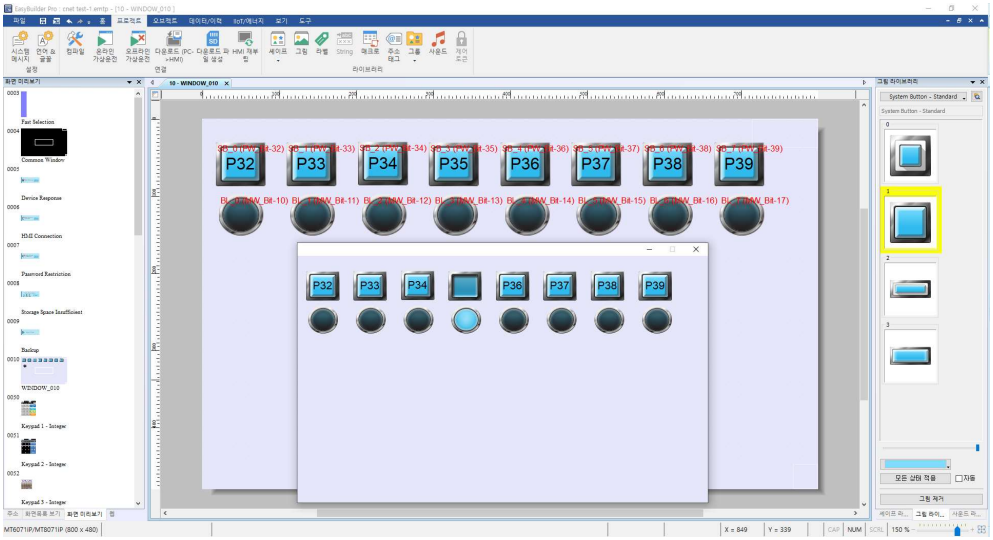


<그림 7 - MP STUDIO 프로그램 작성 및 EasyBuilder Pro 비트램프 생성 >

- 총 8개(MW_Bit 10 ~ 17)의 램프를 버튼 하단에 생성하고 P35 버튼을 터치하면 아래와 같이 동작됩니다.



<그림 8 - MP STUDIO 프로그램 작성 및 EasyBuilder Pro 비트램프 동작 >

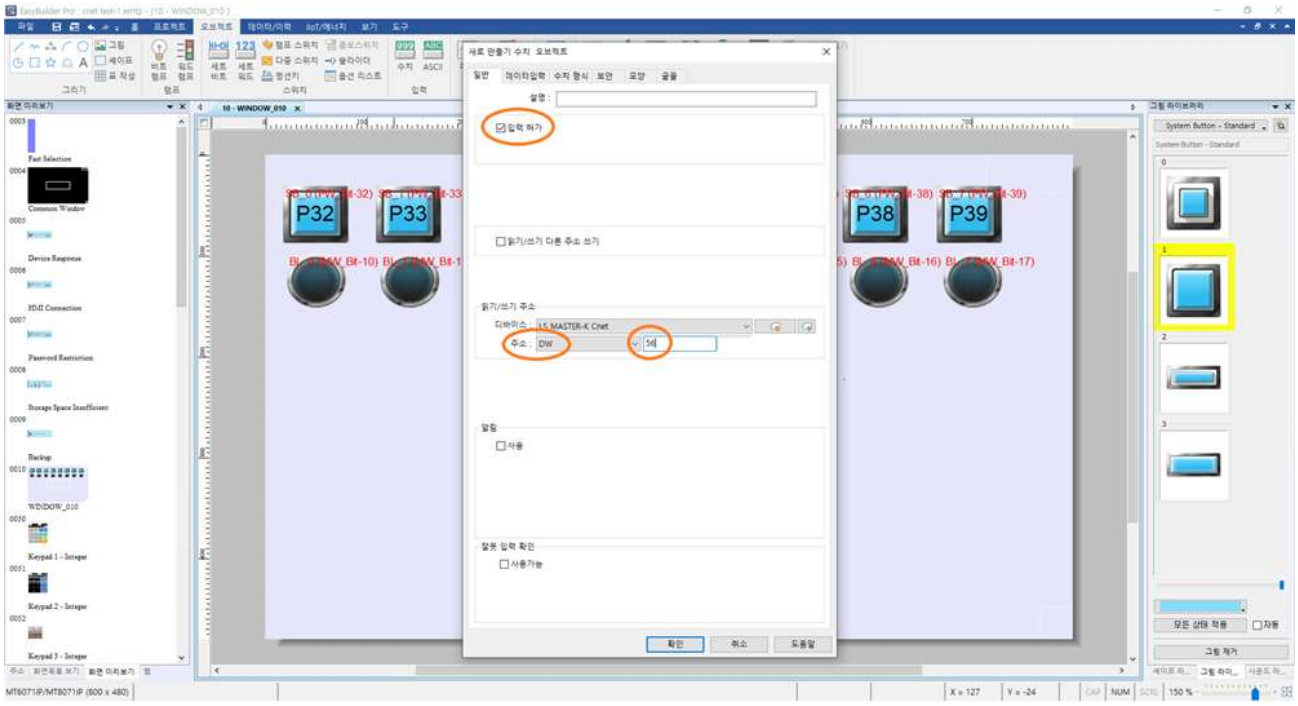


<그림 8 - MP STUDIO 프로그램 작성 및 EasyBuilder Pro 비트램프 동작 >

Ⅶ. EasyBuilder Pro 수치 생성

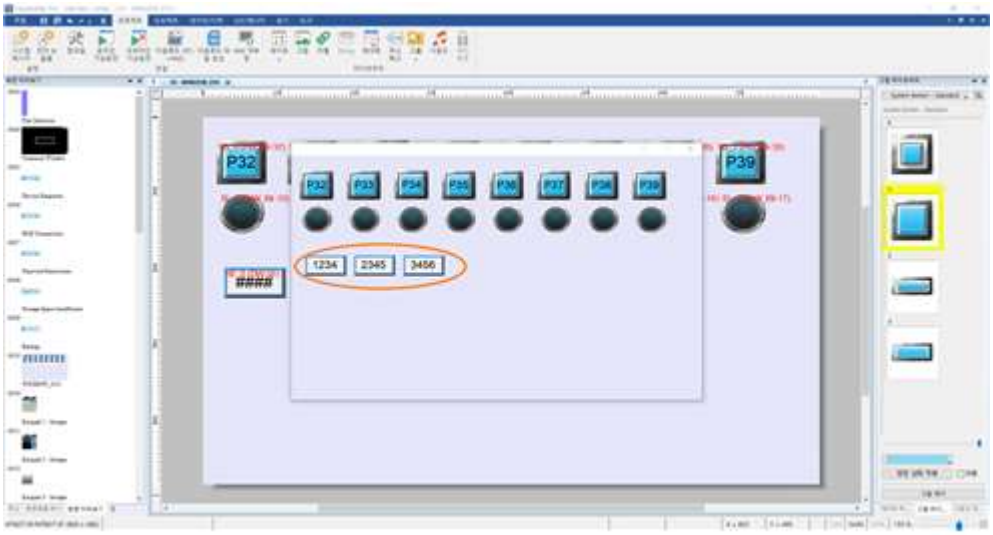
EasyBuilder Pro 작화 화면의 상단 메뉴의 오브젝트 -> 수치를 선택하고 읽기주소 파라미터를 변경합니다. 본 매뉴얼에서는 D56에 1234를 D67에 2345를 D123에 3456을 MP STUDIO로 프로그램을 작성&다운로드 하고 EasyBuilder Pro에 수치 오브젝트 3개를 만들었습니다.

- 입력허가 : 디스플레이만을 할 경우 체크박스를 해제하고 입력까지 할 경우 체크를 합니다.
- 주소 : D56의 경우 DW를 선택하고 주소값에 56을 입력합니다.



<그림 9 - EasyBuilder Pro 세트비트(스위치) 생성 >

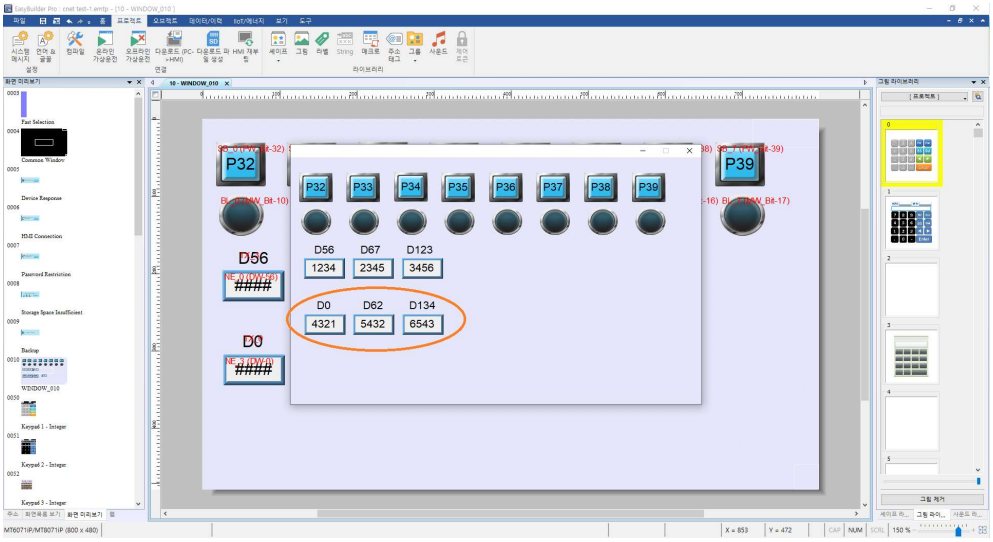
- 아래와 같이 MP STUDIO에서 D56,D67,D123에 각각 1234, 2345, 3456을 계속 저장하고 있어 EasyView에서도 그 수치가 모니터링 되고 있습니다.



<그림 10 - EasyBuilder Pro 수치 모니터링 확인 >

입력허가를 그대로 유지하고 D0, D62,D134에 각각 4321, 5432, 6543을 EasyView에서

터치하여 입력하고 MP STUDIO에서 모니터링하여 값이 변경되는지 아래와 같이 확인하였습니다.

