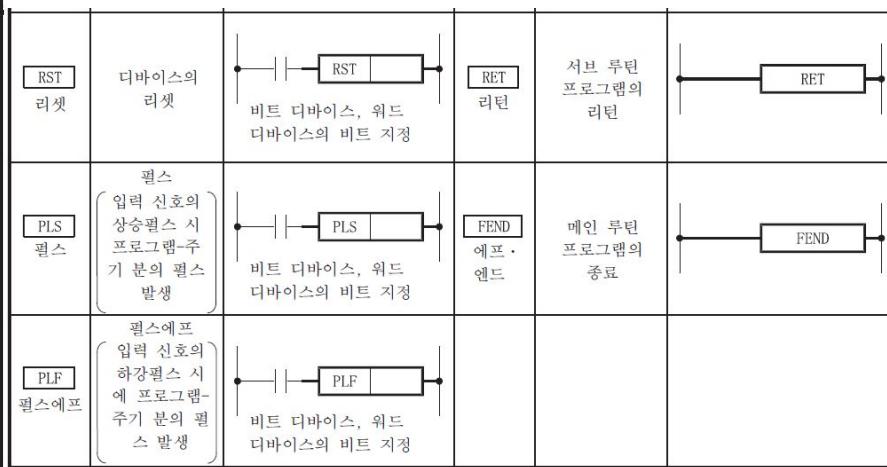


시퀀스 명령과 기본 명령 1

명령 기호 (호칭)	기 능	도면 표시(사용 디바이스)	명령 기호 (호칭)	기 능	도면 표시(사용 디바이스)
OUT 아웃	코일 출력	비트 디바이스, 워드 디바이스의 비트 지정	CJ 시제이	조건 점포 (즉시 실행형)	CJ Pn n=0~4095 포인터
MC 마스터 컨트롤	마스터 컨트롤 시작*1	비트 디바이스, 워드 디바이스의 비트 지정 Nn MC Nn n=0~14 네스팅	SCJ 에스 · 시제이	조건 점포 1스캔 후 실행형	SCJ Pn n=0~4095 포인터
MCR 마스터 컨트롤 리셋	마스터 컨트롤 종료	MCR Nn n=0~14 네스팅	CALL 콜	서브 루틴 프로그램의 콜	CALL Pn n=0~4095 포인터
SET 세트	디바이스 세트	SET 비트 디바이스, 워드 디바이스의 비트 지정	CALLP 콜피	서브 루틴 프로그램의 호출 (펄스 동작)	CALLP Pn n=0~4095 포인터



>> OUT / SET / RST

- SET 명령은 해당 디바이스를 ON 시킴
- 입력 조건이 OFF 되어도 ON 상태를 유지
- RST 명령으로 OFF 시킴

제3장 : 명령어 사용하기

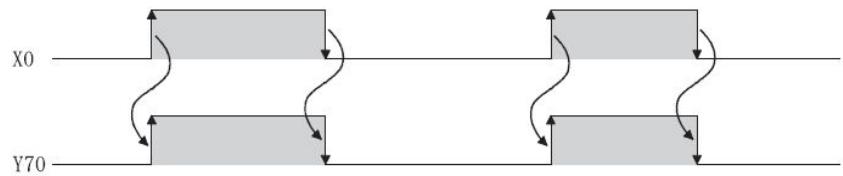
OUT 과 SET · RST 의 차이

[OUT 명령]

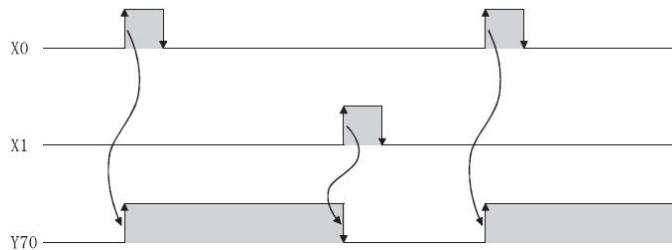


• OUT 명령은 입력 조건이 "ON"되면 지정 디바이스를 "ON"하고, 입력 조건이 "OFF"된 시점에서 지정 디바이스도 "OFF"합니다.

[타이밍 차트]



[타이밍 차트]

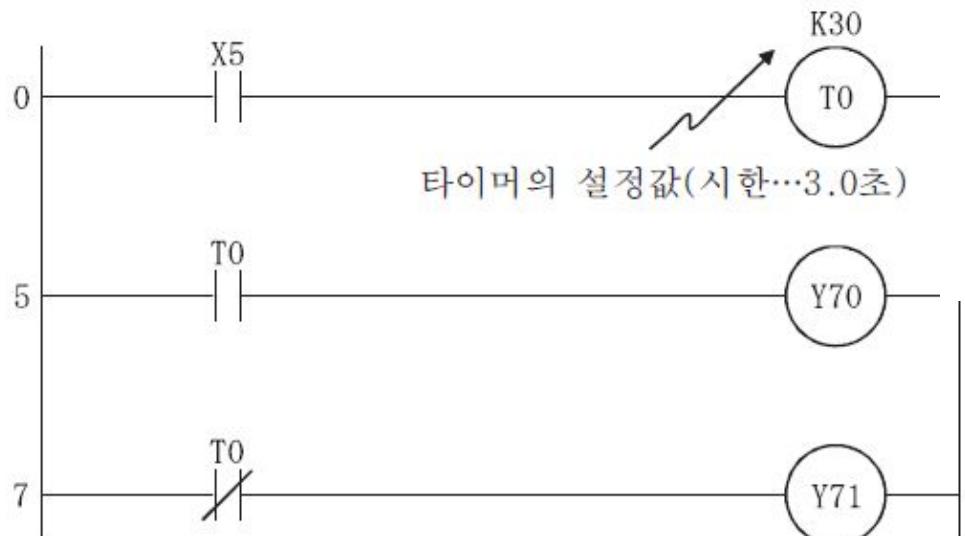


» OUT / SET / RST



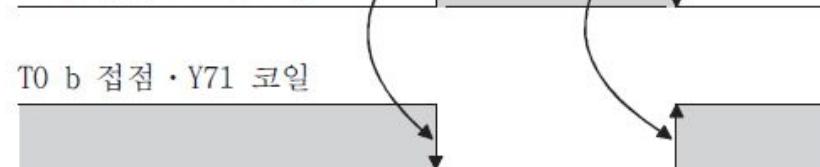
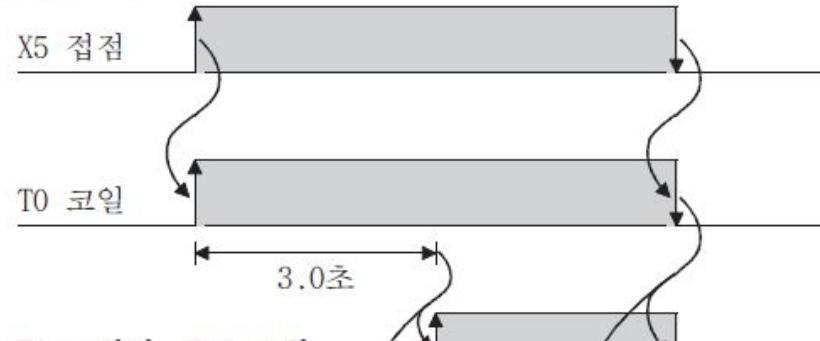
제3장 : 명령어 사용하기

타이머 사용하기



* OUT T는 4스텝 명령입니다.

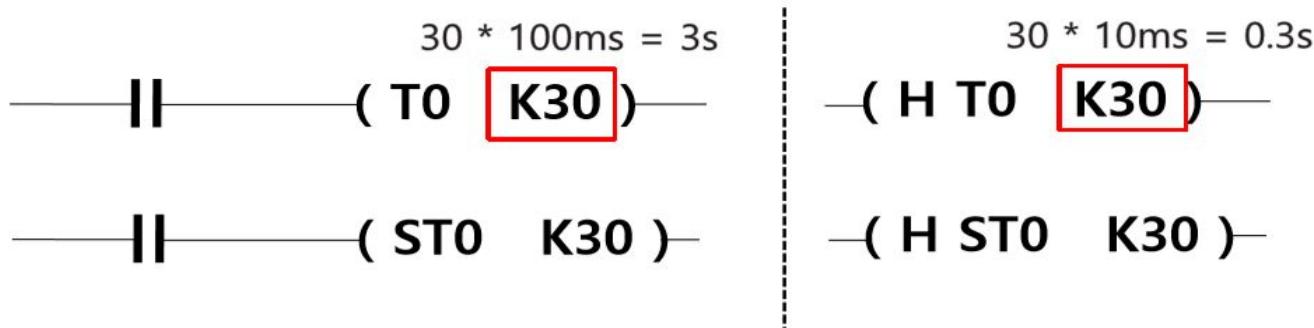
[타이밍 차트]



종 류	타이머 번호(초기값)
저속 타이머 100ms 단위로 타임 카운트	초기값 T0-T2047(2048개)
고속 타이머 10ms 단위로 타임 카운트	
저속 적산 타이머... 100ms 단위로 타임을 적산	초기값 0개 파라미터에서 변경 가능
고속 적산 타이머... 10ms 단위로 타임을 적산	

일반 / 적산 / 고속타이머 이해하기

» 타이머 T / ST



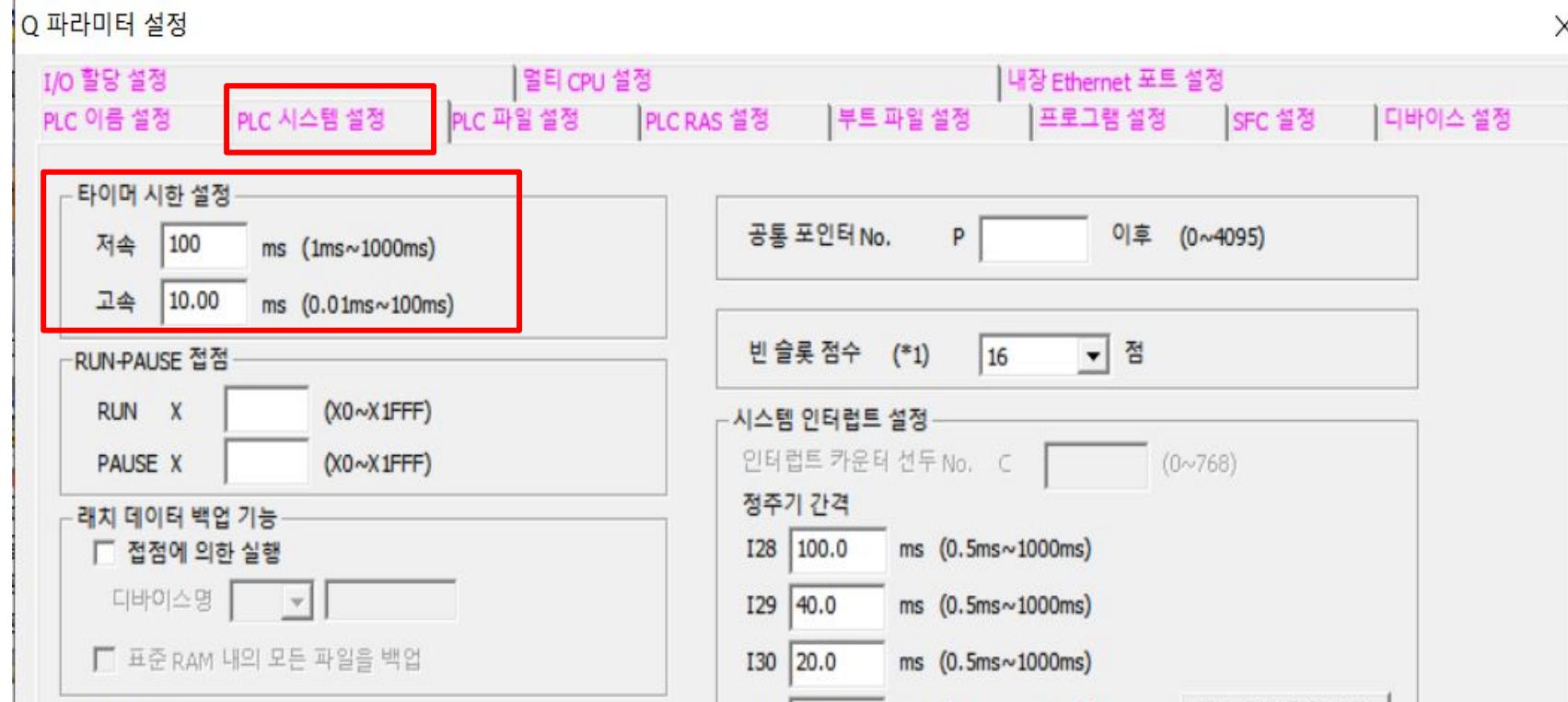
- T(타이머) : 타이머 코일이 ON 되면 지정 시간만큼 계측을 한 뒤, 타이머 접점 상태 바뀜
- ST(적산 타이머) : 입력 신호가 OFF 되어도 값 유지
(적산 타이머 사용하려면 **파라미터 설정 필요**)
- 시간 설정 값 : K1~K32767

저속 타이머(T)	디폴트 100ms
고속 타이머(H T)	디폴트 10ms
저속 적산 타이머(ST)	파라미터 설정 필요
고속 적산 타이머(H ST)	파라미터 설정 필요

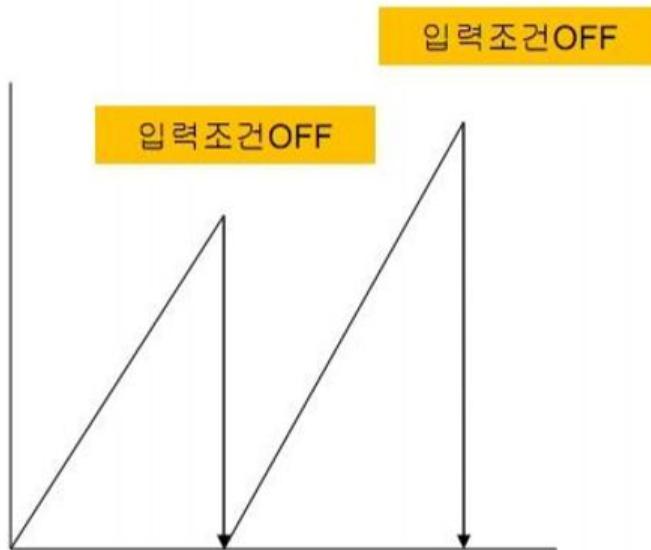
PLC parameter → PLC system

제3장 : 명령어 사용하기

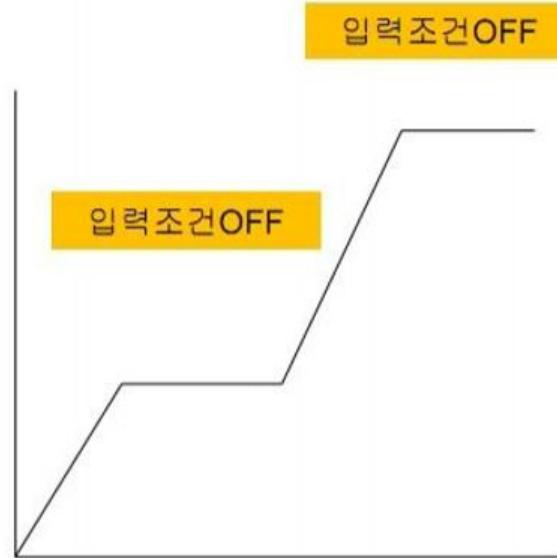
일반 / 적산 / 고속타이머 이해하기



제3장 : 명령어 사용하기



타이머(T)



적산 타이머(ST)

제3장 : 명령어 사용하기

적산타이머 디바이스 설정하기

Q 파라미터 설정

I/O 할당 설정			멀티 CPU 설정			내장 Ethernet 포트 설정			
PLC 이름 설정			PLC 시스템 설정			PLC 파일 설정			
			PLC RAS 설정			부트 파일 설정			
						프로그램 설정			
						SFC 설정			
						디바이스 설정			
	기호	진	디바이스 점수	래치(1) 선두	래치(1) 최종	래치(2) 선두	래치(2) 최종	로컬 디바이스 선두	로컬 디바이스 최종
입력 릴레이	X	16	8K						
출력 릴레이	Y	16	8K						
내부 릴레이	M	10	8K						
래치 릴레이	L	10	8K						
링크 릴레이	B	16	8K						
어년시에이터	F	10	2K						
링크 특수	SB	16	2K						
에지 릴레이	V	10	2K						
스텝 릴레이	S	10	8K						
타이머	T	10	2K						
적산 타이머	ST	10	0K						
카운터	C	10	1K						
데이터 레지스터	D	10	12K						
링크 레지스터	W	16	8K						
링크 특수	SW	16	2K						
인덱스	Z	10	20						

디바이스 합계: 28.8 K워드

워드 디바이스: 25.0 K워드

비트 디바이스: 44.0 K비트

디바이스 점수의 합계는 29K워드까지입니다.