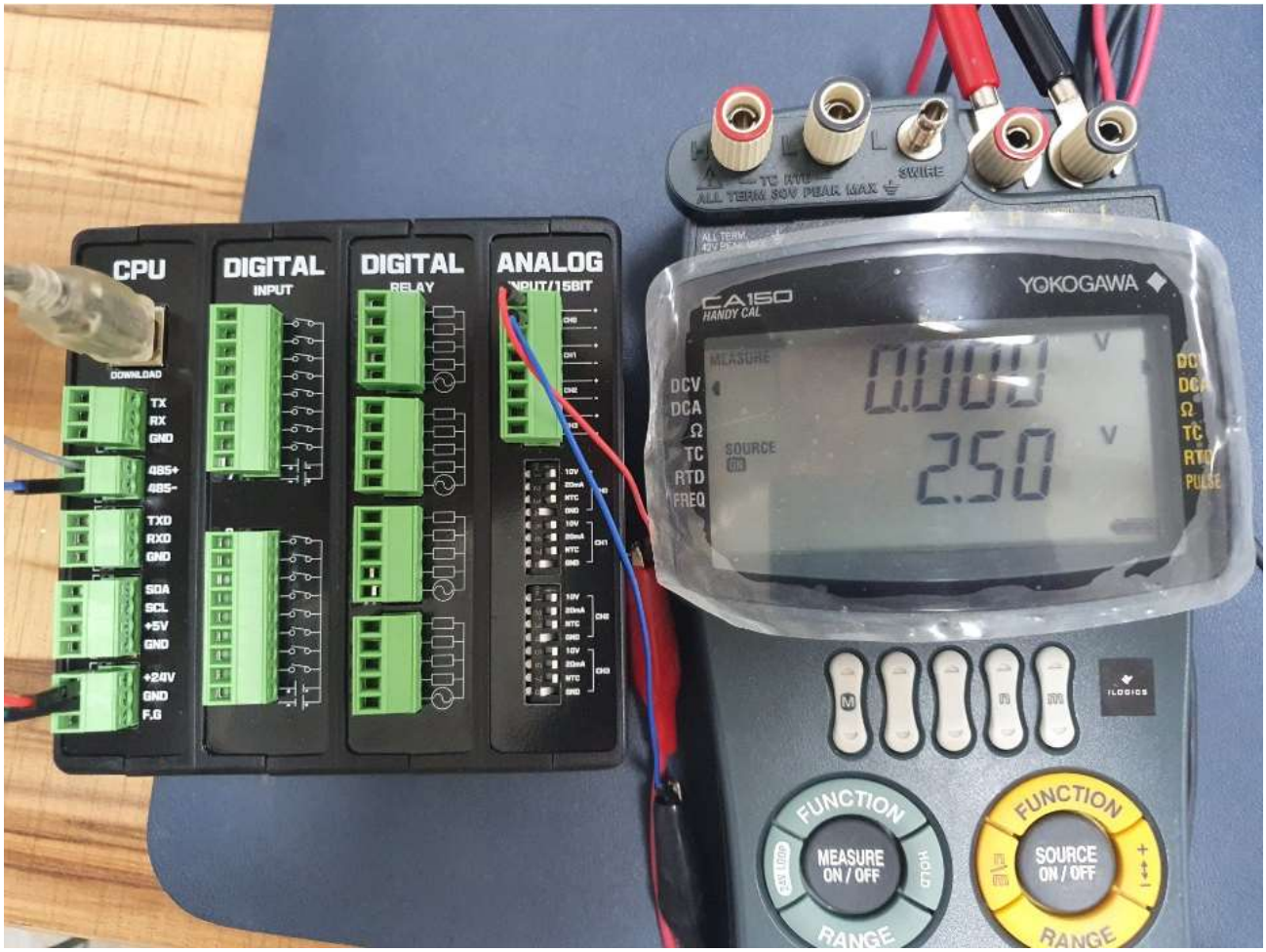


<그림 11 - EasyBuilder Pro 수치입력 및 MP STUDIO 수치 입력 모니터링 >

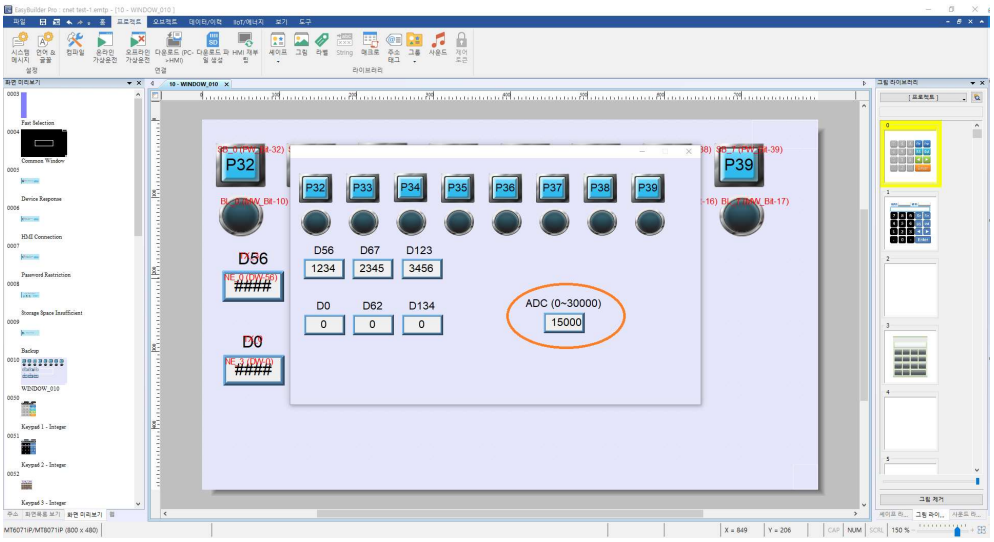
Ⅷ. 아날로그 입력값 표시

MP STUDIO에서 ADC 펑션블럭을 이용하여 프로그램하고 다운로드 합니다. 본 매뉴얼에서는 CHANNEL: 0, MAX: 30000, MIN: 0, OUT: D80을 하였습니다. EasyBuilder Pro에서는 작화 화면의 상단 메뉴의 오브젝트 -> 수치를 선택하고 읽기주소 파라미터를 DW 80으로 입력후 오브젝트를 생성합니다. (PLC: MPA-16A16RX, 아날로그입력 0번채널은 0~5V 범위로 설정)

- 수치 오브젝트 속성에서 수치형식->표시형식 칸에 있는 소수점이상 값이 기본으로 4로 설정되어 있으므로 5자리까지 가능하도록 5로 변경해야 합니다.)



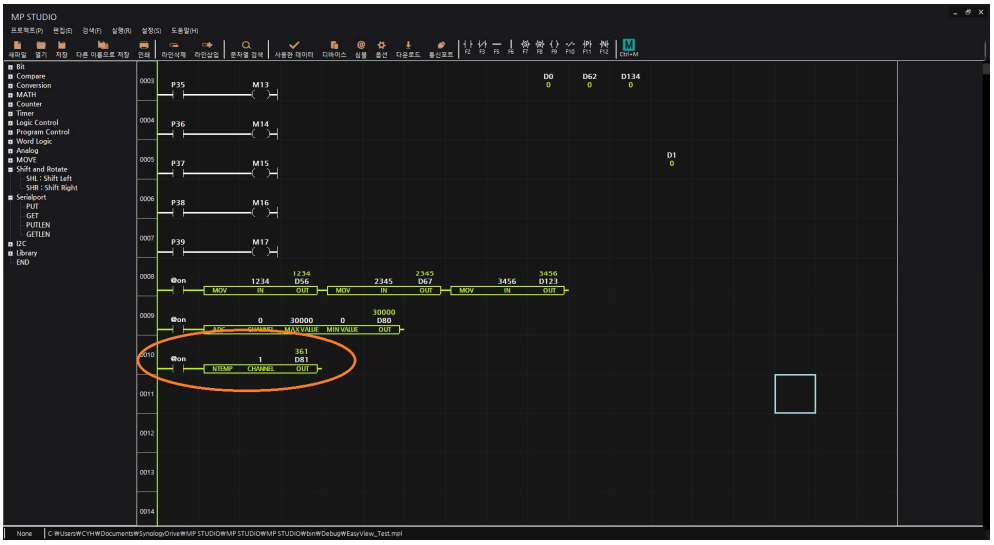
<그림 12 - 아날로그 입력모듈 CH0에 캘리브레이터로 2.5V 입력 >



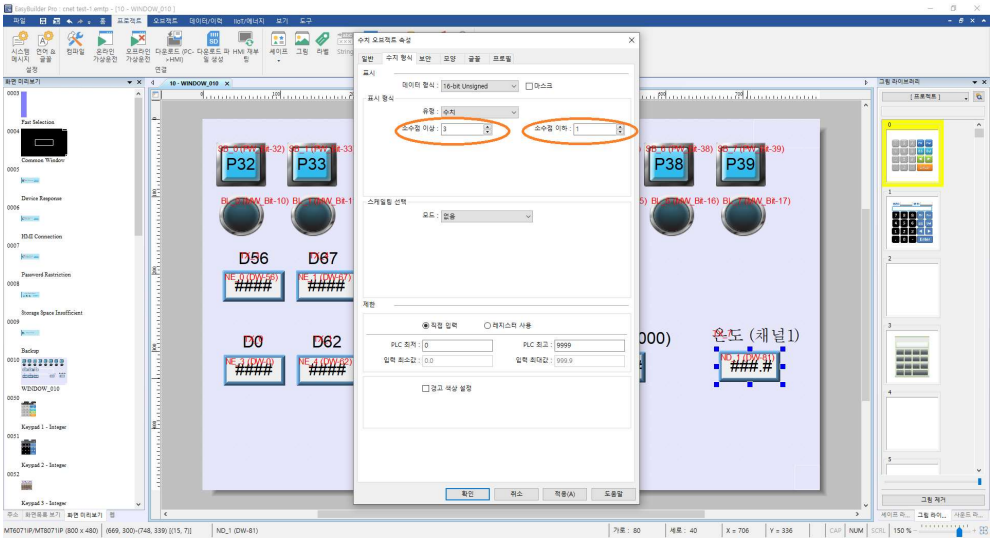
<그림 11 - EasyBuilder Pro 수치입력 및 MP STUDIO 수치 입력 모니터링 >

IX. 온도 표시

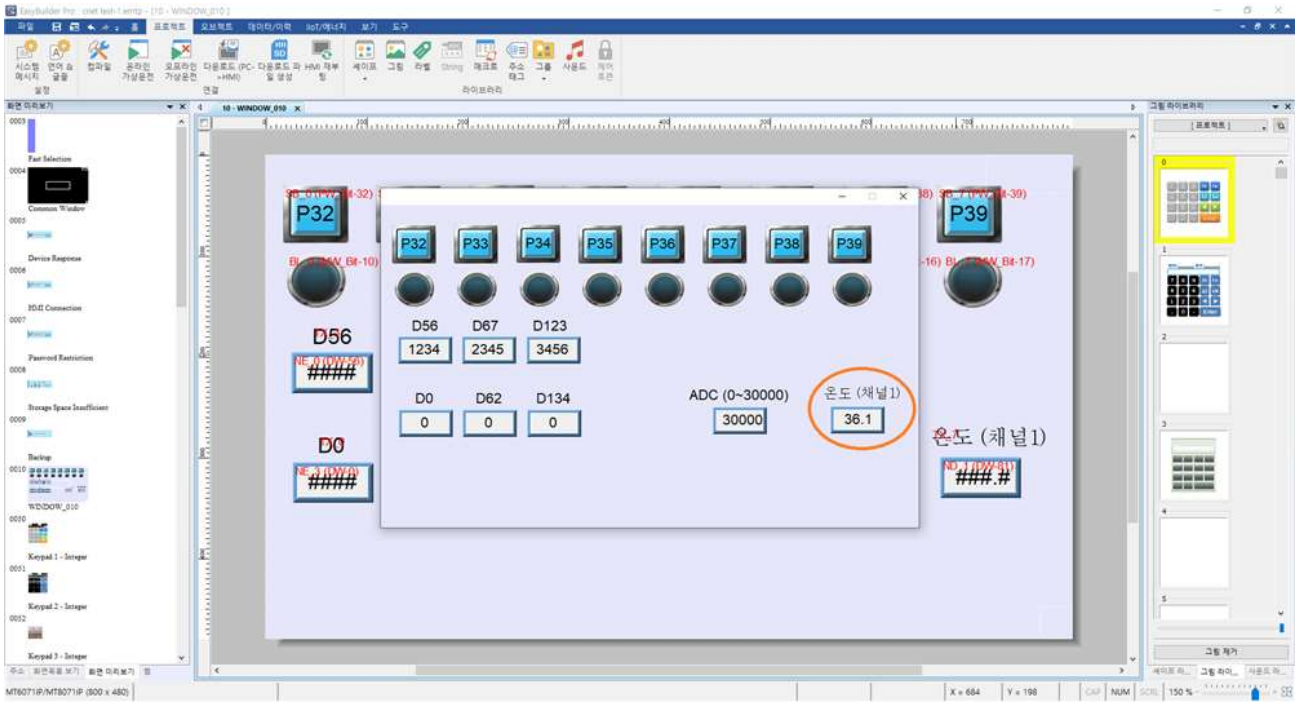
MP STUDIO에서 NTEMP 펑션블럭을 이용하여 프로그램하고 다운로드 합니다. 본 매뉴얼에서는 CHANNEL: 1, OUT: D81으로 프로그램 하였습니다. EasyBuilder Pro에서는 작화 화면의 상단 메뉴의 오브젝트 -> 수치를 선택하고 읽기주소 파라미터를 DW 81으로 입력하고 수치형식 탭에서 소수점이상:3, 소수점이하1로 설정하여 오브젝트를 생성합니다. (PLC: MPA-16A16RX, 아날로그입력 1번채널은 NTC 온도센서 입력으로 설정)



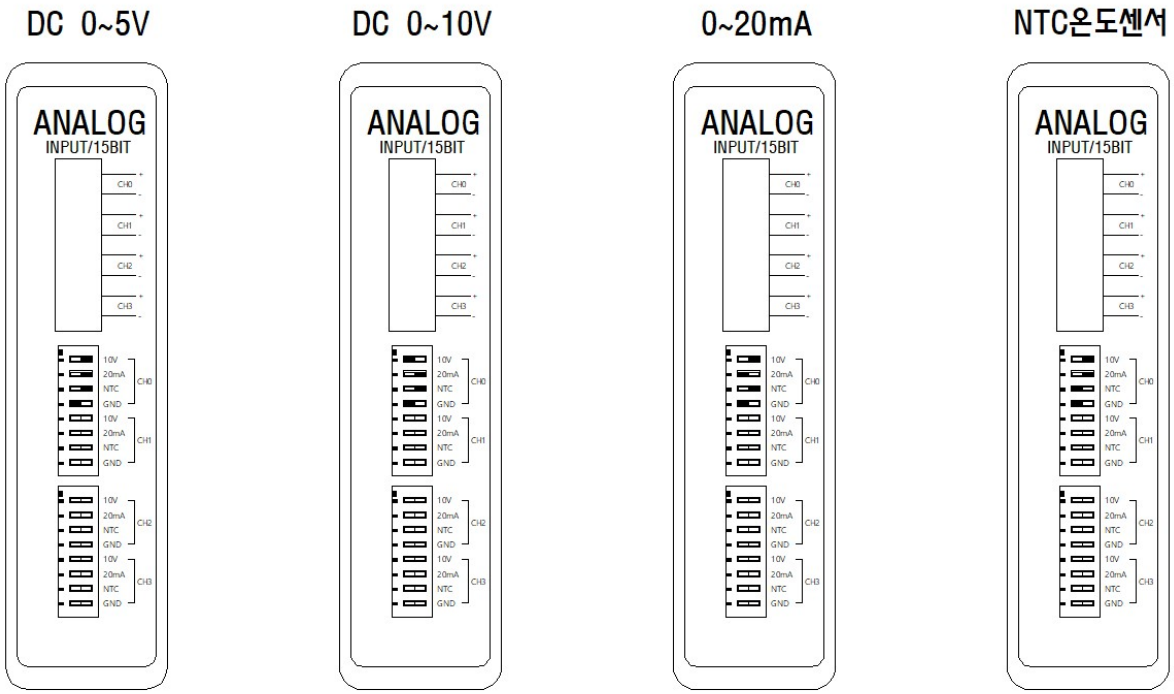
<그림 14 - MP STUDIO 프로그램에서 NTEMP펑션블럭으로 온도값 읽기 >



<그림 14 - EasyBuilder Pro 수치 오브젝트 생성 >



<그림 15 - EasyBuilder Pro에서 온도 모니터링 >



*딥스위치 설정은 예시로 CH0만 표시함
*GND는 연결한 아날로그출력 제품의 전원이 PLC와 공통이면 OFF 권장 (ON설정도 동작에 무관함)
*GND는 연결한 아날로그출력 제품의 전원이 PLC와 별도로 사용할 경우 ON (OFF설정은 안됨)

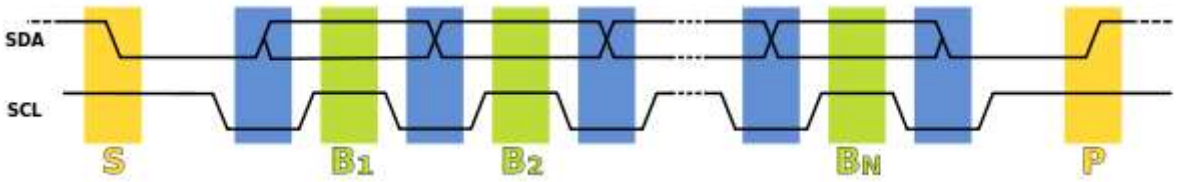
<그림 16 - NTC 온도센서를 CH1에 연결하고 CH1의 딥스위치를 NTC로 설정 >

9. I2C

I2C는 필립스(Philips)사에서 제안한 근거리용 표준 직렬통신 방식입니다. 2가닥의 통신선으로 하나의 마스터 장비가 여러개의 슬레이브와 반이중으로 통신을 할 수 있는 특징이 있습니다. 2가닥의 통신선은 아래와 같습니다.

- SDA : Data Line
- SCL : Clock Line

I2C는 마스터가 송신하는 SCL(CLock)을 기준으로 슬레이브는 동기식으로 동작합니다. 마스터가 SCL을 HIGH를 출력하고 SDA를 LOW로 출력하면, 슬레이브는 0으로 인식하고 SDA를 HIGH로 출력하면, 슬레이브는 1로 인식합니다.



S : Start, B1..Bn : Data, P : Stop

MPS 제품군의 I2C 특징은 아래와 같습니다.

- 마스터모드만 제공합니다. 슬레이브모드가 필요할 경우, MPINO 제품군을 사용해 주시기 바랍니다.
- 400kbps 속도로 통신합니다.

9.1. ISTA

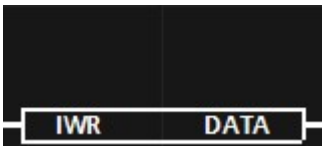
START 컨디션을 만듭니다.

<그림7-1 ISTA 펑션블럭>

Parameter	Data Type	
ISTA	-	펑션블럭명으로서, I2C START의 약자입니다.
ADDRESS	unsigned byte	슬레이브 제품에서 I2C통신을 위해 지정한 슬레이브 주소를 입력합니다. 슬레이브 주소는 7비트로서, 1 ~ 127 사용이 가능합니다. 0은 슬레이브 전체를 호출하기 위해 사용됩니다.
READ	bit	I2C는 송신모드와 수신모드가 있습니다. READ에 FALSE 또는 @OFF를 입력하면 송신모드가 되고, TRUE 또는 @ON을 입력하면 수신모드가 됩니다.

9.2. IWR

마스터인 MPS 제품에서 슬레이브 제품으로 1개의 바이트를 송신합니다.



<그림7-2 IWR 펑션블럭>

Parameter	Data Type	
IWR	-	펑션블럭명으로서, I2C WRITE의 약자입니다.
DATA	unsigned byte	슬레이브 제품으로 송신할 바이트를 입력합니다.

9.3. IRD

마스터인 MPS 제품에서 슬레이브 제품으로 1개의 바이트를 수신합니다.

<그림7-3 IRD 펑션블럭>

Parameter	Data Type	
IRD	-	펑션블럭명으로서, I2C READ의 약자입니다.
OUT	unsigned byte	슬레이브 제품으로 수신한 데이터를 저장할 메모리를 입력합니다.
ACK	bit	슬레이브 제품으로 수신을 받은 후에, 슬레이브 제품으로 ACK 신호를 송신할지를 결정합니다. FALSE 또는 @OFF 이면, ACK를 송신하지 않고 TRUE 또는 @ON이면, ACK를 송신합니다.

9.4. ISTP

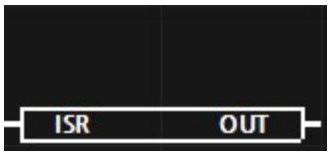
STOP 컨디션을 만듭니다.

〈그림7-4 ISTP 펑션블럭〉

Parameter	Data Type	
ISTP	-	펑션블럭명으로서, I2C STOP의 약자입니다.

9.5. ISR

마스터인 MPS 제품에서 STOP 컨디션을 만듭니다.



<그림7-5 ISR 펑션블럭>

Parameter	Data Type	
ISR	-	펑션블럭명으로서, I2C STATUS REGISTER의 약자입니다.
OUT	unsigned byte	I2C 상태값을 저장할 메모리를 입력합니다.

9.6. I2C Example

정밀하고 제품의 전원이 차단되어도 유지되는 시간을 사용하기 위해 DS3231 RTC와의 통신을 하는 예제입니다. DS3231는 DS3231_WR과 DS3231_RD의 라이브러리 펑션블럭을 제공하고 있지만, DS3231 I.C가 I2C 통신방식이기 때문에, 여기서 예를 들어 설명하도록 하겠습니다.

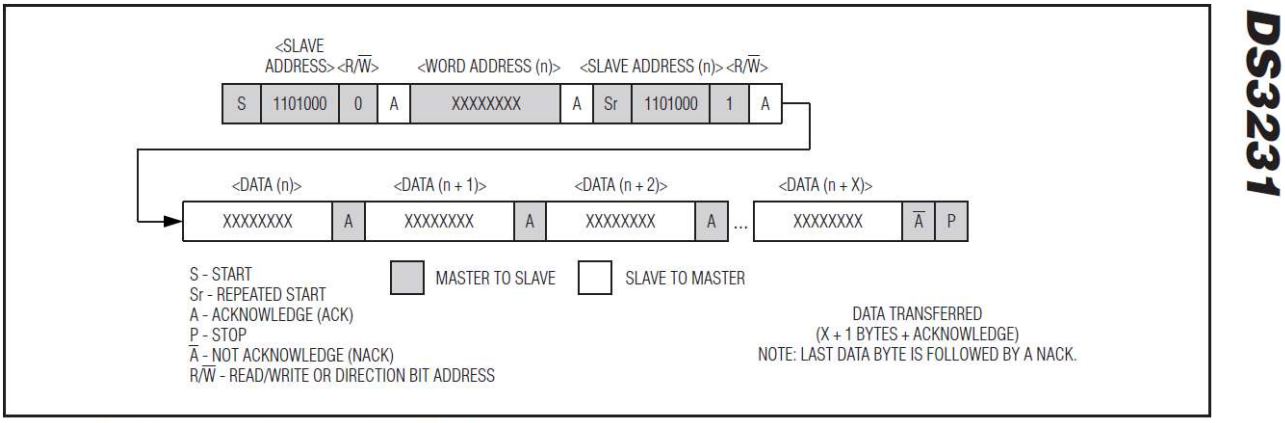
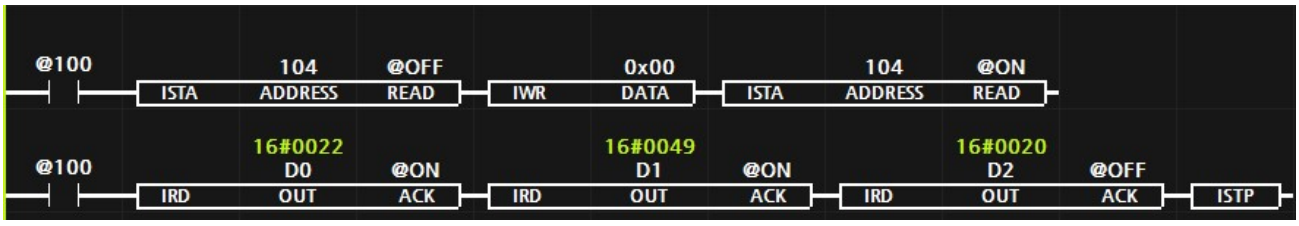


Figure 5. Data Write/Read (Write Pointer, Then Read)—Slave Receive and Transmit

<그림7-6-1 DS3231의 시간값 송신 프로토콜>



<그림7-6-2 MP STUDIO에서 초,분,시간을 읽기 위한 프로그램>

① ISTA 펑션블럭 실행

STEP	Data	
START	S	마스터에서 START 컨디션을 만듭니다.
SLAVE ADDRESS	1101000	ISTA 펑션블럭의 ADDRESS에 104를 입력합니다. DS3231 I.C는 슬레이브 주소가 104(1101000)로 정해져 있습니다.
<R/$\overline{W}$>	0	ISTA 펑션블럭의 READ에 @OFF를 입력하여, 마스터를 송신모드를 만듭니다.
A	1	DS3231 I.C에서 ACK신호가 옵니다. ISTA 펑션블럭에서 알아서 처리합니다.

② IWR 펑션블럭 실행

STEP	Data	
WORD ADDRESS	0x00	IWR 펑션블럭의 DATA에 0을 입력합니다. DS3231에서 읽어올 번지수를 입력하는 것으로서 0x00은 초, 0x01은 분, 0x02는 시간입니다.
A	1	DS3231 I.C에서 ACK신호가 옵니다. ISTA 펑션블럭에서 알아서 처리합니다.

③ ISTA 펑션블럭 실행

STEP	Data	
SR	-	마스터에서 REPEAT START 컨디션을 만듭니다.
SLAVE ADDRESS	1101000	ISTA 펑션블럭의 ADDRESS에 104를 입력합니다. DS3231 I.C는 슬레이브 주소가 104(1101000)로 정해져 있습니다.
<R/$\overline{W}$>	1	ISTA 펑션블럭의 READ에 @ON을 입력하여, 마스터를 수신모드를 만듭니다.
A	1	DS3231 I.C에서 ACK신호가 옵니다. ISTA 펑션블럭에서 알아서 처리합니다.

④ IRD 펑션블럭 실행

STEP	Data	
DATA	초	DS3231 I.C로부터 초값이 수신되며, 수신된 초값을 IRD 펑션블럭의 OUT 파라메타에 기입된 메모리에 저장됩니다. 예제에서는 D0에 초가 저장됩니다.
ACK	1	마스터에서 데이터를 수신 받았으므로, 슬레이브에게 ACK를 송신해야 합니다. ACK를 송신하면 슬레이브인 DS3231는 0x00(초) 레지스터리 번지에서 +1을 더한 0x01(분) 레지스터리 값을 마스터에게 송신합니다.

⑤ IRD 펑션블럭 실행

STEP	Data	
DATA	초	DS3231 I.C로부터 분값이 수신되며, 수신된 분값을 IRD 펑션블럭의 OUT 파라메타에 기입된 메모리에 저장됩니다. 예제에서는 D1에 분이 저장됩니다.
ACK	1	마스터에서 데이터를 수신 받았으므로, 슬레이브에게 ACK를 송신해야 합니다. ACK를 송신하면 슬레이브인 DS3231는 0x01(분) 레지스터리 번지에서 +1을 더한 0x02(시간) 레지스터리 값을 마스터에게 송신합니다.