래치(1): 래치 클리어에서 클리어가 가능합니다.
 래치(2): 래치 클리어에서 클리어가 불가능합니다. 프로그램에서 클리어를 실행하십시오.
 래치 범위 설정만큼(L을 포함) 스캔 타임이 연장됩니다.
 래치 할 필요가 있는 경우 래치 범위를 필요 최저한으로 설정하십시오.
 로컬 디바이스 사용 시는 PLC 파일 설정에서 파일의 설정을 실행하십시오.

적산타이머 디바이스 설정하기

Q 파라미터 설정

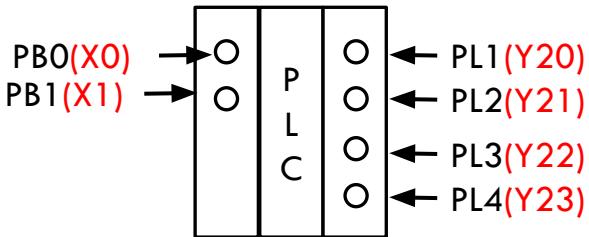
	I/O 할당 설정	멀티 CPU 설정			
PLC 이름 설정	PLC 시스템 설정	PLC 파일 설정	PLC RAS 설정	부트 파	
입력 릴레이	X	16	8K		
출력 릴레이	Y	16	8K		
내부 릴레이	M	10	8K		
래치 릴레이	L	10	8K		
링크 릴레이	B	16	8K		
어년시에이터	F	10	2K		
링크 특수	SB	16	2K		
에지 릴레이	V	10	2K		
스텝 릴레이	S	10	8K		
타이머	T	10	2K		
적산 타이머	ST	10	1K	0k → 1k로 수정	
카운터	C	10	1K		
데이터 레지스터	D	10	11K	12k → 11k로 수정	
링크 레지스터	W	16	8K		
링크 특수	SW	16	2K		
인덱스	Z	10	20		

타이머 실습하기



제3장 : 명령어 사용하기

프로그램 연습하기 1



PB0(X0)		■																							
PB1(X1)																							■		
PL1(Y20)		■	■	■	■																				
PL2(Y21)		■	■	■	■																				
PL3(Y22)		■	■	■	■																				
PL4(Y23)		■	■	■	■																				
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

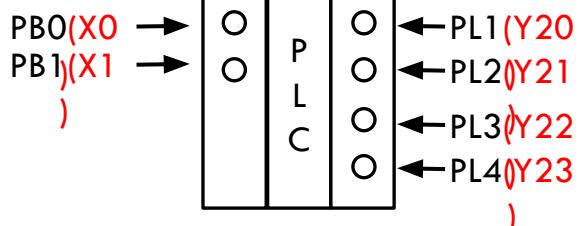
X0 스위치를 누르면 즉시 Y20, Y21, Y22, Y23에 있는 램프에 불이 켜진다.

5초후 Y20 램프가 꺼지고, 이후 5초마다 Y21, Y22, Y23이 꺼진다.

X1이 눌려지만 즉시 모든 램프가 꺼진다.

제3장 : 명령어 사용하기

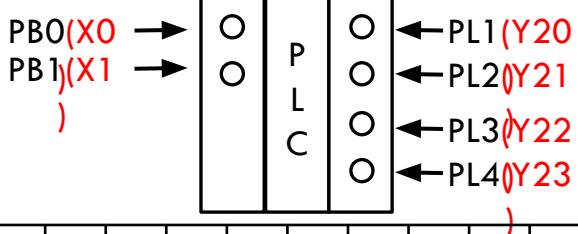
프로그램 연습하기2



PB0(X0)		■																							
PB1(X1)																							■		
PL1(Y20)		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
PL2(Y21)				■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
PL3(Y22)					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
PL4(Y23)						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

제3장 : 명령어 사용하기

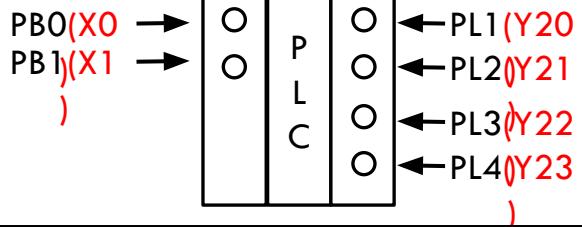
프로그램 연습하기3



PB0(X0)																									
PB1(X1)																									
PL1(Y20)																									
PL2(Y21)																									
PL3(Y22)																									
PL4(Y23)																									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

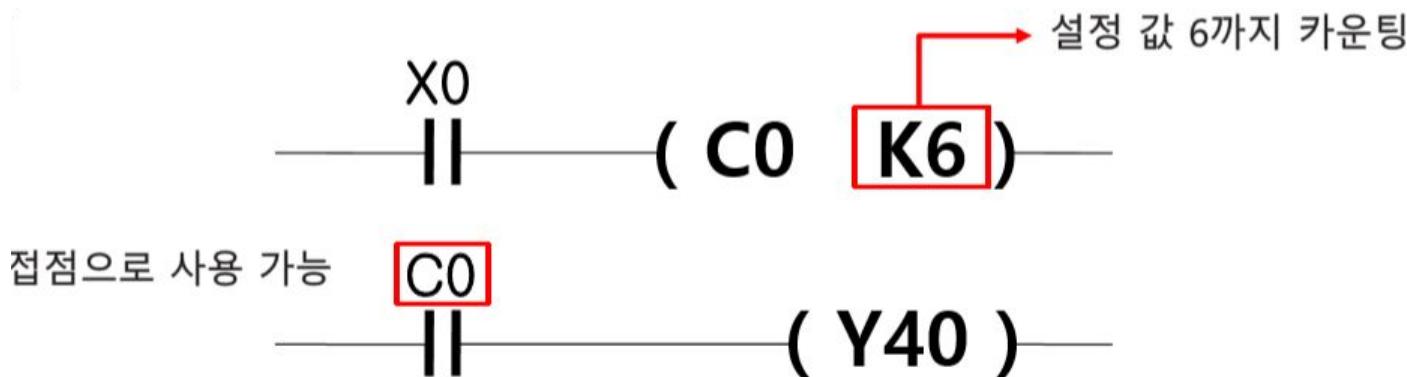
제3장 : 명령어 사용하기

프로그램 연습하기4



PB0(P0)		■																							
PB1(P1)																									
PL1(P40)		■	■	■	■	■																			
PL2(P41)				■	■	■	■	■	■	■															
PL3(P42)											■	■	■	■	■	■	■								
PL4(P43)												■	■	■	■	■									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

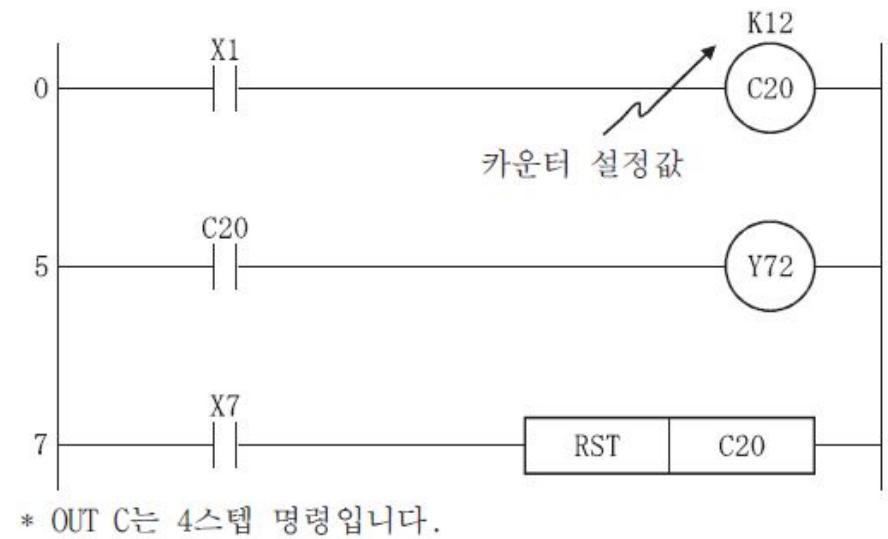
카운터 이해하기



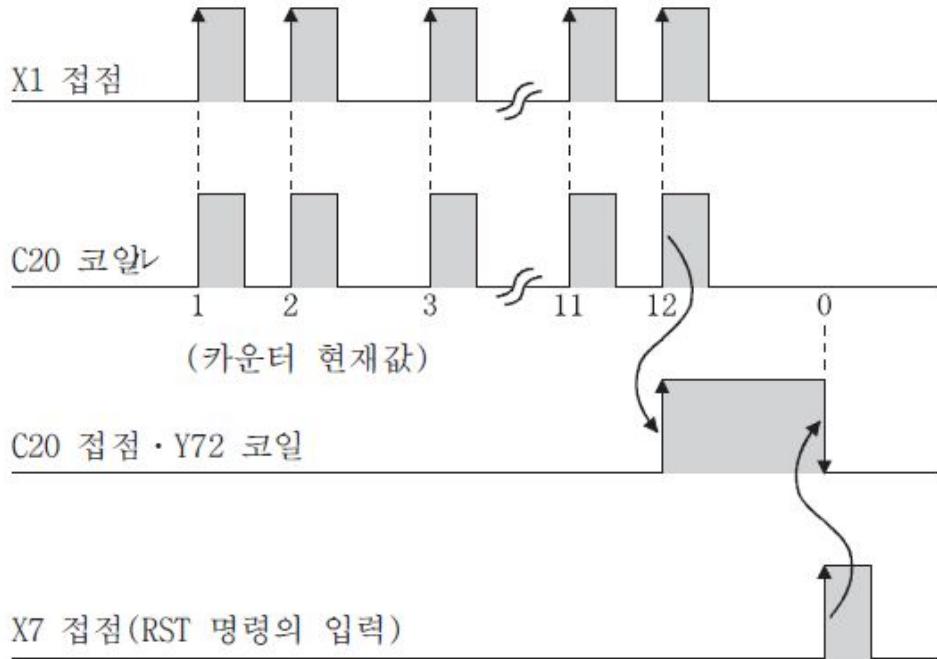
- 입력 조건이 OFF → ON (**상승 펄스**) 일 때 카운터 현재 값 1씩 **증가**
- 현재 값과 설정 값이 **같아질 때 까지** 증가
- 현재 값과 설정 값이 **같아지면** C0 카운터 **접점이 ON** 된 후 상태 유지
- **RST 명령**이 실행될 때까지 접점의 상태와 카운터의 **현재 값 유지**
- 카운터의 설정 값은 **양수**만 가능
- D(**데이터 레지스터**)에 의한 **간접 설정** 가능

제3장 : 명령어 사용하기

카운터 이해하기



[타이밍 차트]



카운터 실습하기



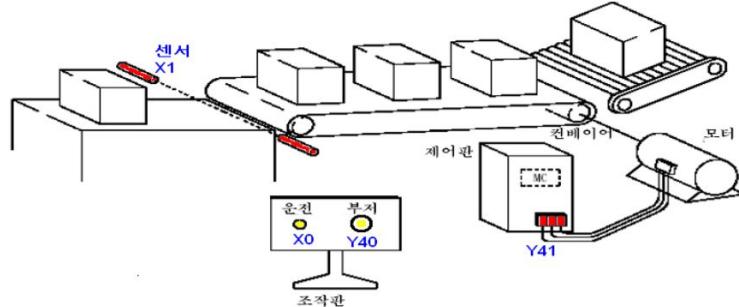
제3장 : 명령어 사용하기

카운터 실습하기

래더 예

컨베이어의 운전 지령(X0) 스위치를 ON 하면, 3초간 부저(Y70)가 울리고 이후 컨베이어가 운전(Y71)을 시작합니다.

제품이 6개 반송된 것을 검출(X1)하면 컨베이어는 자동으로 정지합니다.

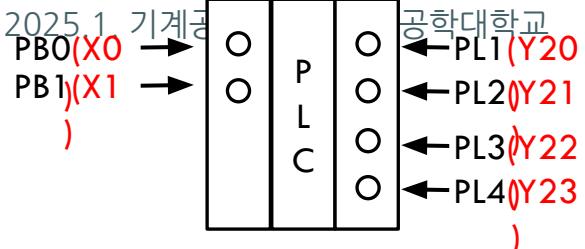


카운터 실습하기



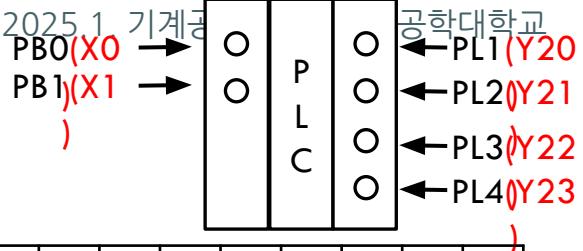
제3장 : 명령어 사용하기

프로그램 연습1 ; 타이머 카운터



PB0(X0)		1			4		6		9																
PB1(X1)																			19		20				
PL1(Y20)		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19					
PL2(Y21)																									
PL3(Y22)																									
PL4(Y23)																									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

제3장 : 명령어 사용하기

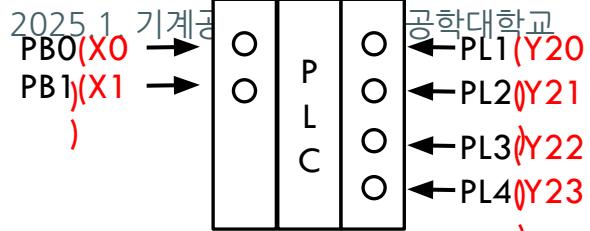


프로그램 연습2 ; 타이머 카운터

PB0(X0)																									
PB1(X1)																									
PL1(Y20)																									
PL2(Y21)																									
PL3(Y22)																									
PL4(Y23)																									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

제3장 : 명령어 사용하기

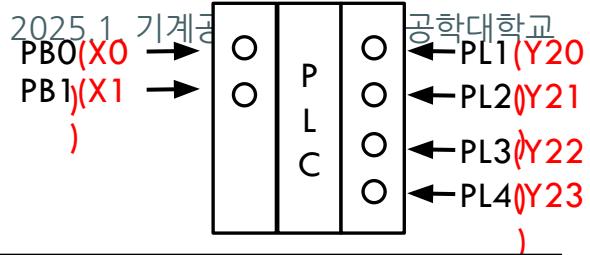
프로그램 연습3 ; 타이머 카운터



PB0(X0)		█																	█						
PB1(X1)																									
PL1(Y20)		█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
PL2(Y21)																									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

제3장 : 명령어 사용하기

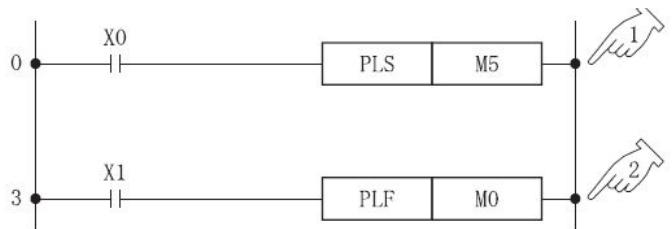
프로그램 연습5 ; 타이머 카운터



PB0(P0)		■																							
PB1(P1)																									
PL1(P40)		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		
PL2(P41)																									
PL2(P42)																									
Time(s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

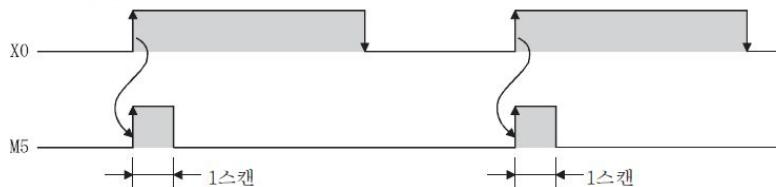
제3장 : 명령어 사용하기

PLS, PLF 이해하기

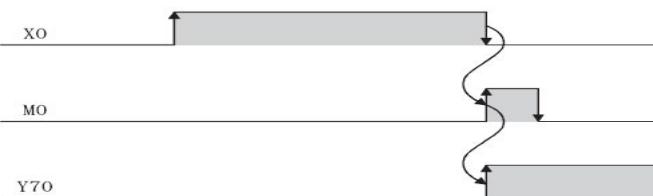


PLS 펄스(입력 조건의 상승펄스 시 지정 디바이스의 1스캔 ON 명령)