⑥ IRD 펑션블럭 실행

STEP	Data	
DATA	초	DS3231 I.C로부터 시간값이 수신되며, 수신된 시간값을 IRD 펑션블럭의 OUT 파라메타에 기입된 메모리에 저장됩니다. 예제에서는 D2에 시간이 저장됩니다.
ACK	1	마스터에서 데이터를 수신 받았으므로, 슬레이브에게 ACK를 송신해야 합니다. ACK를 송신하면 슬레이브인 DS3231는 0x02(시간) 레지스터리 번지에서 +1을 더한 0x03(요일) 레지스터리 값을 마스터에게 송신합니다. 예제에서는 다음 데이터를 수신하지 않고자 ACK를 0... 즉, @OFF를 입력하여 데이터 수신을 마칩니다. ※ 주의.. ACK를 1... 즉, @ON을 입력하고 다음 스텝인 ISTP를 실행하면, 슬레이브와 마스터의 SDA가 충돌이 일어나서 마스터인 MPS는 슬레이브 모드로 자동진입이 됩니다. 때문에 이후, I2C를 사용하지 못합니다. (프로그램을 수정하고 전원을 재투입하면 정상동작합니다.)

⑦ ISTP 펑션블럭 실행

STEP	Data	
ISTP		마스터에서 STOP 컨디션을 만듭니다.

DS3231는 데이터의 형식이 16진수인 HEX값으로 동작됩니다. HEX를 10진수인 Decimal값으로 변경할려면, H2D 펑션블럭을 Decimal을 Hex값으로 변경할려면 D2H 펑션블럭을 이용해주시기 바랍니다.

10. Library

10.1. DS3231 (Real Time Clock)

MP STUDIO에서 1초마다 초를 증가시키고 60초가 되면 분이 증가시키는 방식으로 시계를 만들 수 있습니다. 하지만, MPS 제품의 전원이 차단되면 계산된 시간을 유지하지 못하고 전원이 차단되어 있는 동안은 시간이 증가하지 않는 문제가 있습니다. 또한, MPS 제품의 타이머는 MPS 제품에 삽입되어 있는 오실레이터(14.7456Mhz)의 클럭에 의해 동작되는 것입니다. 하지만, 컨트롤러에 들어가는 클럭의 주파수는 다소 오차범위가 많습니다. 때문에, MPS에서 시간을 만들면 오래 지날수록 시간에 오차가 발생합니다.

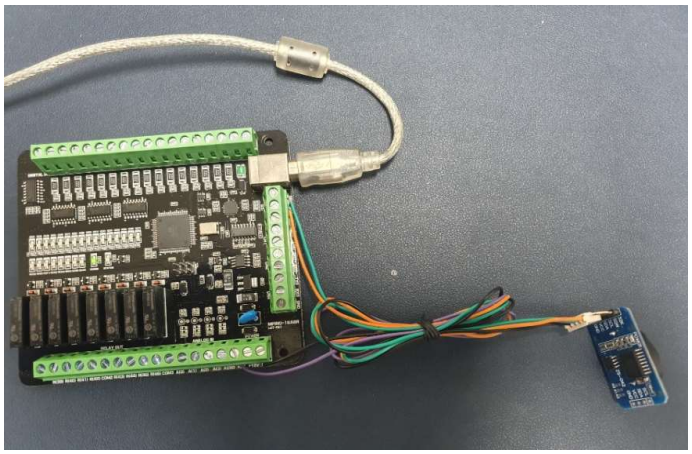
이러한 문제를 DS3231를 사용하는 방법으로 해결할 수 있습니다. 이렇게 정밀한 시계 기능이 있는 것을 RTC(Real Time Clock)이라고 말하며, 여러 RTC중에서 DS3231는 매우 정밀한 RTC입니다.

DS3231의 오차율은 아래와 같습니다.

♦ Accuracy ±2ppm from 0°C to +40°C ♦ Accuracy ±3.5ppm from -40°C to +85°C

상온에서 분당 2개의 펄스의 오차율로서, 1년으로 계산하면 약32.4초의 오차가 발생할 수 있습니다.

DS3231는 I.C의 이름입니다. MPS는 DS3231를 기본으로 장착하지 않고, 모듈로서 연결할 수 있도록 지원하고 있습니다.



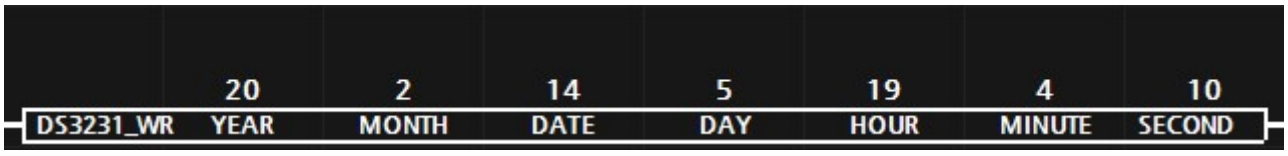
<그림8-1 MPS-16A8R와 DS3231 모듈 연결>

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99

<그림8-1-0 DS3231의 시간 레지스터리>

10.1.1. DS3231_WR (DS3231 Write)

DS3232 I.C에 년,월,일,요일,시,분,초를 설정합니다. 모든 파라메타는 10진수를 사용합니다.

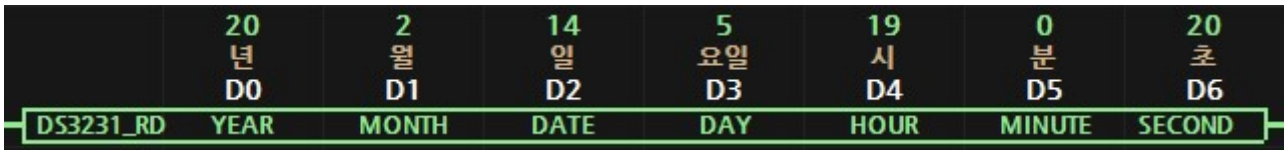


<그림8-1-1 DS3231_WR 펑션블럭>

Parameter	Data Type	
YEAR (년)	unsigned byte	0~99(년)을 입력할 수 있습니다. 2000년을 뺄셈한 값을 입력하시면 됩니다. 20은 2020을 의미합니다.
MONTH (월)	unsigned byte	1 ~ 12(월)을 입력할 수 있습니다.
DATE (일)	unsigned byte	1 ~ 31(일)을 입력할 수 있습니다.
DAY (요일)	unsigned byte	1 ~ 7(요일)을 입력할 수 있습니다.
HOUR (시)	unsigned byte	0 ~ 23(시)를 입력할 수 있습니다.
MINUTE (분)	unsigned byte	0 ~ 59(분)을 입력할 수 있습니다.
SECOND (초)	unsigned byte	0 ~ 59(초)를 입력할 수 있습니다.

10.1.2. DS3231_RD (DS3231 Read)

DS3232 I.C에 년,월,일,요일,시,분,초를 읽어옵니다. 모든 파라메타는 10진수를 사용합니다.



<그림8-1-2 DS3231_RD 펑션블럭>

Parameter	Data Type	
YEAR (년)	unsigned byte	0~99(년)이 파라메타에 입력한 메모리에 저장됩니다. 저장된 메모리 값에서 2000년을 덧셈하여 사용하실 수 있습니다.
MONTH (월)	unsigned byte	1 ~ 12(월)이 파라메타에 입력한 메모리에 저장됩니다.
DATE (일)	unsigned byte	1 ~ 31(일)이 파라메타에 입력한 메모리에 저장됩니다.
DAY (요일)	unsigned byte	1 ~ 7(요일)이 파라메타에 입력한 메모리에 저장됩니다.
HOUR (시)	unsigned byte	0 ~ 23(시)이 파라메타에 입력한 메모리에 저장됩니다.
MINUTE (분)	unsigned byte	0 ~ 59(분)이 파라메타에 입력한 메모리에 저장됩니다.
SECOND (초)	unsigned byte	0 ~ 59(초)이 파라메타에 입력한 메모리에 저장됩니다.

10.2. CLCD (Character LCD / PCF8574)

캐릭터 LCD는 DB0~DB7을 직접제어하는 방식으로 사용할 수 있지만, 직접제어하기 위해서는 MCU의 GPIO를 많이 사용해야 하므로 I2C 통신을 이용하여 제어할 수 있도록 모듈로 만들어진 것이 있습니다.

본 라이브러리를 통하여 저렴하고 많이 사용하고 있는 PCF8574 모듈을 캐릭터 LCD에 연결하여 I2C 통신으로 캐릭터 LCD를 제어할 수 있습니다.

<그림8-2-1 PCF8574 모듈 및 CLCD와의 연결>

✋ CLCD에 표현되는 문자의 선명도가 흐릴 경우, PCF8574에 있는 가변저항을 조절하여 선명도를 설정할 수 있습니다.

10.2.1. CLCD 주소

PCF8574 모듈의 기본주소는 0x27입니다. 2개 이상의 캐릭터 LCD를 사용할 경우, 같은 주소인 제품 2개를 사용할 수 없습니다. 한개의 캐릭터 LCD가 0x27이면, 다른 한개는 0x27 이외의 주소를 가져야 합니다. 이렇게 주소를 변경하기 위해서는 PCF8574 모듈의 PCB 보드에 있는 A0, A1, A2의 패드를 숏트시키는 것으로 변경할 수 있습니다.

PCF8574의 주소는 아래처럼 이루어져 있습니다.

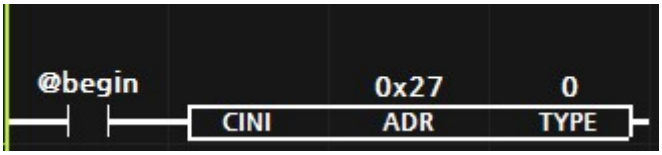
0	1	0	0	A2	A1	A0

<그림8-2-2 PCF8574 모듈의 어드레스 주소>

여기서 A0, A1, A2는 PCB보드에서 풀업(HIGH)이 되어 있어 1로 인식이 되어 있습니다.
따라서, A0의 패드 2개를 숏트 시키면, 0x27에서 1을 뺀 0x26이 주소가 됩니다.
A1의 패드 2개를 숏트 시키면, 0x27에서 2를 뺀 0x25가 주소가 됩니다.
A2의 패드 2개를 숏트시키면, 0x27에서 4를 뺀 0x23이 주소가 됩니다.
A0 패드 2개를 숏트 시키고 A1 패드 2개를 숏트 시키면, 0x27에서 1을 빼고 2를 뺀 0x24가 주소가 됩니다.

10.2.2. CINI (CLCD Initialize)

CLCD는 전원이 투입되면, 항상 초기화를 해주어야 합니다. 초기화를 하지 않을 경우, CLCD는 작동하지 않습니다.



<그림8-2-2-1 CLCD 펑션블럭>

@begin 비트접점은 MPS의 전원투입시 최초 1회만 ON되는 특수 비트 메모리입니다. CLCD의 초기화는 전원투입시 1회만 실행되면 되므로 @begin을 사용하시기 바랍니다.

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	초기화할 CLCD의 주소값을 입력합니다.
TYPE	BYTE, WORD, DWORD	초기화할 CLCD의 TYPE을 입력합니다.

여러 가지의 CLCD 형식이 있으므로, CLCD 형식에 맞는 TYPE을 기입해야 합니다.
현재는 TYPE을 구분하지 않습니다. 숫자 "0"을 기입해 주시기 바랍니다.

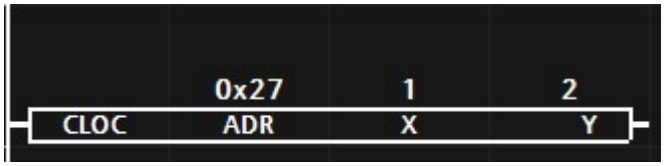
현재 사용할 수 있는 CLCD의 형태는 아래와 같습니다. (문자 x 행)

① 16문자 x 1행
② 16문자 x 2행
③ 20문자 x 2행
④ 20문자 x 4행
⑤ 24문자 x 2행
⑥ 40문자 2행

16 x 4행의 CLCD는 사용할 수 없습니다. 사용하고자 할 경우, 저희 아이로직스로 연락해 주시기 바랍니다.

10.2.3. CLOC (CLCD Location)

CLCD에 글씨를 쓰기 전에 위치를 잡아주기 위해 사용합니다. CLOC의 X,Y좌표로 커서가 이동합니다.



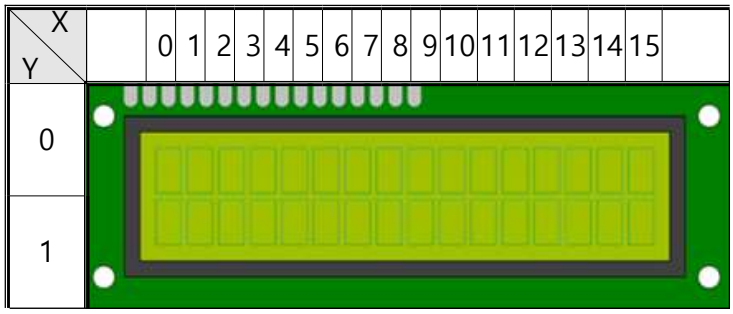
<그림8-2-3-1 CLOC 펑션블럭>

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	커서의 좌표를 이동할 CLCD의 주소값 입력
X	BYTE, WORD, DWORD	X좌표 입력
Y	BYTE, WORD, DWORD	Y좌표 입력

X, Y좌표는 0부터 시작하며, 캐릭터 LCD의 종류에 따라 좌표의 최대값은 변경됩니다.

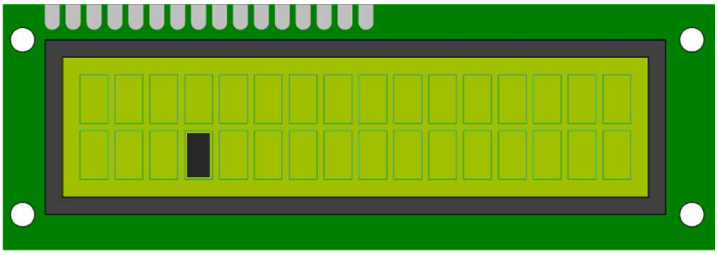
- 20 x 4 : X는 19, Y는 3이 최대값입니다.
- 16 x 2 : X는 15, Y는 1이 최대값입니다.

X, Y좌표는 아래와 같이 X(가로), Y(세로)의 좌표를 의미합니다.



<그림8-2-3-2 CLCD 좌표>

X가 3이고, Y가 1인 좌표는 아래와 같습니다.



<그림8-2-3-3 CLCD X:3, Y:1 좌표>

10.2.4. CSTR (CLCD String)

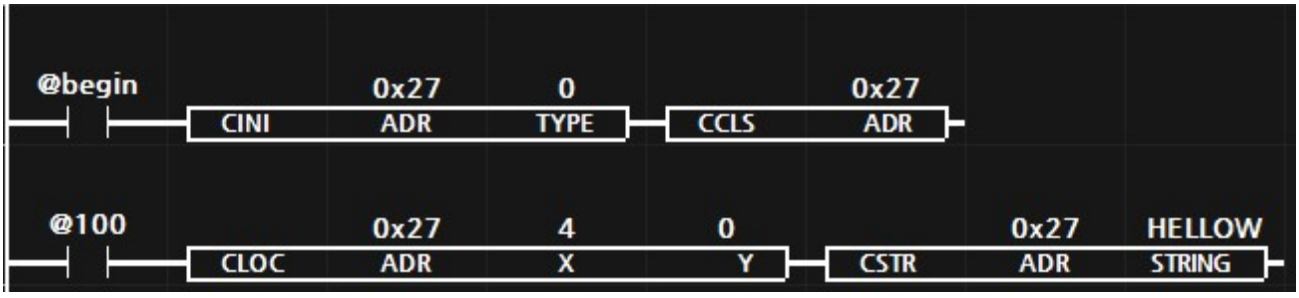
CLCD에 문자열을 표시합니다.
CSTR로 표현할 수 있는 문자는 ASCII코드로서 영문, 숫자, 특수문자로 한정되어 있습니다.
자세한 문자열은 아래의 표에서 확인할 수 있습니다.

<그림8-2-4-1 CSTR로 표현할 수 있는 문자열>

<그림8-2-4-2 CSTR 펑션블럭>

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	문자열을 표시할 CLCD의 주소값 입력
STRING	BYTE, WORD, DWORD	문자열 입력

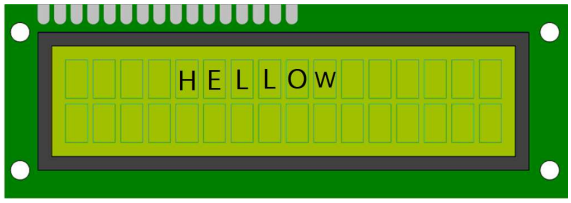
아래에 문자열을 표현하는 예를 보시기 바랍니다.



<그림8-2-4-3 CSTR 예제>

MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초(@100)마다 CLOC와 CSTR의 펑션블럭을 실행하여, X좌표:4와 Y좌표:0에 "HELLOW" 문자열을 표현합니다.

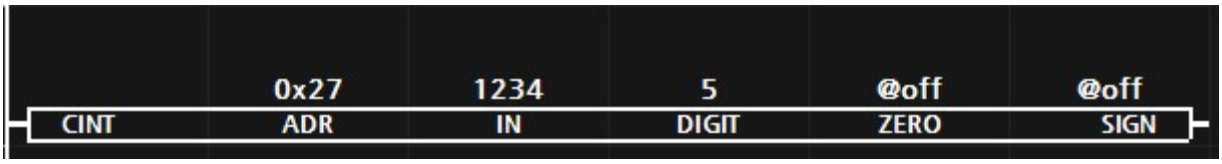
아래에 16x2 CLCD에 표현된 문자열을 나타내었습니다.



<그림8-2-4-4 CLCD에 "HELLOW"를 표시>

10.2.5. CINT (CLCD Integer)

CLCD에 숫자(10진수)를 표시합니다.



<그림8-2-5-1 CINT 펑션블럭>

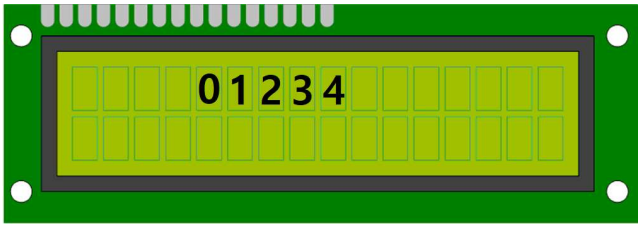
Parameter	Data Type	
ADR	BYTE, WORD, DWORD	숫자(10진수)를 표시할 CLCD의 주소값 입력
IN	BYTE, WORD, DWORD	표시할 숫자(10진수)를 입력
DIGIT	BYTE, WORD, DWORD	표시할 숫자(10진수)의 자릿수를 입력
ZERO	BIT	표시할 숫자(10진수)의 앞자리 0의 표시 유/무
SIGN	BIT	표시할 숫자(10진수)의 양(+), 음(-)의 부호표시 유/무

아래에 문자열을 표현하는 예1를 보시기 바랍니다. (ZERO : @on)

<그림8-2-5-2 CINT 예제1>

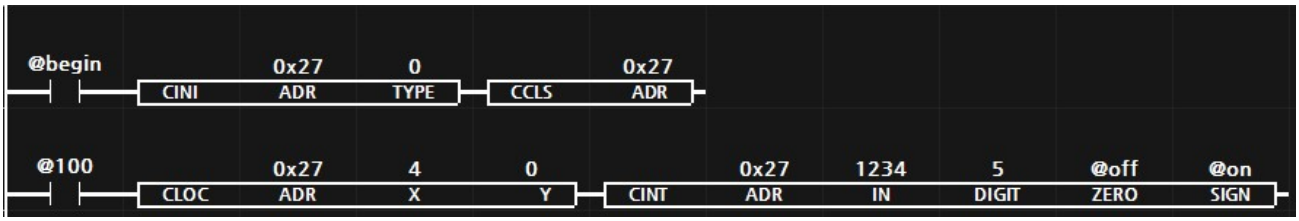
MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초(@100)마다 CLOC와 CINT의 펑션블럭을 실행하여, X좌표:4와 Y좌표:0에 1234를 5자리로 앞자리 0을 포함하여 표시합니다.

아래에 16x2 CLCD에 표현된 문자열을 나타내었습니다.



<그림8-2-5-3 CLCD에 01234를 표시>

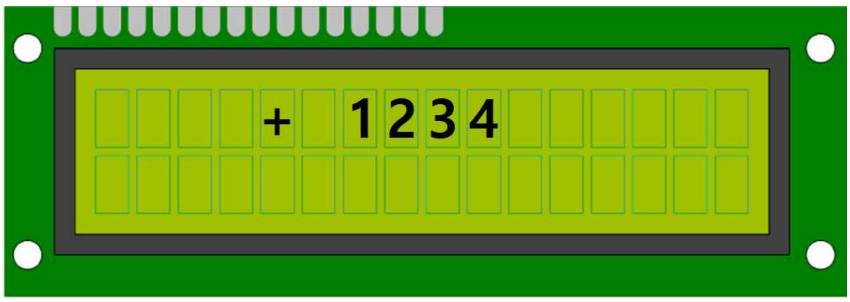
또한, 아래에 문자열을 표현하는 예2를 보시기 바랍니다. (ZERO : @off, SIGN : @on)



<그림8-2-5-4 CINT 예제2>

MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초(@100)마다 CLOC와 CINT의 펑션블럭을 실행하여, X좌표:4와 Y좌표:0에 1234를 5자리로 앞자리 0을 제외하고 양,음의 부호와 함께 표시합니다.

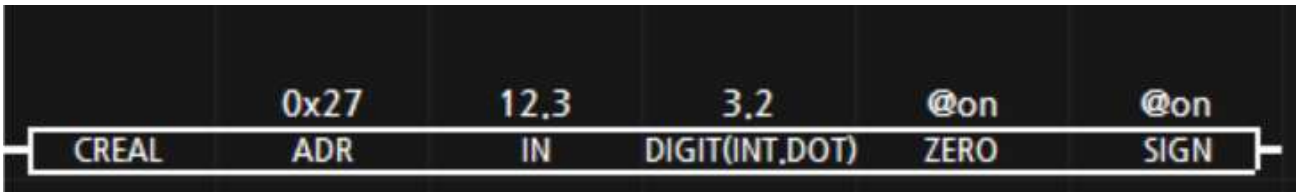
아래에 16x2 CLCD에 표현된 문자열을 나타내었습니다.



<그림8-2-5-5 CLCD에 + 1234를 표시>

10.2.6. CREAL (CLCD Flating Point)

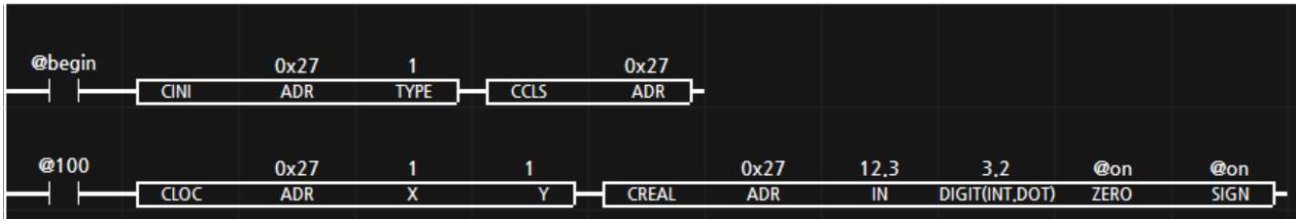
CLCD에 숫자(실수)를 표시합니다.



< CREAL 펑션블럭 >

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	숫자(10진수)를 표시할 CLCD의 주소값 입력
IN	REAL(Floating Point)	표시할 숫자(실수) 또는 실수(R) 메모리를 입력
DIGIT(INT.DOT)	숫자.숫자	표시할 정수(10진수) 자릿수 및 소수(10진수) 자릿수를 입력 Ex) 3.2 는 정수 3자리 및 실수 2자리를 표현
ZERO	BIT	표시할 숫자(실수)의 앞자리 0의 표시 유/무
SIGN	BIT	표시할 숫자(실수)의 양(+), 음(-)의 부호표시 유/무 부호표시는 정수 자릿수에 미포함

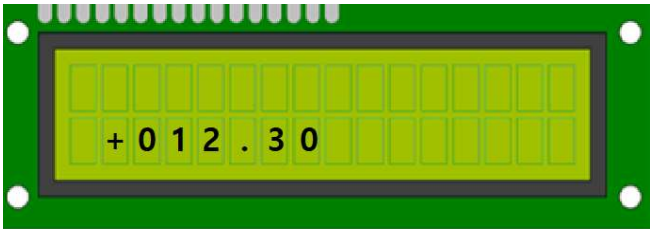
아래에 문자열을 표현하는 예1를 보시기 바랍니다.