[타이밍 차트]

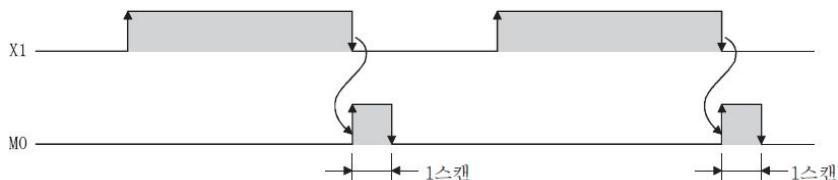


[타이밍 차트]

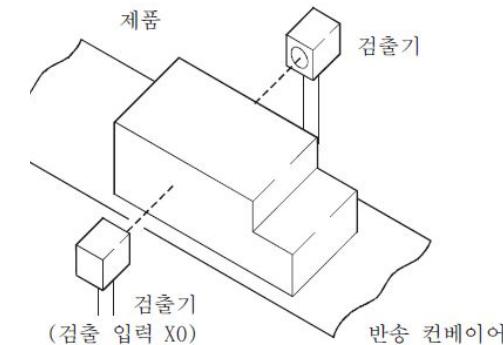
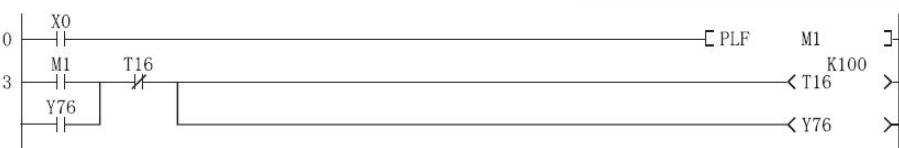


PLF 펄스에프(입력 조건의 하강펄스 시 지정 디바이스의 1스캔 ON 명령)

[타이밍 차트]



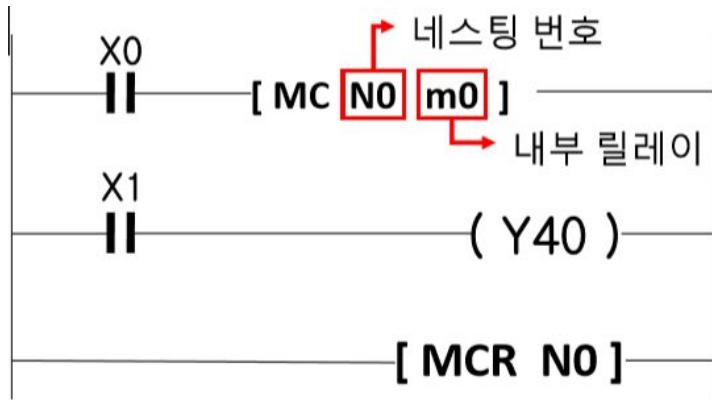
[프로그램 예]



제3장 : 명령어 사용하기

MC 마스터 컨트롤 (시작)

MCR 마스터 컨트롤 리셋 (종료)

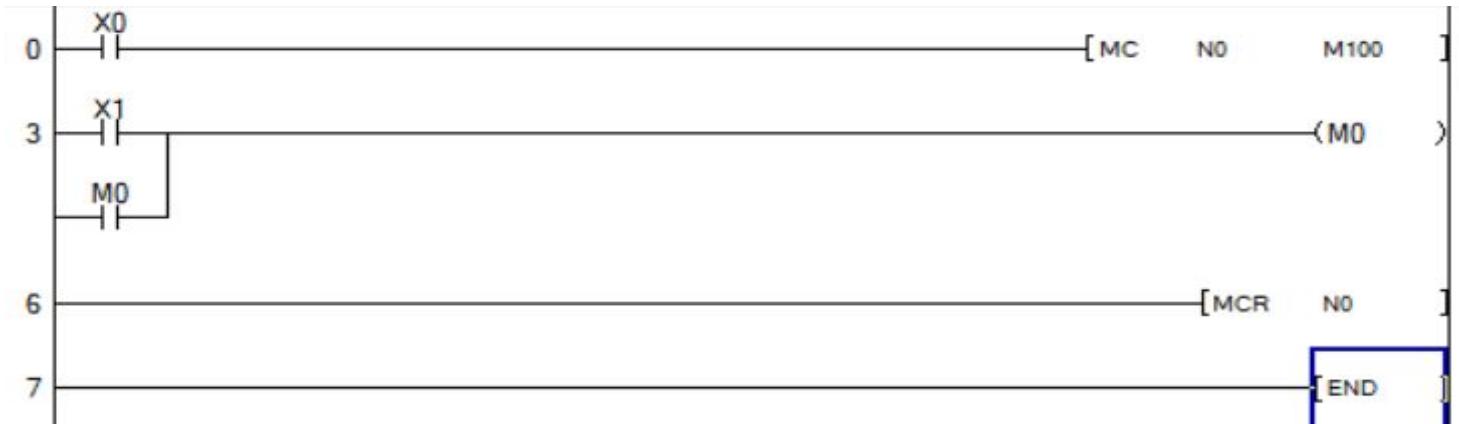


- MC (마스터컨트롤) : 구문시작
 - MCR (마스터 컨트롤 리셋) : 끝
 - N_(0~14) : 네스팅 번호
 - M_ : 접점(Y/M/L/S/B/F)
- *MC-MCR OFF시 값
- 모두 OFF : OUT 명령(코일출력)
 - 상태유지 ; SET,RST, SFT명령,
 - 카운터 값, 적산타이머 값
 - 수치가 0: 일반타이머, 고속타이머

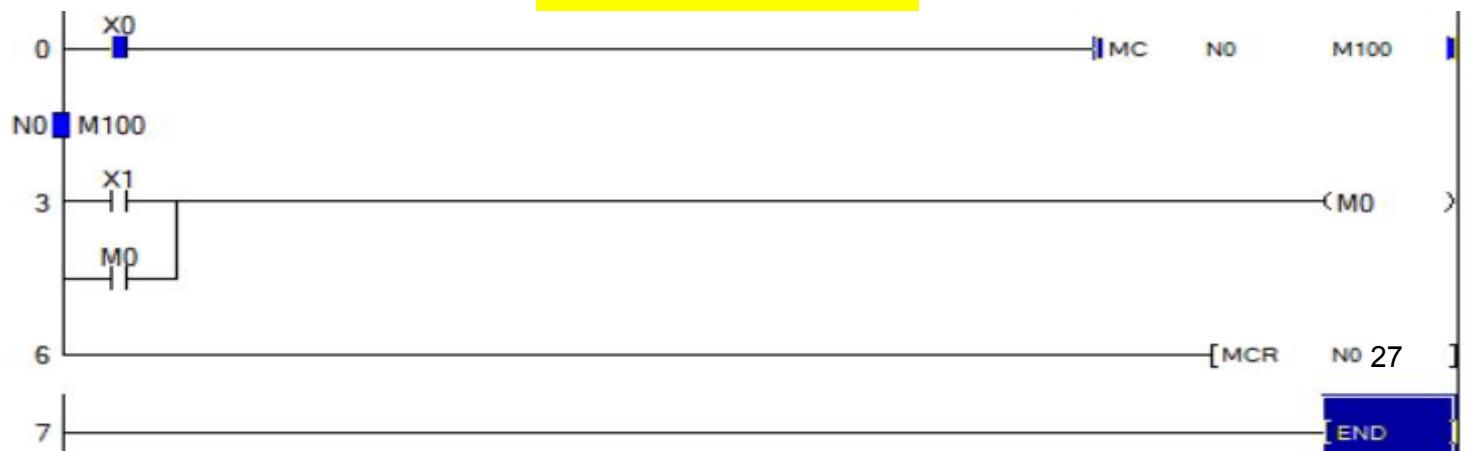
- MC 안에 있는 래더는 MC(마스터 컨트롤)의 통제를 받음
- MC가 동작하고 있지 않다면 MC 안에 있는 명령어 또한, 동작하지 않음
- MCR을 통해 리셋 가능

제3장 : 명령어 사용하기

프로그램 작성



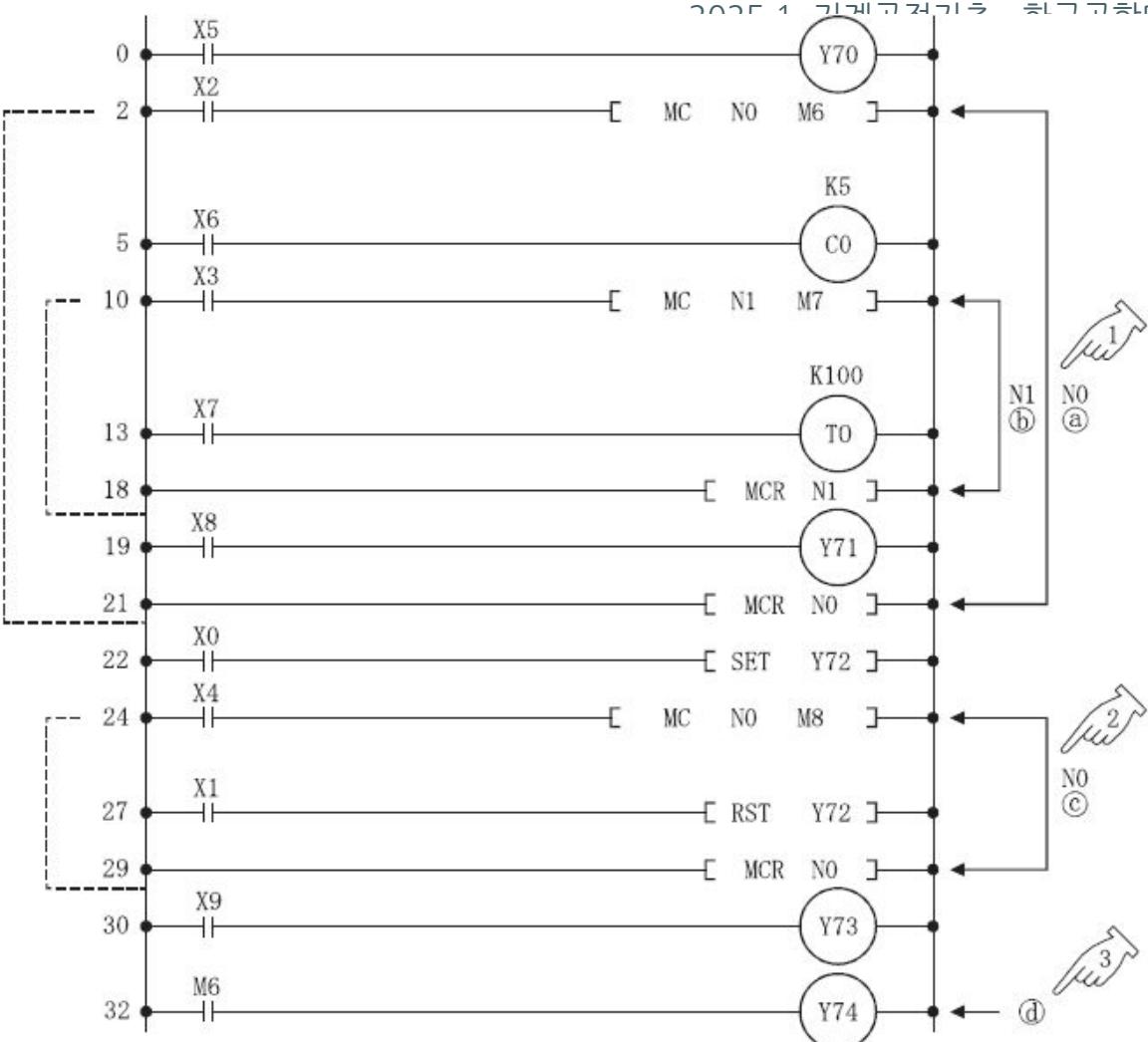
프로그램 시뮬레이션



제3장 : 명령어 사용하기

MC 마스터 컨트롤 (시작)

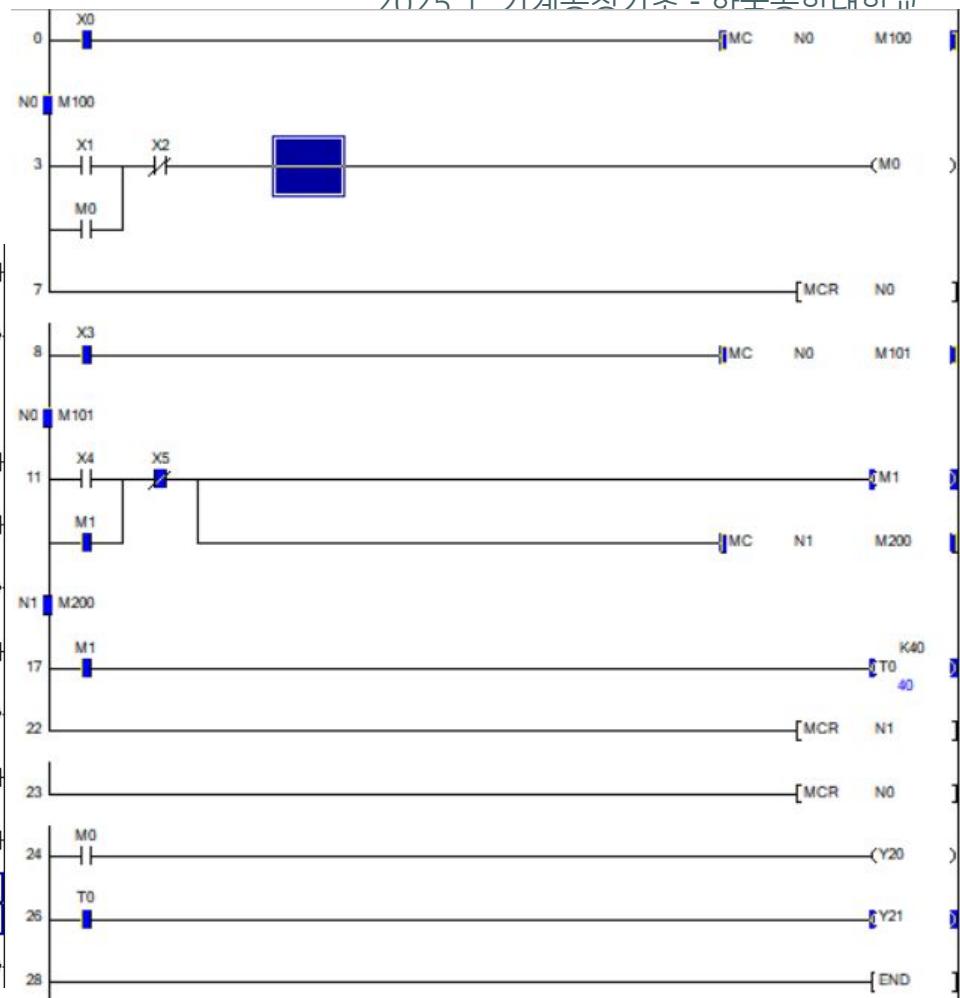
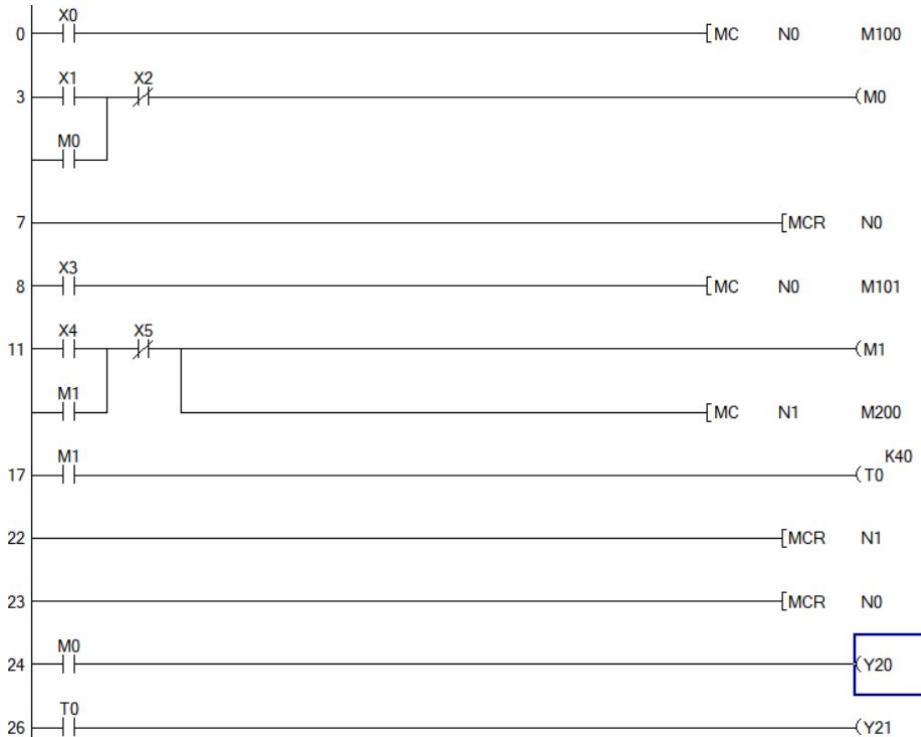
MCR 마스터 컨트롤 리셋 (종료)