< CREAL 예제 >

MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초(@100)마다 CLOC와 CREAL의 펑션블럭을 실행하여, X좌표:1와 Y좌표:1에 실수 12.3을 부호를 표시하고, 정수 3자리 소수점 2자리로 표시하고 정수의 앞자리는 0을 포함하여 표시합니다.

아래에 16x2 CLCD에 표현된 문자열을 나타내었습니다.



< CLCD에 12.3을 표시>

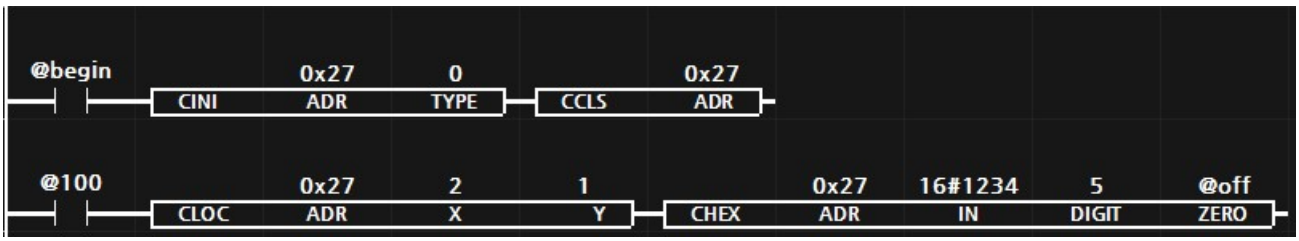
10.2.7. CHEX (CLCD Hexadecimal)

CLCD에 숫자(16진수)를 표시합니다.

<그림8-2-6-1 CHEX 펑션블럭>

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	숫자(10진수)를 표시할 CLCD의 주소값 입력
IN	BYTE, WORD, DWORD	표시할 숫자(16진수)를 입력
DIGIT	BYTE, WORD, DWORD	표시할 숫자(16진수)의 자릿수를 입력
ZERO	BIT	표시할 숫자(16진수)의 앞자리 0의 표시 유/무

아래에 문자열을 표현하는 예1를 보시기 바랍니다. (ZERO : @off)

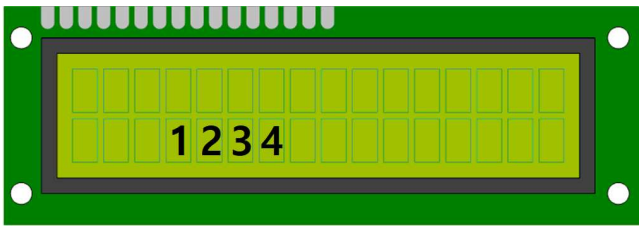


<그림8-2-6-2 CHEX 예제1>

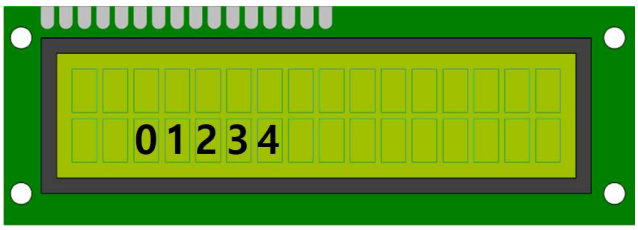
MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초(@100)마다 CLOC와 CHEX의 펑션블럭을 실행하여, X좌표:2와 Y좌표:1에 16진수 1234를 5자리로 앞자리 0을 제외하여 표시합니다.

또한, ZERO를 @On으로 하면 앞자리 0을 포함하여 표시합니다.

아래에 16x2 CLCD에 표현된 문자열을 나타내었습니다.



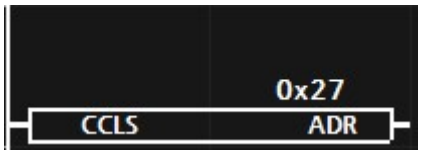
<그림8-2-6-3 ZERO:@off일 경우>



<그림8-2-6-4 ZERO:@on일 경우>

10.2.8. CCLS (CLCD Clear)

CLCD의 화면에 표시되고 있는 문자를 모두 지웁니다.

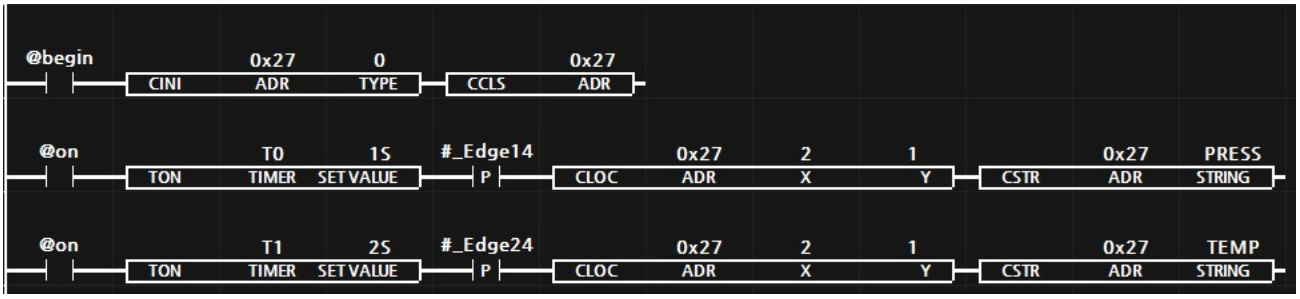


<그림8-2-7-1 CCLS 펑션블럭>

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	화면을 클리어 하고자 하는 CLCD의 주소값 입력

CLCD에 서로 다른 길이의 문자열을 표현할 경우, 기존 문자열이 지원지지 않는 경우가 발생하므로, 주기적으로 CCLS 펑션블럭으로 화면을 클리어 시켜주는 것이 좋습니다.

아래와 같이 프로그램 되었을 경우, 기존 문자열과 새로운 문자열이 겹쳐지는 문제점이 생길 수 있습니다.



<그림8-2-7-2 문자열 중첩>

MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이후, 1초후에 한번만 CLOC와 CSTR 펑션블럭을 실행하여 X좌표:2와 Y좌표:1에 “PRESS"를 표시합니다. 이후, 1초후에 한번만 CLOC와 CSTR 펑션블럭을 실행하여 X좌표:2와 Y좌표:1에 "TEMP"를 표시합니다.

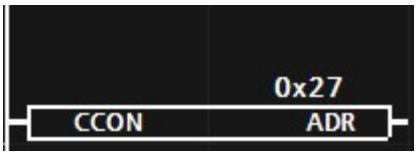
위와 같이 동작했을 경우, 처음에 표시한 “PRESS”보다 뒤에 표시한 “TEMP’의 문자길이가 작아서 CLCD는 “PRESS" -> "TEMP”로 표시되지 않고 “PRESS" -> "TEMPS"로 표시됩니다. "PRESS" 문자에서 “TEMP"문자를 덮어씌우면 “PRESS"의 마지막 글짜인 “S"는 덮어씌워지지 않게 되기 때문입니다.

따라서, 이러한 오류를 격지 않게 하기 위해서는 CLCD를 잘 활용하여 프로그램해야 합니다.

✋ 캐릭터LCD의 특징으로 화면을 클릭어 시키면(CCLS 펑션블럭을 실행하면), 캐릭터 LCD의 커서 좌표가 X좌표:0, Y좌표:0으로 초기화 됩니다. CCLS를 실행 후에는 CLOC 펑션블럭을 사용하여 커서 좌표를 변경해 주시기 바랍니다.

10.2.9. CCON (CLCD Cursor ON)

CLCD에 커서의 표시를 ON 시킵니다.

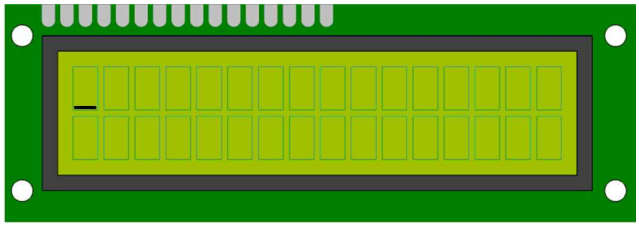


<그림8-2-8-1 CCON 펑션블럭 >

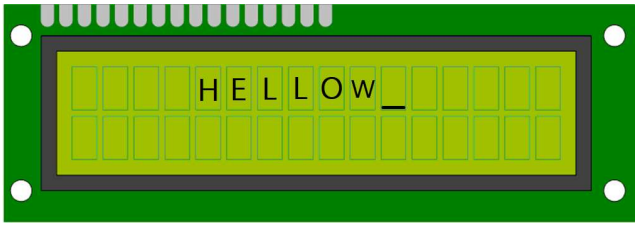
Parameter	Data Type	
ADR	BYTE, WORD, DWORD	커서를 ON시키고자 하는 CLCD의 주소값 입력

커서는 CLCD의 전원이 투입되었을 때 OFF되어 있습니다. 커서를 ON시키고자 할 경우, CCON 펑션블럭을 사용하여 커서를 표시할 수 있고, CCOF 펑션블럭을 사용하여 커서를 OFF시킬 수 있습니다.

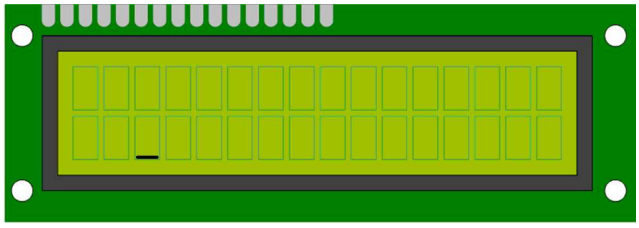
커서는 CLCD의 전원이 투입되었을 때 X:0, Y:0 좌표에 표시됩니다. 이후, 문자열을 표시하면 표시한 문자열의 뒷 쪽에 커서가 표시됩니다. CLOC 펑션블럭을 사용하여 커서의 좌표를 이동시킬 수 있습니다.



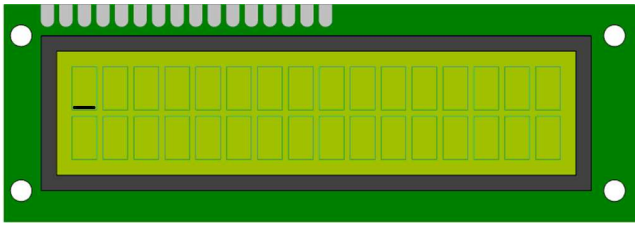
<그림8-2-8-2 커서의 초기위치 >



<그림8-2-8-3 HELLOW 문자열 표시후 커서위치 >



<그림8-2-8-4 CLOC X좌표:2, Y좌표:1의 커서위치 >

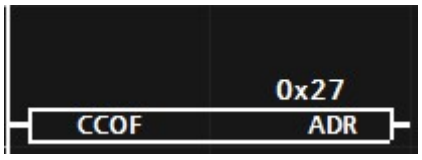


<그림8-2-8-5 CCLS 이후 커서위치 >

CCLS 펑션블럭을 사용하여 CLCD의 화면을 클리어 시키면 커서의 위치는 X좌표:0, Y좌표:0으로 초기화 되므로 주의해야 합니다.

10.2.10. CCOF (CLCD Cursor OFF)

CLCD에 커서의 표시를 OFF 시킵니다.



<그림8-2-9-1 CCOF 펑션블럭 >

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	커서를 OFF시키고자 하는 CLCD의 주소값 입력

커서는 CLCD의 전원이 투입되었을 때 OFF되어 있습니다. 커서를 ON시키고자 할 경우, CCON 펑션블럭을 사용하여 커서를 표시할 수 있고, CCOF 펑션블럭을 사용하여 커서를 OFF시킬 수 있습니다.

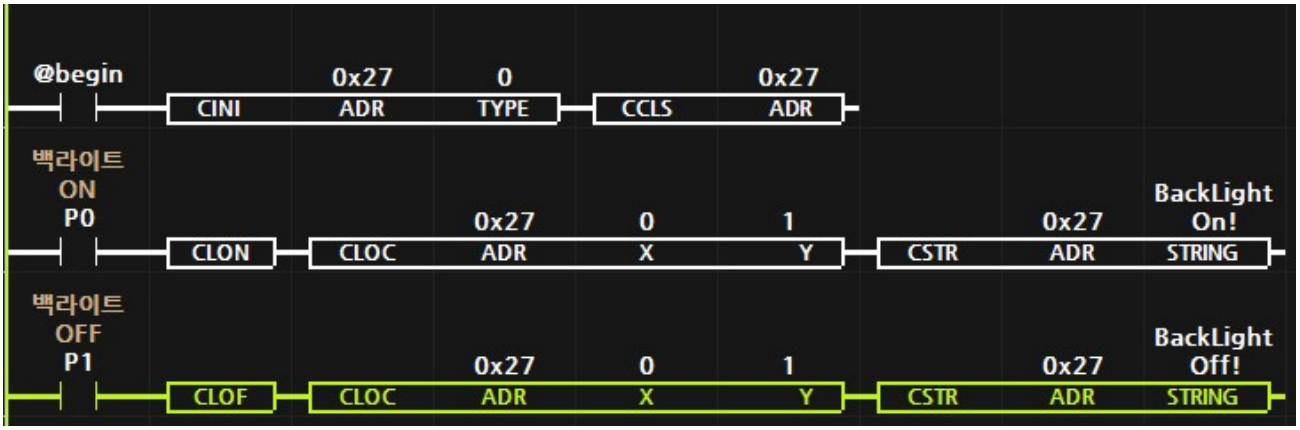
10.2.11. CLON (CLCD Light ON)

CLCD에 BackLight를 ON 시킵니다.