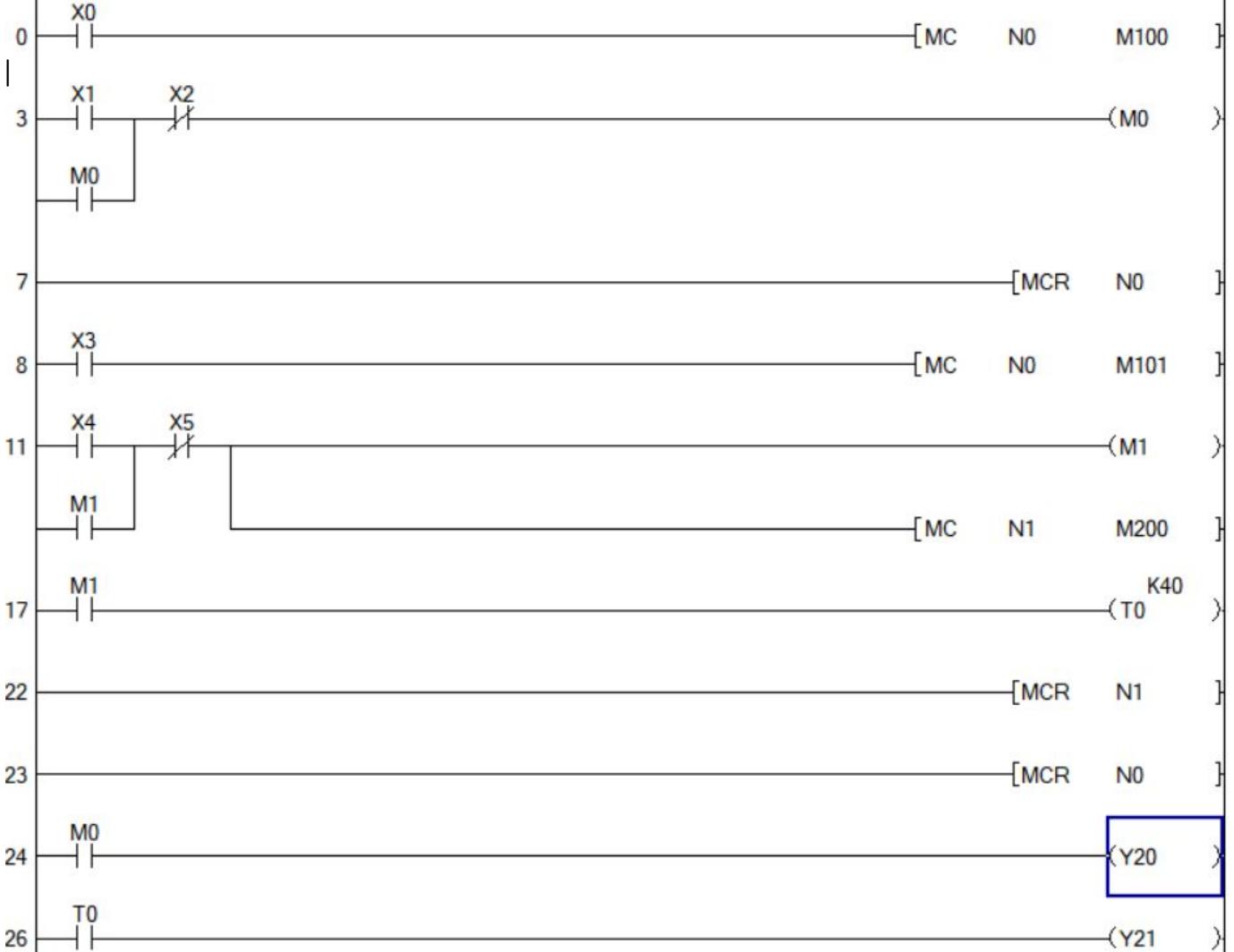
제3장 : 명령어 사용하기

MC 마스터 컨트롤 (시작)

MCR 마스터 컨트롤 리셋 (종료)



제3장 : 명령어 사용하기

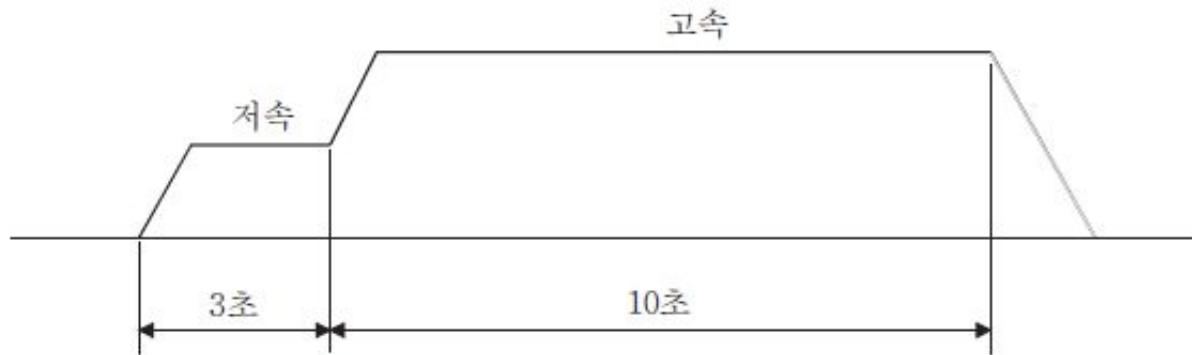


제3장 : 명령어 사용하기

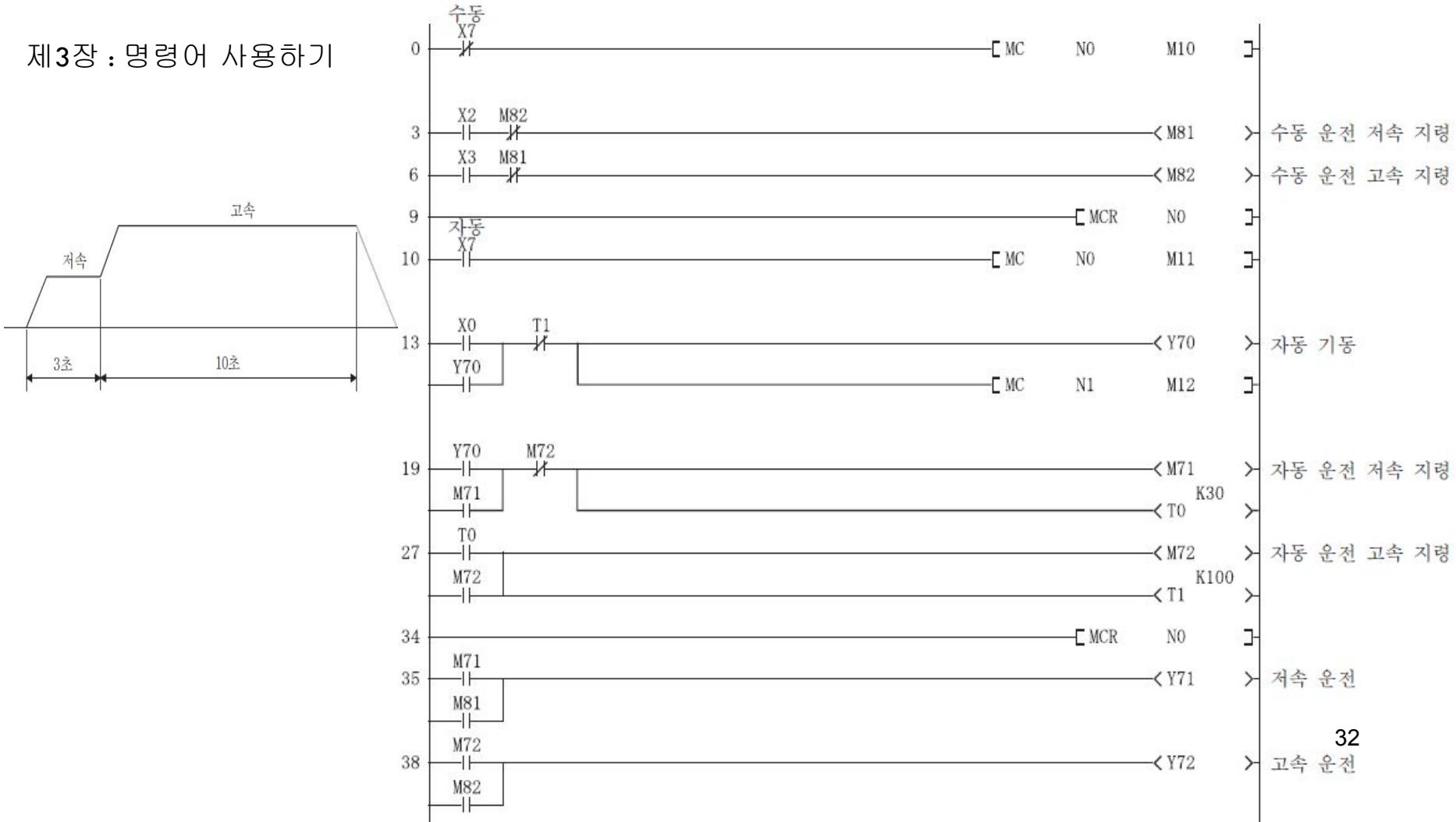
래더 예

다음의 래더는 MC, MCR 명령을 사용하여 수동 운전, 자동 운전을 전환할 수 있는 프로그램입니다.

- X7을 OFF 하여 수동 운전을 선택하였을 때는
 - X2를 ON 하면 저속 운전이 됩니다.
 - X3을 ON 하면 고속 운전이 됩니다.
- X7을 ON 하여 자동 운전을 선택한 경우, X0을 ON 하면 다음과 같이 기동 후 3초간 저속 운전을 한 후에 10초간 고속 운전을 하고 나서 정지합니다.

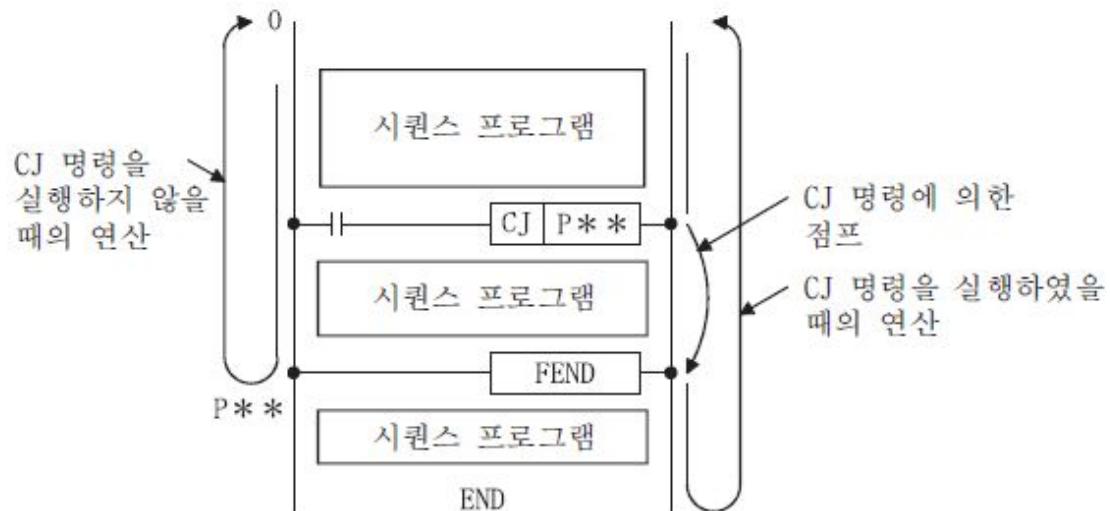


제3장 : 명령어 사용하기

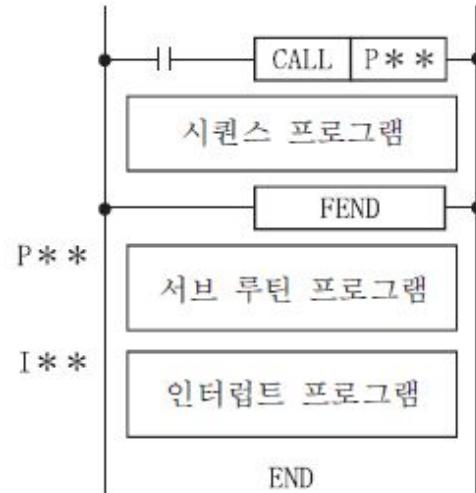


FEND 에프엔드

- FEND 명령은 다음과 같을 때 END 명령에 상당하는 명령으로 이용합니다.
 - 시퀀스 프로그램을 래더 블록 마다 연산을 종료시키고 싶을 때.
 - 예를 들어, CJ, SCJ 명령과 함께 이용합니다.
 - 서브 루틴 프로그램(CALL, RET 명령) 사용 시.
 - 인터럽트 프로그램 사용 시.
- FEND 명령을 실행하면, PLC는 타이머, 카운터의 현재값 처리 및 자기 진단 체크 등을 실행한 후에 0스텝부터 다시 연산을 합니다.



(a) CJ 명령에 의한 래더 블록별 연산의 경우



(b) 서브 루틴 프로그램, 인터럽트 프로그램이 있는 경우

제3장 : 명령어 사용하기

FEND 에프엔드



(1) X0이 OFF 되어 있을 때

- ① 0~FEND까지 연산
- ② X2를 ON/OFF 하면 Y20은 ON/OFF 됨
- ③ X5가 ON/OFF 되어도 Y21는 변경되지 않음

(2) X3이 ON 되어 있을 때

- ① CJ 명령에 의해 P10의 포인터로 점프
- ② X4가 ON/OFF 되어도 Y70은 변경
- ③ X5를 ON/OFF 하면 Y72는 ON/OFF됨

CJ

(컨디셔널 점프)

즉시 실행 조건 점프)

SCJ

(에스컨디셔널 점프)

1스캔 후 실행 조건 점프)

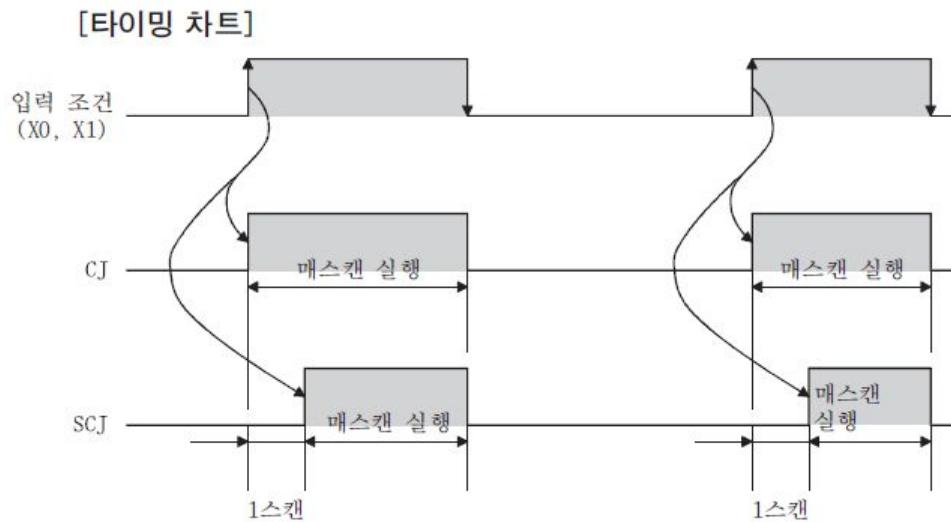
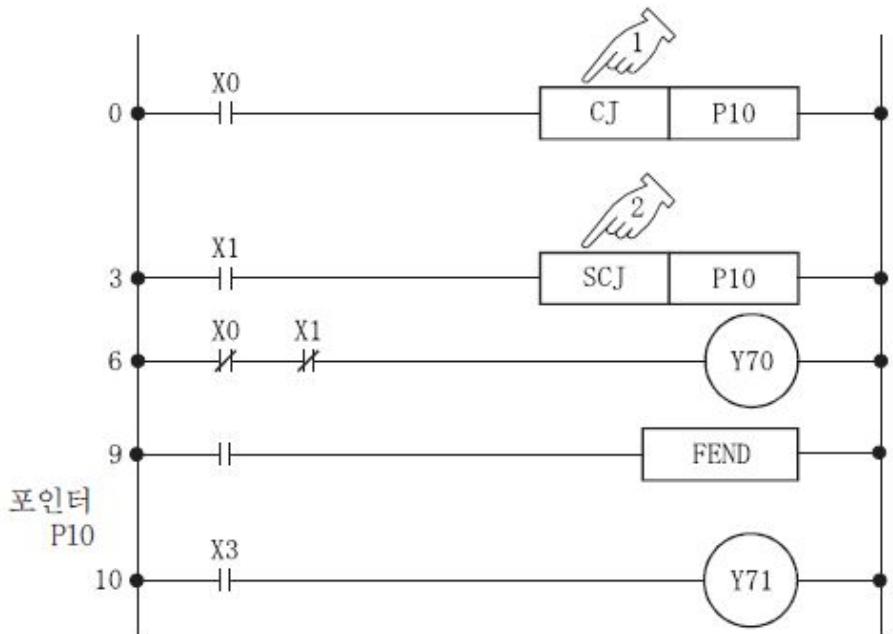
» CJ / SCJ



- CJ (즉시 실행 조건 점프)
- 입력 ON 시, 지정한 포인터로 이동하여 프로그램 실행
- CJ 명령으로 인해 건너 뛴 래더는 CJ 명령을 실행하기 전 상태를 유지



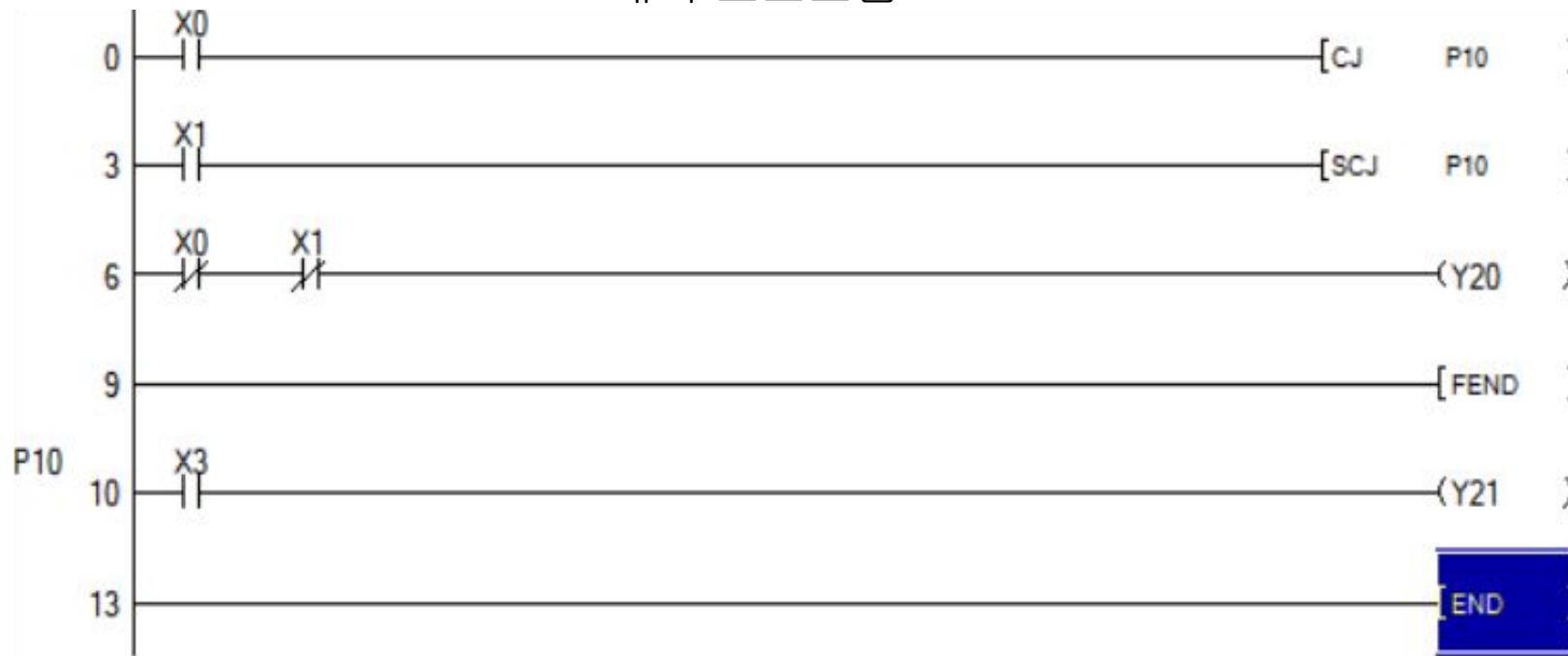
- SCJ (1스캔 후 실행 조건 점프)
- SCJ 명령의 기본 동작은 CJ와 동일
- 입력 조건 ON시, 처음 1스캔 실행 후, 지정한 포인터로 이동하여 프로그램 실행



CJ (컨디셔널 점프 즉시 실행 조건 점프)

SCJ (에스컨디셔널 점프 1스캔 후 실행 조건 점프)

래더 프로그램



제3장 : 명령어 사용하기

X0 실행으로 CJ 실행하여 10번 스텝으로 점프



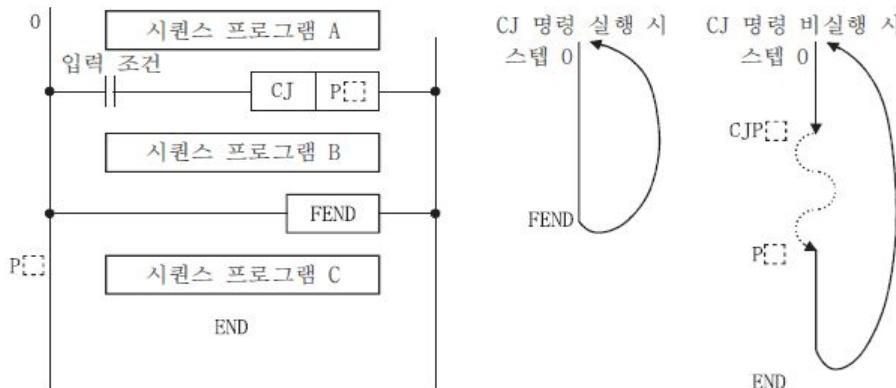
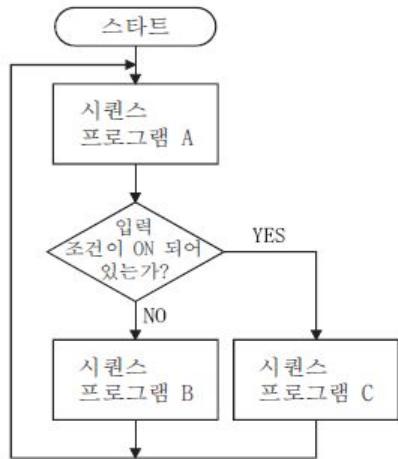
제3장 : 명령어 사용하기

X0 미실행, X3를 실행해도 Y21 미동작



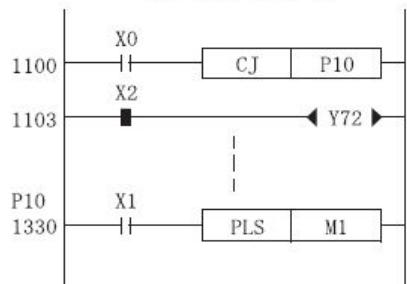
제3장 : 명령어 사용하기

- CJ, SCJ 명령 모두 포인터 번호는 P0~4095를 사용할 수 있습니다.
 - CJ, SCJ 명령에서 프로그램 블록별로 처리하고 싶을 때는 다음과 같이 FEND 명령을 사용하십시오. (FEND에 대해서는 이후의 항에서 설명합니다.)

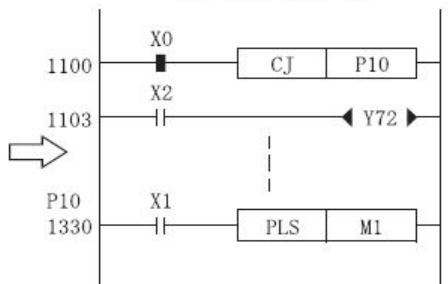


- CJ 명령으로 접포한 래더는 CJ 명령 실행 전의 상태를 유지합니다.

(CJ 명령 실행 전)



(CJ 명령 실행 중)

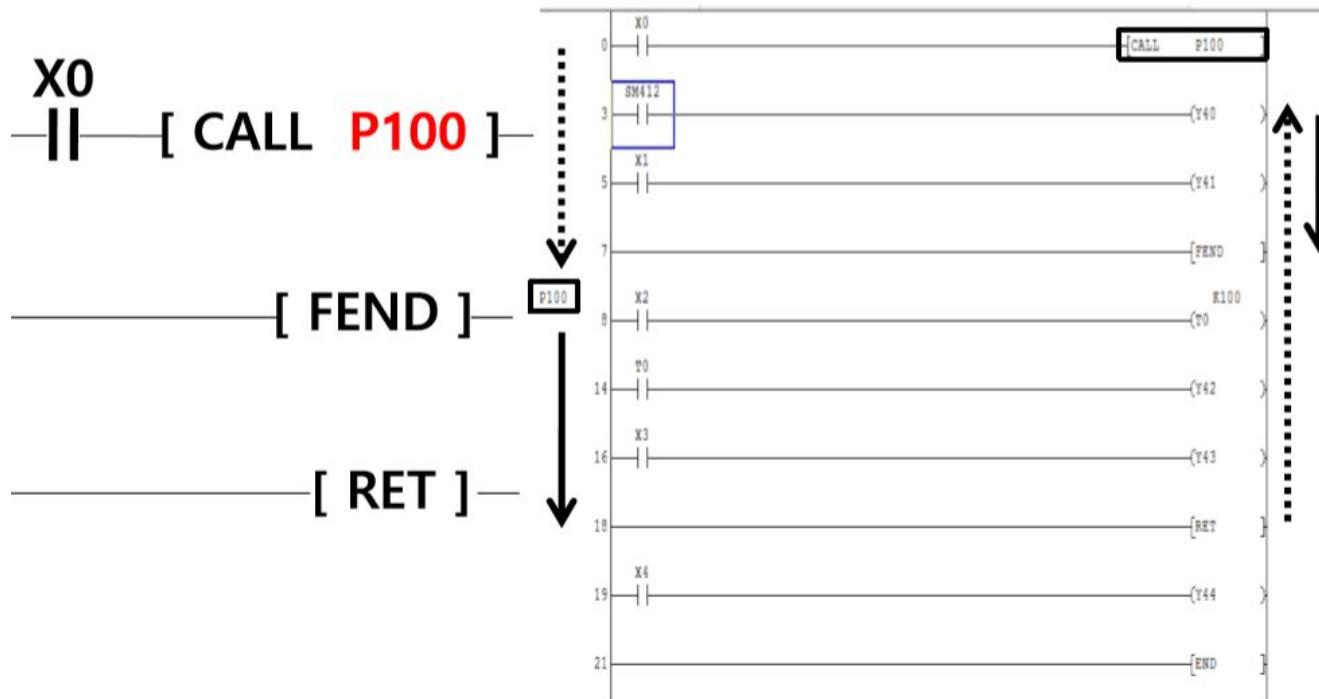


X0이 ON 되어 있으므로 이 부분의 래더는
각 명령 모두 실행되지 않습니다.
따라서 X2를 OFF 해도 Y72는 ON 상태를
유지합니다.

제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

서브 루틴 프로그램의 실행

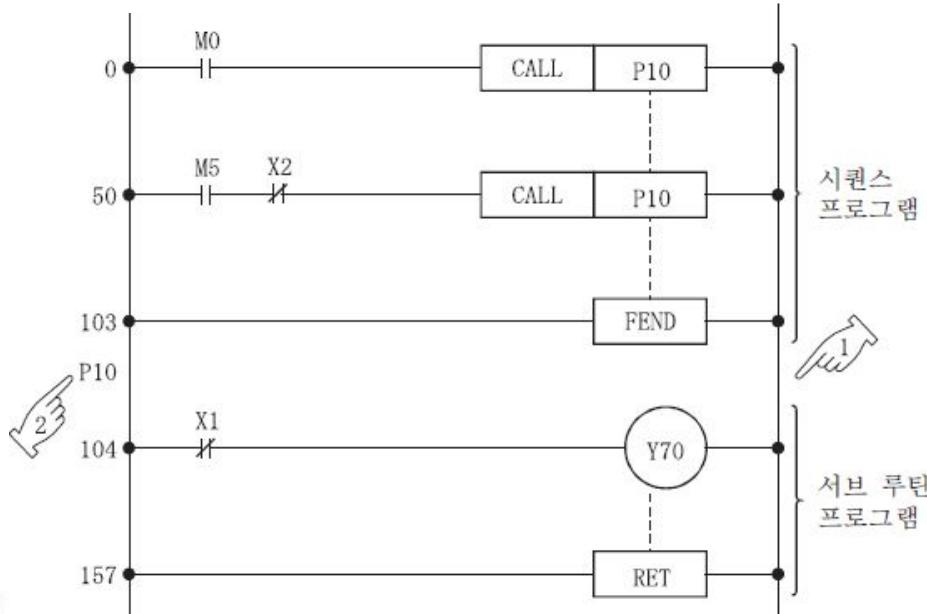


*CALL, RET : 서브 루틴 프로그램

제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

서브 루틴 프로그램의 실행



- 상기 프로그램의 형식이 서브 루틴 프로그램을 실행하도록 하는 CALL, RET 명령의 기본 사용 방법입니다.

이 형식을 지키지 않으면 에러가 되어 PLC는 정지합니다.

- 서브 루틴 프로그램이란 하나의 프로그램 내에 몇번이고 같은 내용을 계속해서 실행하고자 하는 경우에 사용하는 프로그램입니다.

포인터 P□에서 시작하고 RET 명령에서 종료합니다.