<그림8-2-10-1 CLON 펑션블럭 >

CLCD 초기화 펑션블럭인 CINI 펑션블럭을 실행하면 CLCD의 백라이트는 ON이 됩니다. 이후, CLOF 펑션블럭으로 백라이트를 OFF 시킬 수 있습니다.

CLCD의 여러 명령어를 실행할 때 바로 전에 CLON 펑션블럭이 실행되었다면, 백라이트를 ON 시키고 CLOF 펑션블럭이 실행되었다면, 백라이트를 OFF 시킵니다.



<그림8-2-10-2 CLON 및 CLOF 펑션블럭 사용 예 >

MPS제품에 전원이 투입되면, 최초 1회만 CINI와 CCLS 펑션블럭을 실행하여, CLCD를 초기화 하고 화면을 클리어 시킵니다. 이때, CLCD의 백라이트는 ON되어져 있습니다. 이후, 백라이트OFF:P1 디지털입력이 ON되면 X:0, Y:1 좌표에 "BackLight Off!" 문자열이 표시되면서 백라이트는 OFF가 됩니다. 이후, 백라이트ON:P0 디지털입력이 ON되면 X:0,Y:1 좌표에 "BackLight On!" 문자열이 표시되면서 백라이트는 ON이 됩니다.

10.2.12. CLOF (CLCD Light OFF)

CLCD에 BackLight를 OFF 시킵니다.

〈그림8-2-11-1 CLOF 펑션블럭 〉

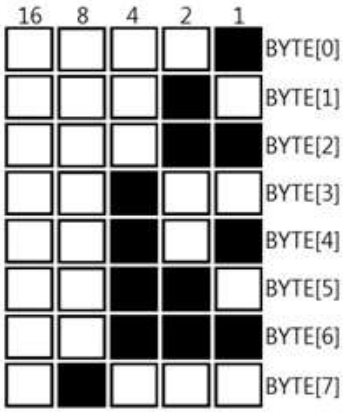
CLCD 초기화 평션블럭인 CINI 펑션블럭을 실행하면 CLCD의 백라이트는 ON이 됩니다. 이후, CLOF 펑션블럭으로 백라이트를 OFF 시킬 수 있습니다.

자세한 사항은 8.2.10. CLON 펑션블럭에서 설명하고 있습니다.

10.2.13. CPUT (CLCD PUT)

캐릭터LCD는 정해진 ASCII 문자열(영문, 숫자, 특수문자)과 일어(카타카나)만을 LCD에 표시할 수 있습니다. 이렇게 정해진 문자열 이외에 사용자가 정의하여 LCD에 표시할 수 있도록 캐릭터LCD는 8개의 사용자 메모리를 내장하고 있습니다.

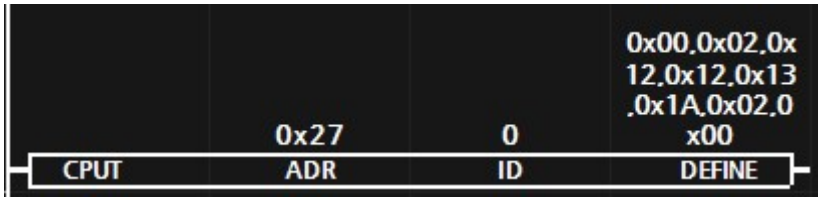
CPUT 펑션블럭은 5x8 도트로 구성된 하나의 문자를 CLCD의 사용자 메모리에 기록시킵니다.
CLCD에서 하나의 문자는 아래와 같이 5x8개의 도트로 구성되어 있습니다.



<그림8-2-12-1 CLCD의 문자 구성 >

이중 맨밑줄(BYTE[7])은 커서로 사용되어집니다. 때문에, 맨밑줄은 제외하면 5x7개의 도트로 구성하는 것이 좋습니다.

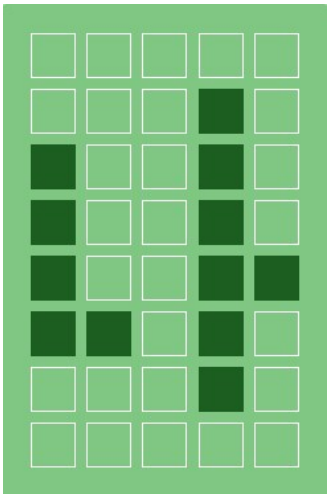
위의 문자는 맨윗줄(BYTE[0])부터 1,2,3,4,5,6,7,8 값으로 표현할 수 있습니다.
한글로 "나"를 표현하고자 할 경우에도 아래와 같이 사용할 수 있습니다.



<그림8-2-12-2 CPUT 펑션블럭 >

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	사용자 문자를 기록시킬 CLCD의 주소값 입력
ID	BYTE, WORD, DWORD	기록시킬 사용자 문자의 ID를 입력 (0 ~ 7)
DEFINE	BYTE x 8	사용자 문자의 형태를 8개의 바이트로 입력

하나의 캐릭터LCD는 최대 8개의 사용자 문자를 사용할 수 있으므로 ID는 0~7을 입력할 수 있습니다. 또한, 하나의 문자는 8개의 행을 바이트 형태로 표현을 해야 합니다. 때문에, DEFINE에는 8개의 바이트를 " , "로 구분하여 입력해야 합니다.



왼쪽의 "나"는 윗줄(1행)부터 아래와 같이 계산됩니다.

0x00,
0x02,
0x12,
0x12,
0x13,
0x1A,
0x02,
0x00

<그림8-2-12-3 사용자 문자 "나" >

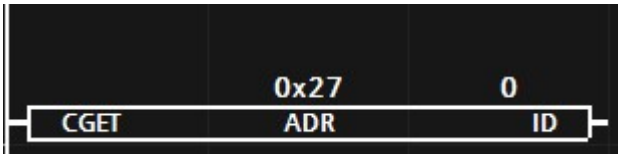
위와 같이 사용자 문자를 8개의 바이트로 만들어 주도록 지원해주는 프로그램이 여러 있습니다. 그중 하나의 주소를 소개합니다. (https://maxpromer.github.io/LCD-Character-Creator/)

✋ CPUT 펑션블럭이 실행된 후, 캐릭터 LCD의 커서가 X:0, Y:0 좌표로 초기화 됩니다. CPUT 펑션블럭을 실행 후에는 CLOC 펑션블럭을 사용하여 커서 좌표를 변경해 주시기 바랍니다.

✋ CPUT 펑션블럭을 실행하여 캐릭터 LCD에 기록시킨 사용자 정의 문자는 캐릭터 LCD의 전원이 차단되면 데이터가 상실됩니다.

10.2.14. CGET (CLCD GET)

CPUT 펑션블럭을 통하여 기록시킨 사용자 문자를 LCD에 표시하는 펑션블럭입니다.



<그림8-2-13-1 CGET 펑션블럭 >

Parameter	Data Type	
ADR	BYTE, WORD, DWORD	사용자 문자를 표시시킬 CLCD의 주소값 입력
ID	BYTE, WORD, DWORD	표시할 사용자 문자의 ID를 입력 (0 ~ 7)

캐릭터LCD는 8개의 사용자 문자를 기록하고 표시할 수 있습니다. CPUT에서도 사용자가 정의한 5열x8행으로 구성된 문자를 0 ~ 7 의 ID로 기록시키고, CGET으로 기록시킨 문자를 0 ~ 7의 ID로 불러와서 LCD에 표시할 수 있습니다.

10.3. FND (7SEGMENT)

FND (7SEGMENT)는 7개의 LED로 숫자를 표시할 수 있는 표시장치입니다. 하나의 숫자를 표시하기 위하여 7개의 LED를 제어해야 하므로 제어를 위한 신호선이 많이 필요한 단점이 있습니다. 현대에는 TM1637 I.C를 사용하여 2가닥의 제어선만을 사용하여 FND를 제어할 수 있습니다.

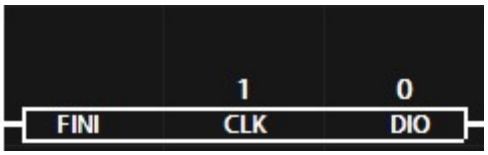
<그림8-3-1 FND 0.56" 시계기능 >

<그림8-3-2 FND 0.36" 소수점기능 >

TM1637 I.C를 사용해도 하나의 I.C로 표현할 수 있는 수의 제한이 있어 시계기능을 위한 제품과 소수점을 위한 제품으로 나누어져 있습니다.

10.3.1. FINI (FND Initialize)

FND를 사용하기 위해서는 FINI 펑션블럭을 실행하여 최초 1회 초기화를 해야 합니다.



<그림8-3-1-1 FINI 펑션블럭 >

Parameter	Data Type	
CLK	Unsigned BYTE(상수)	CLOCK 통신라인으로 사용할 포트번호
DIO	Unsigned BYTE(상수)	DATA IN/OUT 통신라인으로 사용할 포트번호

MPS제품과 FND제품은 CLK와 DIO 2가닥의 통신선을 연결해야 하며, MPS제품에서 사용할 수 있는 통신포트 번호는 아래와 같습니다.

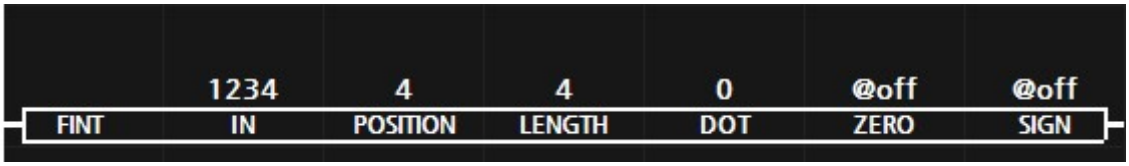
Products	PORT	
MPS-8A4R	AI0 ~ AI7	0~5V 아날로그 입력포트에서만 사용이 가능합니다.
MPS-16A8R	AI0 ~ AI3	0~5V 아날로그 입력포트에서만 사용이 가능합니다.
MPS-8A8R	PWM0 ~ PWM5	PWM 출력포트에서만 사용이 가능합니다.
MPS-16A8R8T	PWM0,1,3,4	PWM 출력포트에서만 사용이 가능합니다.
MPA 모든제품	UART 통신포트	UART 통신프로트를 통하여 1개의 FND 사용이 가능합니다. TXD : DIO RXD : CLK **단, 주문시 유선으로 연락을 주셔야 사용이 가능하도록 변경하여 제품을 발송해드립니다.**

MPS-8A4R-R2, MPS-8A4T-R2, MPS-16A8R-R2, MPS-16A8T-R2 제품에서는 사용이 불가능합니다.

MPA 제품군은 CLK와 DIO에 입력한 숫자와 상관없이 UART 통신포트로 연결이 됩니다. 때문에, CLK와 DIO 파라미터에 아무 숫자나 입력하시고 사용해 주시기 바랍니다.

10.3.2. FINT (FND Integer)

숫자를 표시하기 위한 펑션블럭입니다.



<그림8-3-2-1 FINT 펑션블럭 >

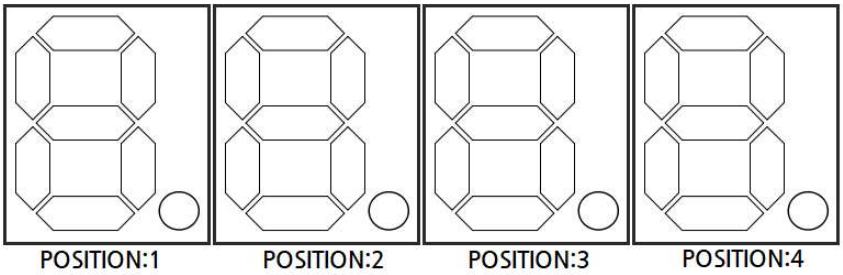
Parameter	Data Type	
IN	Signed BYTE, WORD	표시할 숫자의 값
POSITION	Unsigned BYTE	표시할 숫자의 일의 자리의 FND 위치 (1~4)
LENGTH	Unsigned BYTE	표시할 숫자의 길이 (1~4)
DOT	Unsigned BYTE	표시할 숫자중 점의 위치 (0:사용안함/1~4)
ZERO	BIT	표시할 숫자의 앞에 위치한 0의 표시유무
SIGN	BIT	표시할 숫자의 음기호 표시 유무

< IN >

입력한 숫자 및 메모리의 값으로 -999 ~ 9999까지 표현이 가능합니다.