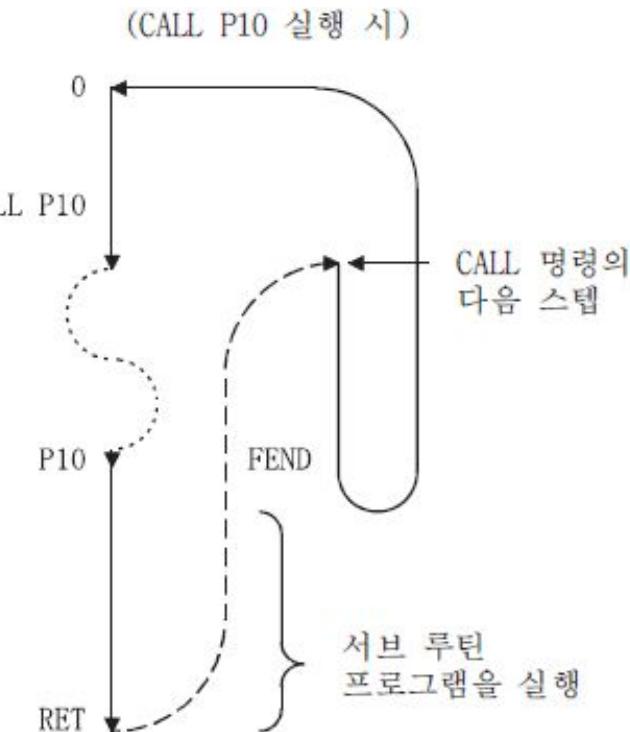
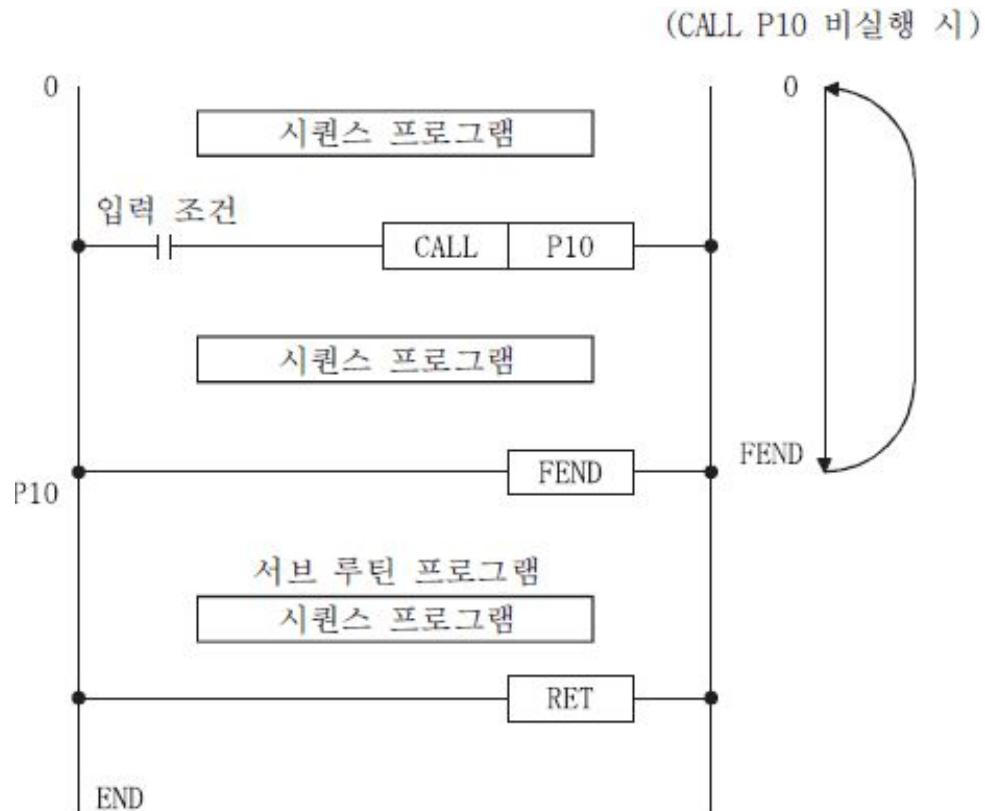
- 포인터 P의 번호는 0~4095입니다.(CJ, SCJ 명령에서 사용하는 포인터 번호와 공용입니다)

- 서브 루틴 프로그램의 실행을 그림으로 나타내면 다음과 같습니다.

제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

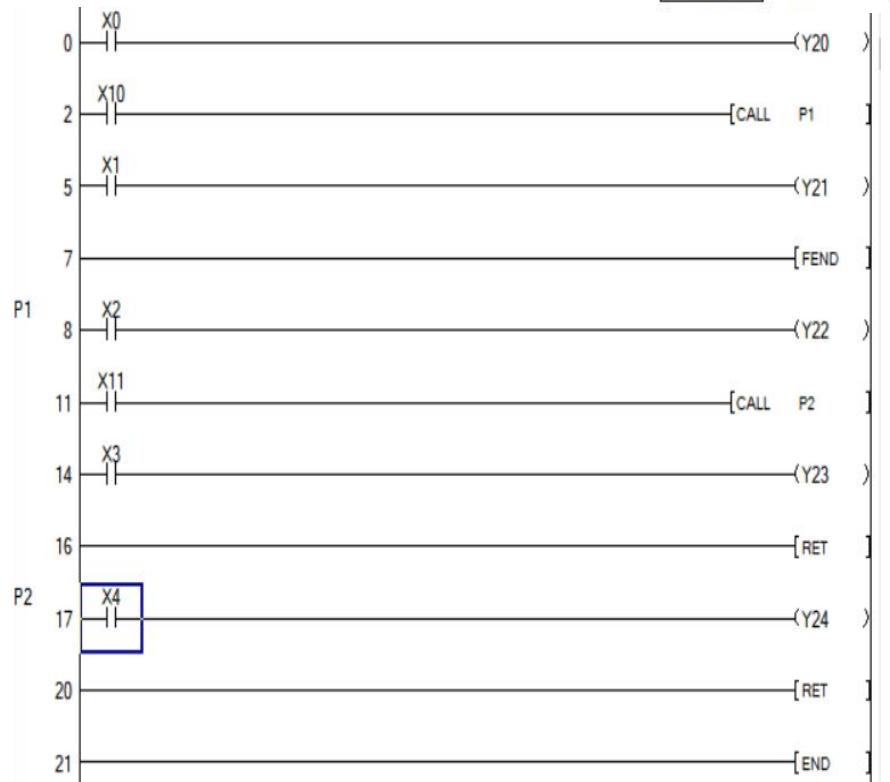
서브 루틴 프로그램의 실행



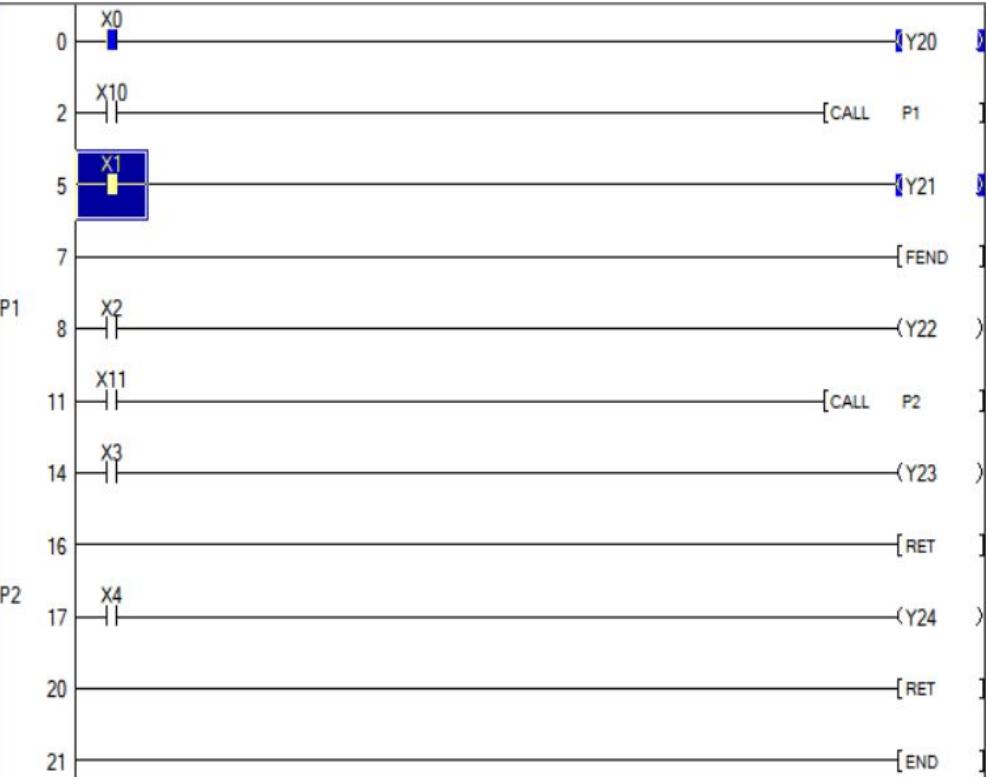
제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

서브 루틴 프로그램의 실행



프로그램

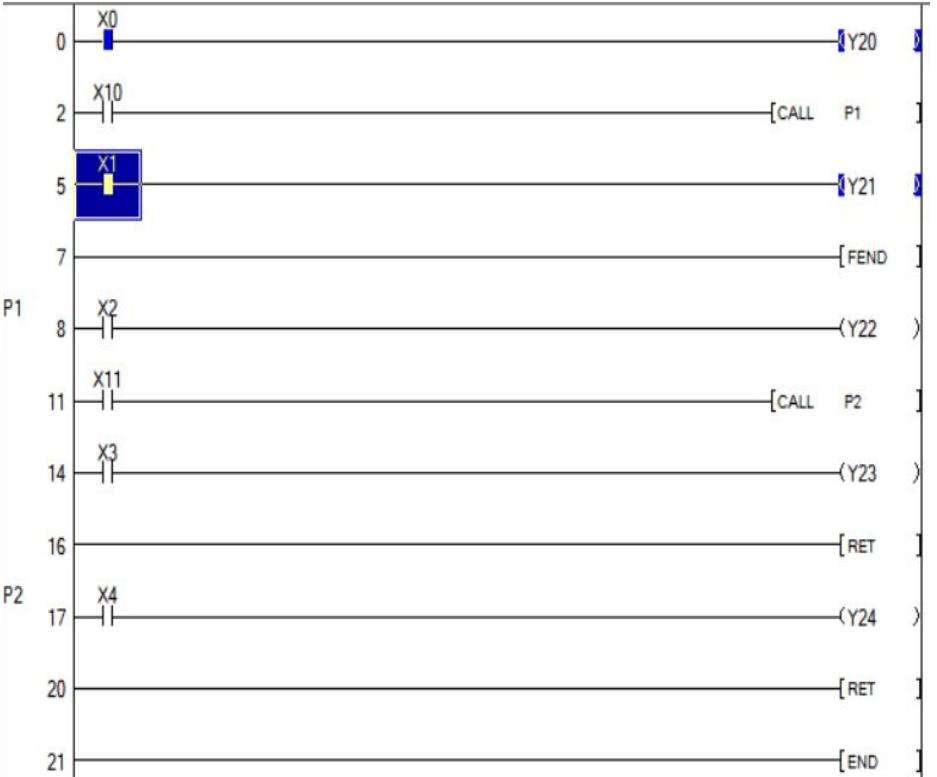


CALL 미실행

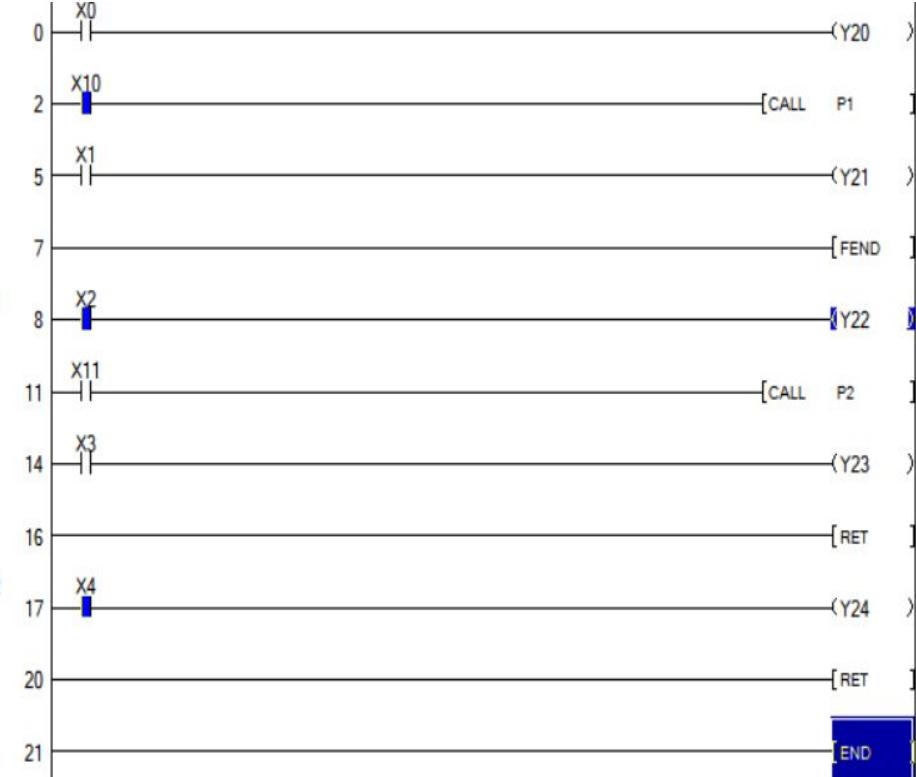
제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

서브 루틴 프로그램의 실행



CALL 미실행

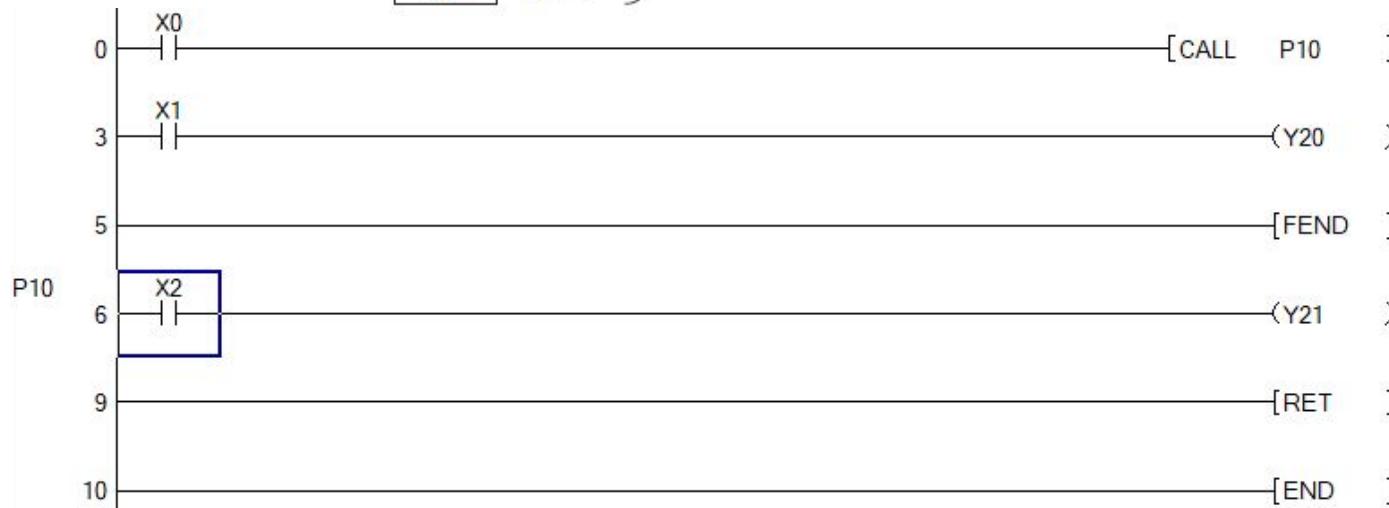


CALL 실행

제3장 : 명령어 사용하기

CALL(P) 콜
RET 리턴

서브 루틴 프로그램의 실행



(1) X0가 OFF 되어 있을 때

- ① 0~FEND까지 연산합니다.
- ② X0을 ON/OFF 하면 Y20은 ON/OFF 됩니다.
- ③ X1가 ON/OFF 되어도 Y21은 변경되지 않습니다.

(2) X0가 ON 되어 있을 때

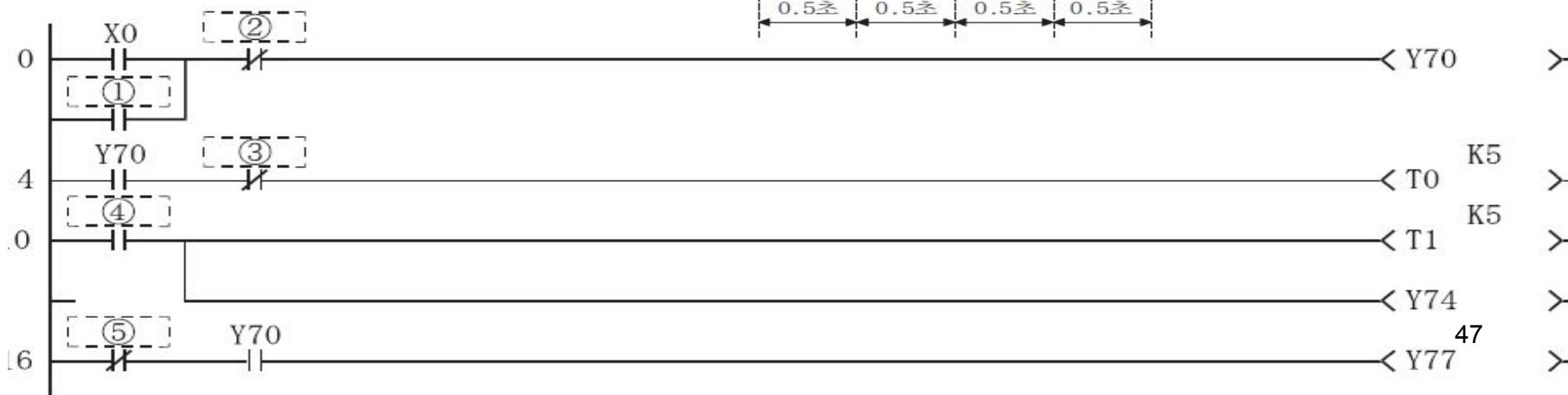
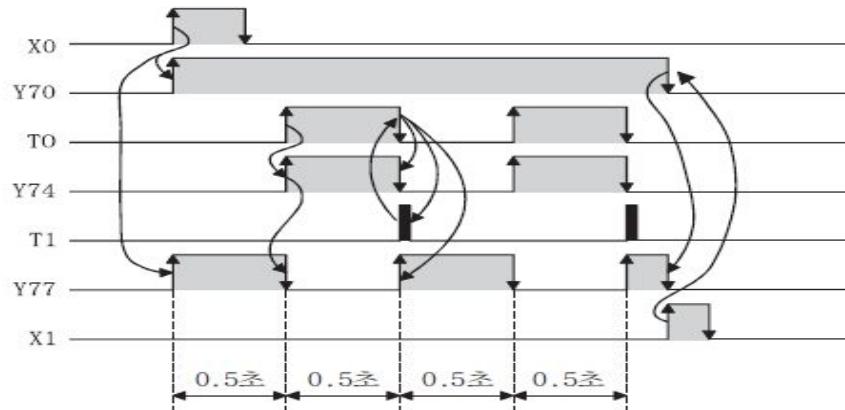
- ① P10의 서브 루틴 실행 후 스텝 3에서 FEND까지 연산합니다.
- ② X1을 ON/OFF 하면 Y20은 ON/OFF 됩니다.

제3장 : 명령어 사용하기

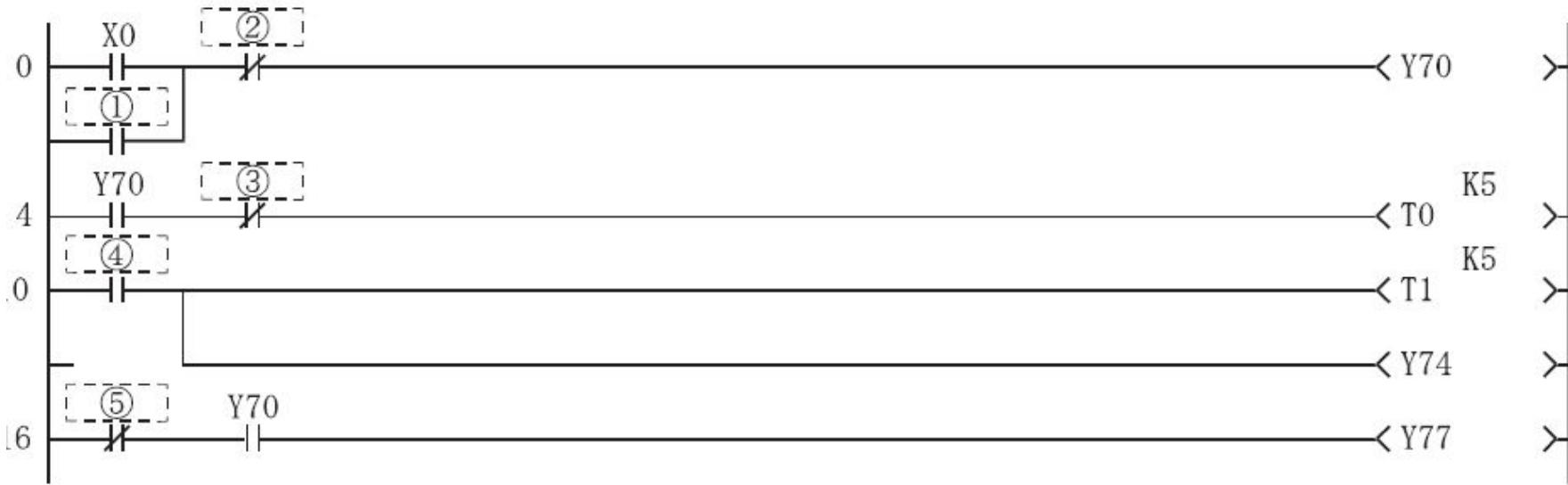
LD~NOP

X0이 ON 되면 Y70이 자기 유지되고, 0.5초마다 Y74와 Y77이 교대로 점멸합니다.
X1이 ON 되면 Y70이 OFF 되고, Y74, Y77의 점멸도 정지하는 프로그램입니다.

[타이밍 차트]



제3장 : 명령어 사용하기



①	Y70
②	X1
③	T1
④	T0
⑤	Y74

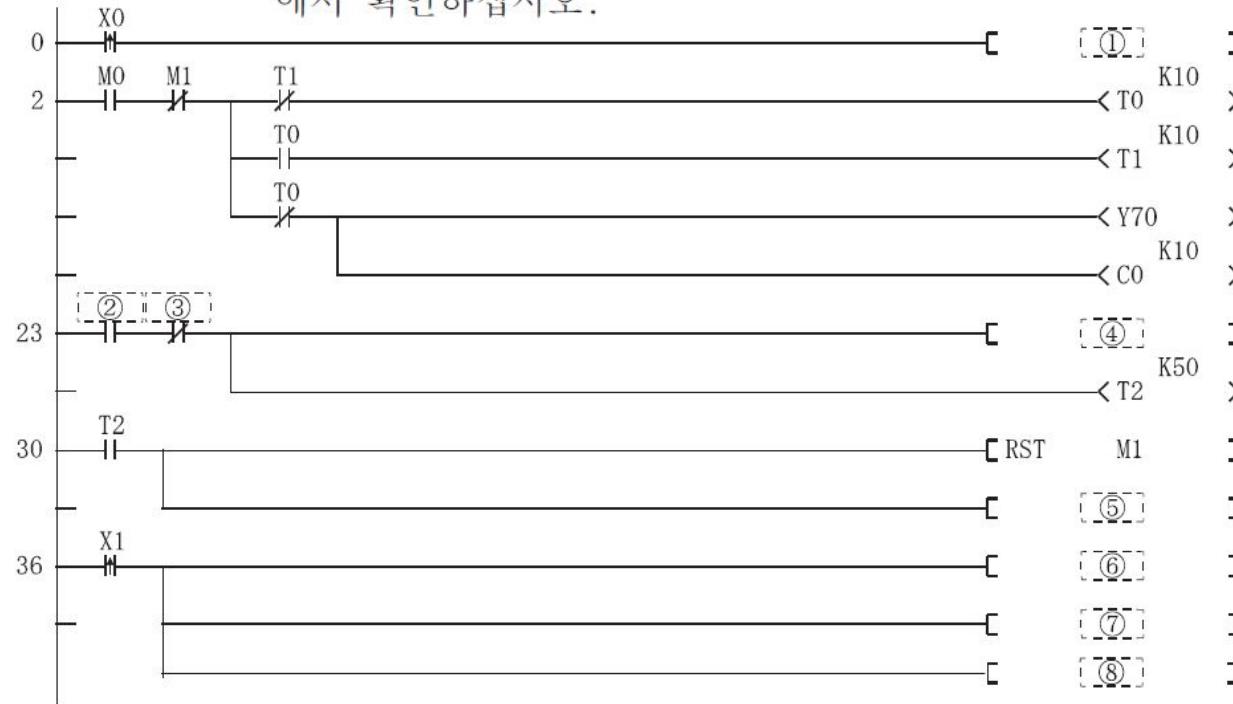
제3장 : 명령어 사용하기

SET, RST

X0을 ON 하면, Y70이 1초 간격으로 점멸을 시작하고, Y70이 10회 점멸하면 5초 간 점멸을 정지한 후에 다시 점멸을 시작합니다.

그리고 X1을 ON 하면, Y70의 점멸을 정지할 수 있는 프로그램입니다.

다음 프로그램의 [] 를 완성 후 GX Works2에서 프로그램을 작성하여 실습기에서 확인하십시오.



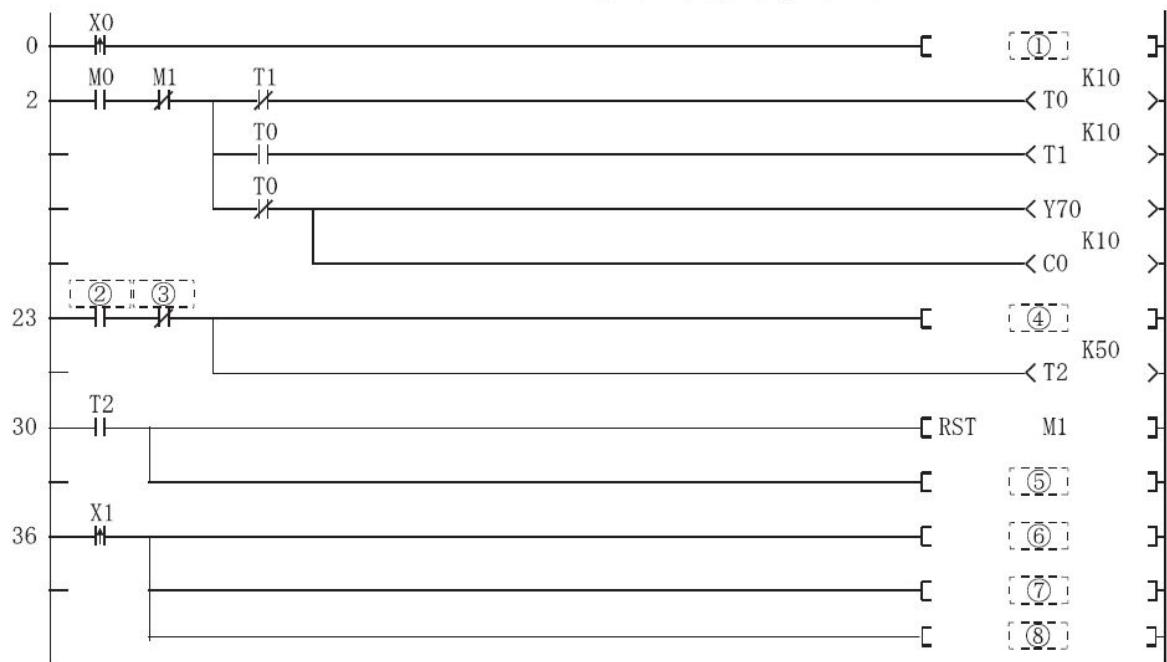
제3장 : 명령어 사용하기

SET, RST

X0을 ON 하면, Y70이 1초 간격으로 점멸을 시작하고, Y70이 10회 점멸하면 5초간 점멸을 정지한 후에 다시 점멸을 시작합니다.

그리고 X1을 ON 하면, Y70의 점멸을 정지할 수 있는 프로그램입니다.

다음 프로그램의 [] 를 완성 후 GX Works2에서 프로그램을 작성하여 실습기에서 확인하십시오.



①	SET M0
②	C0
③	Y70
④	SET M1
⑤	RST C0
⑥	RST M0
⑦	RST C0
⑧	RST M1

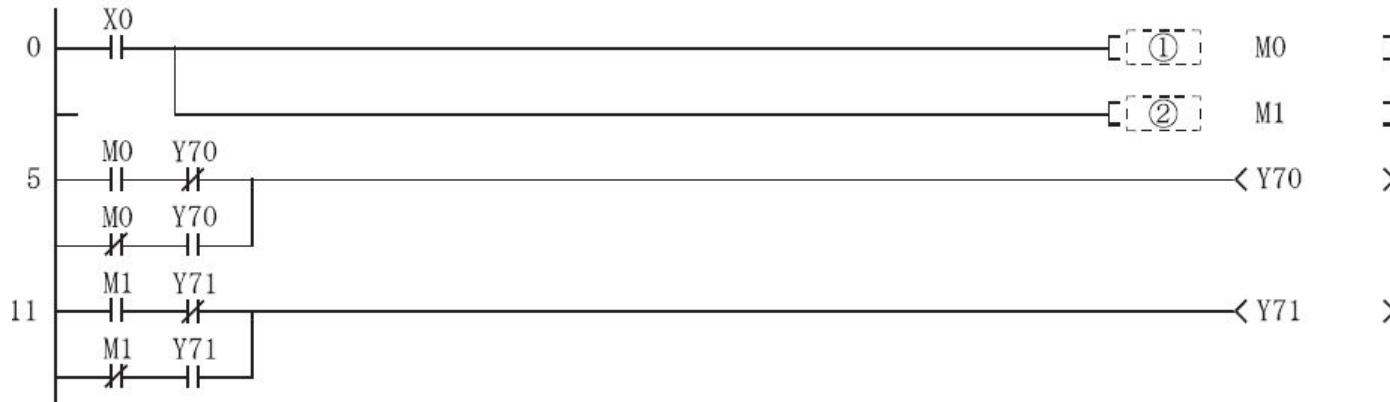
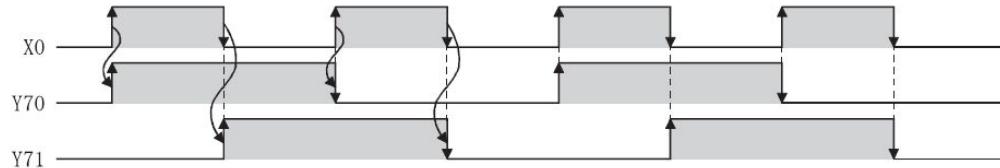
제3장 : 명령어 사용하기

PLS, PLF

X0의 상승펄스 시 Y70이 ON→OFF→ON→OFF.....를 반복하고, X0의 하강펄스 시 Y71이 ON→OFF→ON→OFF.....를 반복합니다.

연습 문제 3

[타이밍 차트]



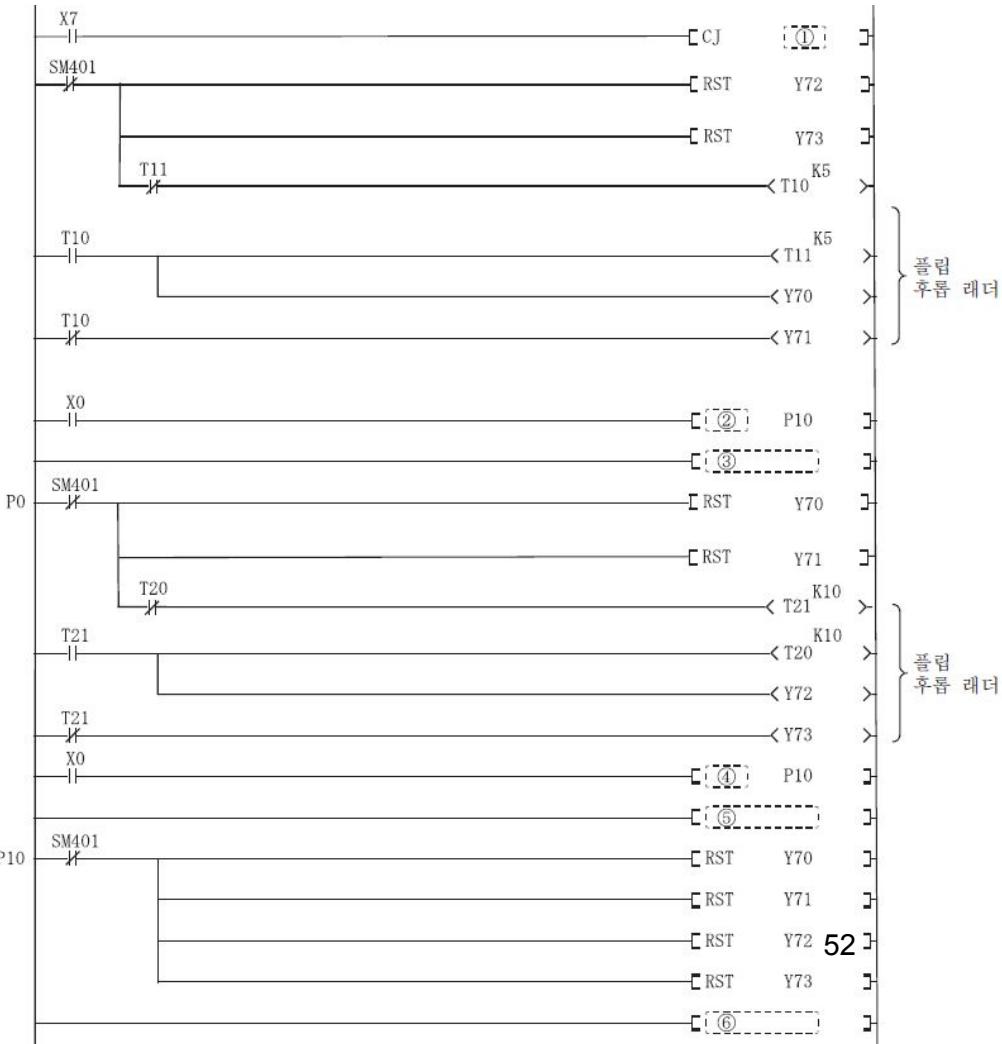
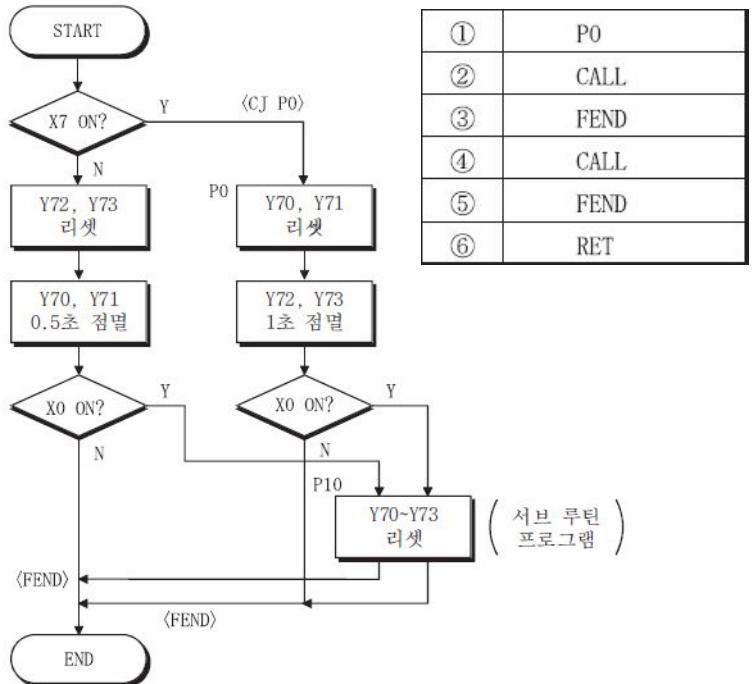
①	PLS
②	PLF