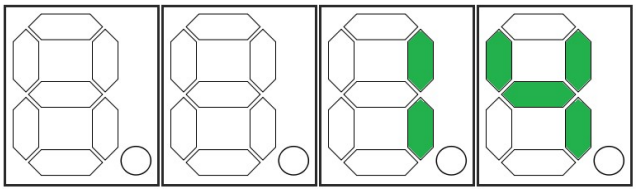
< POSITION >

숫자의 일의 자리의 위치를 지정합니다. 1~4의 입력이 가능합니다.

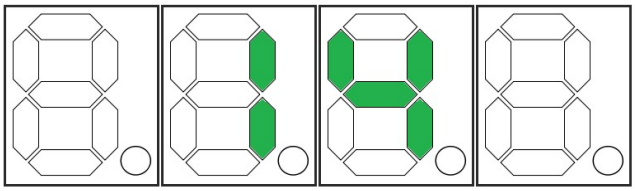


<그림8-3-2-2 FND 위치값 >

IN이 14이고 POSITION이 4의 경우와 3의 경우를 아래 그림으로 확인해 주세요.



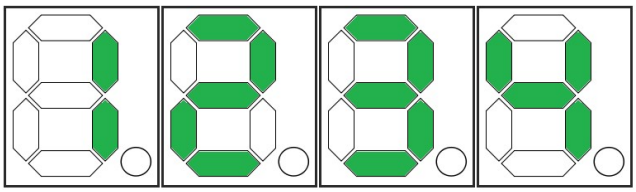
<그림8-3-2-3 Position이 4의 경우 >



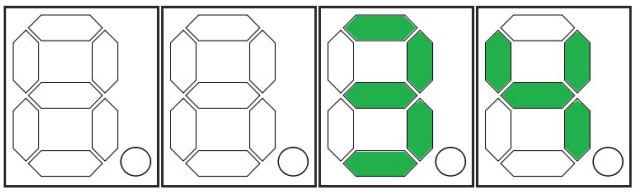
<그림8-3-2-4 Position이 3의 경우 >

< LENGTH >

숫자의 길이를 지정합니다. IN이 1234이고 LENGTH가 4의 경우와 2의 경우를 아래 그림으로 확인해 주세요.



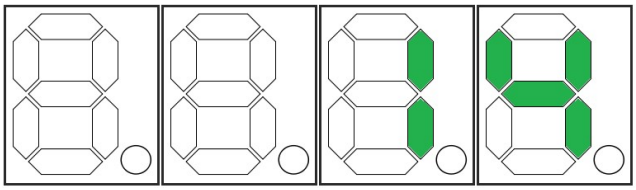
<그림8-3-2-5 Length가 4의 경우 >



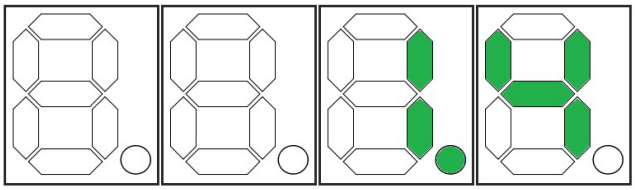
<그림8-3-2-6 Length가 2의 경우 >

< DOT >

숫자의 소수점의 위치를 지정합니다. 소수점 표시를 하지 않을 경우 0을 기입해주세요. IN이 14이고 DOT이 0의 경우와 3의 경우를 아래 그림으로 확인해 주세요.



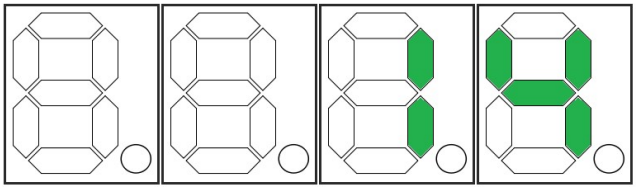
<그림8-3-2-7 DOT이 0의 경우 >



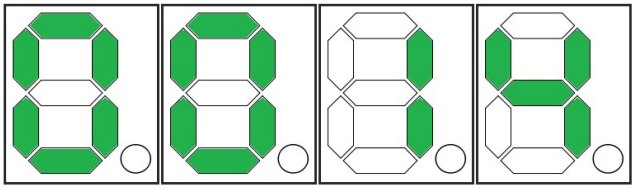
<그림8-3-2-8 DOT이 3의 경우 >

< ZERO >

숫자의 앞에 위치한 0의 표시의 유무를 지정합니다. IN이 14이고 ZERO가 @OFF의 경우와 @ON의 경우를 아래 그림으로 확인해 주세요.



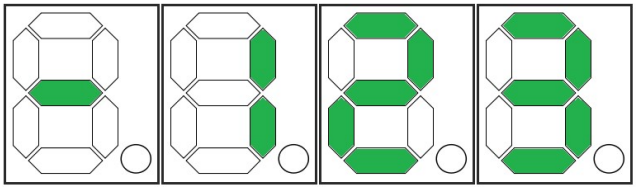
<그림8-3-2-9 ZERO가 @OFF 경우 >



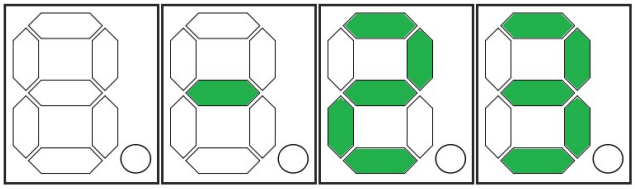
<그림8-3-2-10 ZERO가 @ON 경우 >

< SIGN >

숫자의 음기호 표시의 유무를 지정합니다. SIGN은 LENGTH의 수치와 무관하게 표시됩니다. IN이 -123이고 SIGN이 @ON이고 LENGTH가 3의 경우와 2의 경우를 아래 그림으로 확인해 주세요.



<그림8-3-2-11 LENGTH가 3의 경우 >

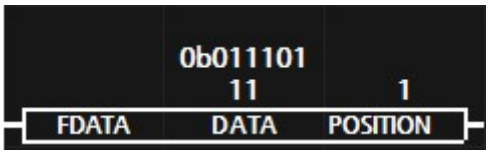


<그림8-3-2-12 LENGTH가 2의 경우 >

SIGN이 @ON일 때 음수만이 -로 표시됩니다. 양수 +의 경우 FND에서 표현할 수 없기 때문입니다.

10.3.3. FDATA (FND DATA)

FND의 7개의 세그먼트의 LED를 개별적으로 제어하여 원하는 모양을 표현할 수 있습니다.



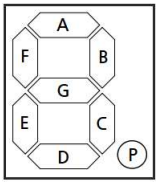
<그림8-3-3-1 FDATA 펑션블럭 >

Parameter	Data Type	
DATA	Unsigned BYTE	표시할 FND 값
POSITION	Unsigned BYTE	표시할 숫자의 일의 자리의 FND 위치 (1~4)

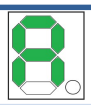
< DATA >

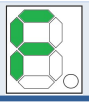
FND의 7개의 LED를 제어하여 원하는 모양의 글씨를 만들어 사용할 수 있습니다. 또한 추가적인 1개의 LED를 사용하여 점을 표현할 수 있습니다. 따라서, 총 8개의 LED의 제어가 가능합니다.

8개의 비트를 사용하여 각각의 LED를 1:1로 제어해야 할 수 있으며 각각의 비트가 제어하는 LED의 위치를 아래 그림에서 확인할 수 있습니다.



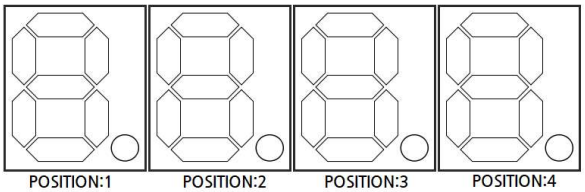
<그림8-3-3-2 FDATA 펑션블럭 >

FND		2진수								16진수
		P	G	F	E	D	C	B	A	
A		0	1	1	1	0	1	1	1	0x77
b		0	1	1	1	1	1	0	0	0x7C
C		0	0	1	1	1	0	0	1	0x39
d		0	1	0	1	1	1	1	0	0x5E
E		0	1	1	1	1	0	0	1	0x79

F		0	1	1	1	0	0	0	1	0x71

< POSITION >

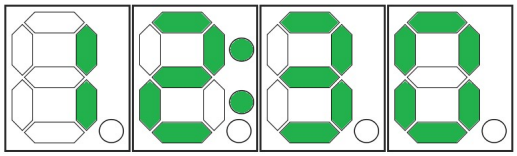
DATA 파리미터에 입력한 값이표시될 FND의 위치를 지정합니다. 1부터 4까지 사용이 가능합니다.



<그림8-3-2-2 FND 위치값 >

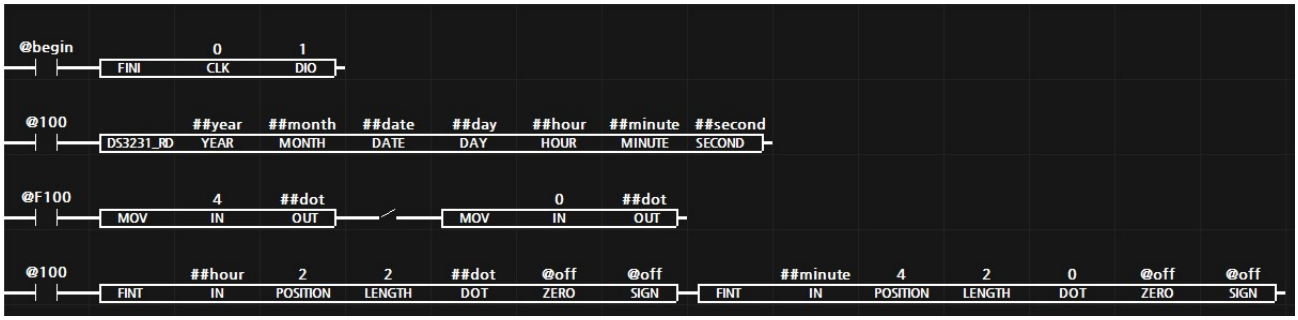
10.3.4. FND 시계표시

시계기능이 있는 FND의 경우 2번째에 시계의 도트 LED 2개가 위치에 있습니다. 때문에, FINT 펑션블럭에서는 DOT에 2를 입력하면 항상 도트 LED가 아래와 같이 ON이 됩니다. 하지만, "시"와 "분을" 별도로 표기할 경우, "시"를 표시할 때 초침 LED를 On/Off해야하며 이때 "시"의 입장에서는 일의 자리의 LED가 켜져야 하므로 4를 입력해야 합니다.



<그림8-3-3-3 FND 위치값 >

초침 LED를 1초 간격으로 점멸하기 위해서 DOT 파리미터를 워드 메모리를 입력해 주시기 바랍니다. DS3231 RTC를 사용하여 시간을 읽어오고 읽어온 시간을 FND에 표시하는 예는 아래와 같습니다.



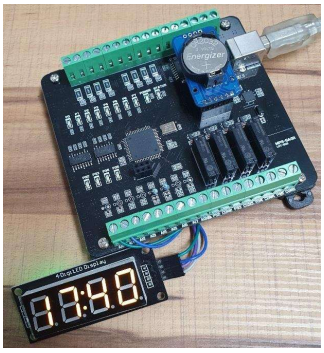
<그림8-3-3-4 FND 위치값 >

<1 Line> FND를 CLK를 A0포트로 DIO를 A1포트로 정의하고 초기화 합니다. (MPS-8A4R 기준)

<2 Line> DS3231 모듈에서 시간 파라미터를 읽어서 각각의 사용자 정의 메모리에 기록합니다.

<3 Line> 1초동안 사용자 정의 메모리 ##dot에 4를 기록하고 다시 1초동안 ##dot에 0을 기록하여 시계의 도트 LED를 1초마다 깜빡이게 합니다.
여기서 도트 LED는 POSITION:2에 위치하지만 4 Line에서 ##year를 입력하는 기준에서는 4번째의 FND에 위치하므로 4를 입력해야 합니다.

<4 Line> 1초마다 DS3231에서 읽어온 시간에서 ##year를 FND의 1~2번째에 표시하고 ##minute를 3~4번째에 표시합니다.



<그림8-3-3-5 FND와 DS321를 사용하여 시간(17시40분)을 표시 >