제3장 : 명령어 사용하기

연습 문제 4

CJ, CALL, RET, FEND

X7이 OFF 되어 있을 때는 Y70과 Y71을 0.5초씩 점멸하고, X7이 ON 되어 있을 때는 Y72와 Y73을 1.0초씩 점멸하는 래더입니다. 또한, X0을 ON 하면 현재 점멸하고 있는 Y70~Y73을 리셋하는 래더입니다.



제3장 : 명령어 사용하기

10진수/2진수/16진수 이해하기

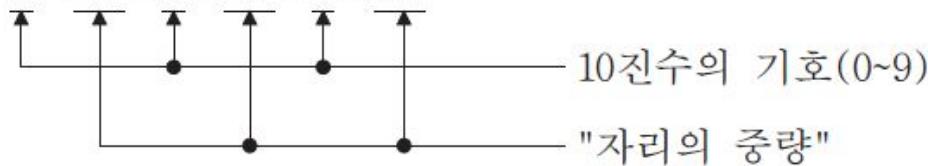
(10진수) Decimal

- 10진수란 "0~9의 10종류의 기호를 사용하여 순서와 크기(양)를 나타내는 수"라고 할 수 있습니다.
0~9까지 수가 진행되면, "10"으로 자리를 옮겨 이후를 진행합니다.
- 예를 들어, 10진수 "153"을 자리와 "자리의 중량"이라고 하는 관점에서 살펴보겠습니다.

$$153 = 100 + 50 + 3$$

$$= 1 \times 100 + 5 \times 10 + 3 \times 1$$

$$= 1 \times 10^2 + 5 \times 10^1 + 3 \times 10^0$$



"자리의 중량"은 다음과 같이 확인할 수 있습니다.

$n \dots \dots$ 자리 번호(0, 1, 2\dots)

10 \dots \dots

- MELSEC-Q의 PLC에서는 10진수를 표현할 때 "K"를 붙입니다.

제3장 : 명령어 사용하기

10진수/2진수/16진수 이해하기

2진수 Binary...BIN

- 2진수란 "0과 1의 2종류의 기호를 사용하여 순서와 크기를 나타내는 수"라고 할 수 있습니다. 0, 1까지 수가 진행되면, "10"으로 자리를 옮겨 이후를 진행 합니다. 0, 1의 1자리는 비트라고 합니다.

2진수	10진수
0	0
1	1
10	2
11	3
100	4

- 예를 들어, 다음 2진수는 10진수로 몇이 될지 생각해 보겠습니다.

"10011101"

10진수로 자리 번호와 자리의 중량으로 살펴 본 것처럼, 오른쪽부터 비트 번호와 비트의 중량을 붙여 보겠습니다.

7	6	5	4	3	2	1	0	←비트 번호
1	0	0	1	1	1	0	1	←2진수
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	←(비트 번호) ←("2진수") } 비트의 중량
128	64	32	16	8	4	2	1	

10진수의 경우처럼 각 비트의 코드와 중량의 곱을 더해 보겠습니다.

$$\begin{aligned}
 &= 1 \times 128 + 0 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\
 &= 128 + 16 + 8 + 4 + 1 \\
 &= 157
 \end{aligned}$$

즉, 2진수는 "코드가 1인 비트의 중량을 더한 것"이 10진수가 되는 것입니다.

제3장 : 명령어 사용하기

10진수/2진수/ 16진수 이해하기

16진수 Hexadecimal

- 16진수나 10진수, 2진수와 같은 방법으로 살펴보면, "0~9, A~F의 16종류의 기호를 사용하여 순서와 크기를 나타내는 수"라고 할 수 있습니다. 그리고, 0, 1, 2…D, E, F로 수가 진행되면, "10"으로 자리를 옮겨 이후를 진행합니다.

10진수	16진수	2진수
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
1 9 1 0 1	4 A 9 D	0 1 0 0 1 0 1 0 1 0 0 1 1 1 0 1

3	2	1	0	←자리 번호
4	A	9	D	←16진수

$$\begin{aligned}
 &= (4) \times \underline{16^3} + (\text{A}) \times \underline{16^2} + (9) \times \underline{16^1} + (\text{D}) \times \underline{16^0} \\
 &= 4 \times 4096 + 10 \times 256 + 9 \times 16 + 13 \times 1 \\
 &= 19101
 \end{aligned}$$

"자리의 중량"
n 자리 번호
16 16진수

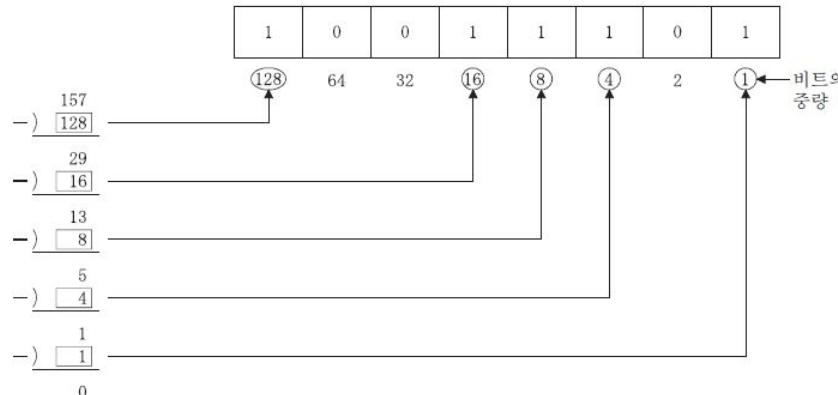
제3장 : 명령어 사용하기

10진수/2진수/ 16진수 이해하기

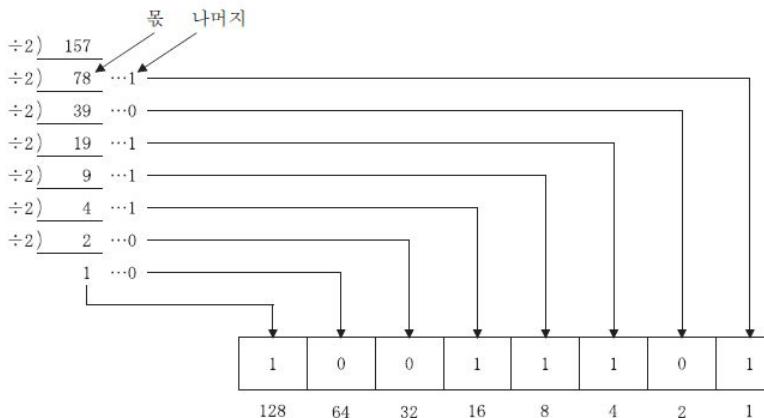
10진수에서 2진수로의 변환 방법

10진수의 "157"을 2진수로 변환하는 예

①



②



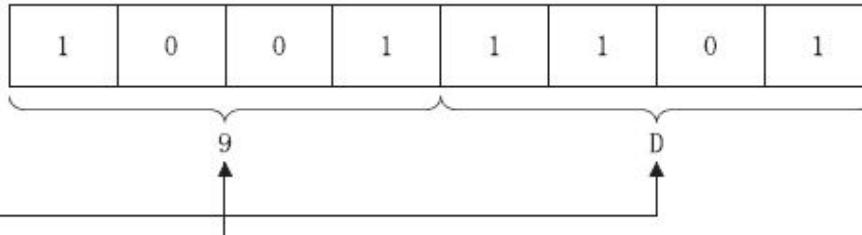
10진수/2진수/16진수 이해하기

- 10진수에서 16진수로의 변환 방법

10진수의 "157"을 16진수로 변환하는 예

1

$$\begin{array}{r} \div 16) \underline{157} \\ 9 \quad \cdots 13(D) \\ \boxed{} \qquad \boxed{} \end{array}$$



BCD 코드 이해하기

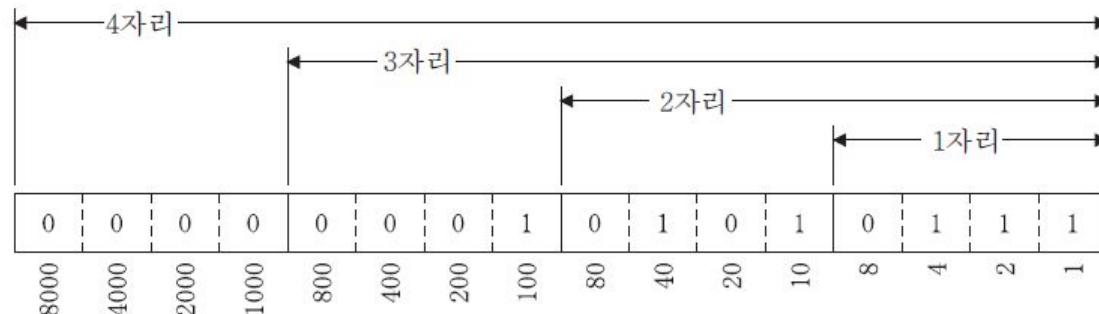
2진화 10진수 Binary Coded Decimal…BCD

- 2진화 10진수는 그 호칭에서 추정되듯이, "10진수의 각 자리의 숫자를 2진수로 표현한 것"이라고 할 수 있습니다.

예를 들어, 10진수의 157은

2	1	0	←자리 번호
1	5	7	←10진수
(100)	(10)	(1)	←자리의 중량
0001	0101	0111	←2진화 10진수
842①	8④2①	8④②①	←2진수의 비트의 중량

- 따라서 2진화 10진수는 10진의 0~9999(4자리의 최대값)를 16비트로 나타냅니다. 각 비트의 중량은 다음과 같습니다.



제3장 : 명령어 사용하기

BCD 코드 디지털 스위치 이해하기

- 2진화 10진수는 다음과 같은 곳에 사용되고 있습니다.

- ① 디지털 스위치의 출력 신호
- ② 7세그먼트 표시기(디지털 표시기) 신호

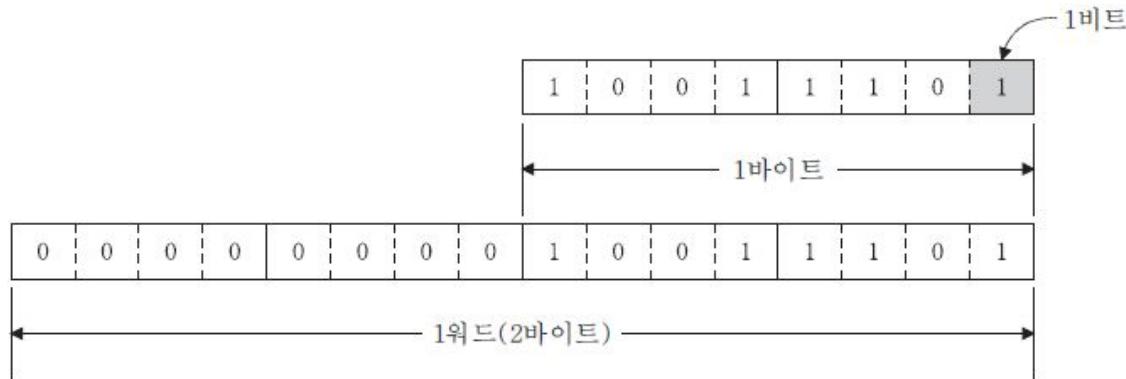
0	1	2	3	4	5	6	7	8	9
 1 (0) 2 (0) 4 (0) 8 (0) COM	 2 (1) 4 (0) 6 (1) 8 (0)	 3 (0) 5 (1) 7 (0) 9 (0)	 4 (1) 6 (0) 8 (1) 0 (0)	 5 (0) 7 (1) 9 (0) 1 (0)	 6 (1) 8 (0) 0 (1) 2 (0)	 7 (0) 9 (1) 1 (0) 3 (0)	 8 (1) 0 (0) 2 (1) 4 (0)	 9 (0) 1 (0) 3 (1) 5 (0)	 0 (1) 2 (0) 4 (1) 6 (0)

BCD 코드 디지털 스위치

제3장 : 명령어 사용하기

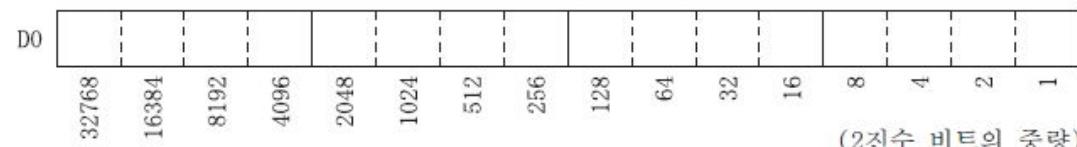
- 일반적으로 8비트를 1바이트, 16비트(2바이트)를 1워드라고 합니다.

- MELSEC-Q가 취급하는 수치



- MELSEC-Q의 각 워드 디바이스의 레지스터는 16비트로 구성되어 있습니다.

- 데이터 레지스터 D
 - 타이머 T의 현재값
 - 카운터 C의 현재값
 - 파일 레지스터 R
 - 링크 레지스터 W
- 등



- 16비트(1워드)로 취급할 수 있는 수치는 다음의 2종류가 있습니다.

- 0~65535
- 32768~0~+32767

제3장 : 명령어 사용하기

- MELSEC-Q가 취급하는 수치

- MELSEC-Q는 ②의 범위 내에서 사용할 수 있습니다.

음수는 양수(1~+32767)에 대하여 2의 보수를 사용하고 있습니다.

- 2의 보수란 2진수의 각 비트에 대해 1을 0으로, 0을 1로 반전한 값에 대해 최하위 비트에 1을 더한 것을 말합니다.

예

1에 대한 2의 보수(-1)를 구하는 방법

1...	0	0	0	0	0	0	0	0	0	0	0	0	0	1
------	---	---	---	---	---	---	---	---	---	---	---	---	---	---



1과 0을 반전시킨다

1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

+)	0	0	0	0	0	0	0	0	0	0	0	0	0	1
-----	---	---	---	---	---	---	---	---	---	---	---	---	---	---

최하위 비트에
1을 더한다

-1...	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-------	---	---	---	---	---	---	---	---	---	---	---	---	---	---

최상위 비트는 음수일 때 반드시 1이 됩니다. 즉 부호 비트입니다.

11111111	11111111	-1	FFFF
11111111	11111110	-2	FFFE
10000000	00000000	-32768	8000

제4장 : 응용명령 사용하기

MOV / MOVP

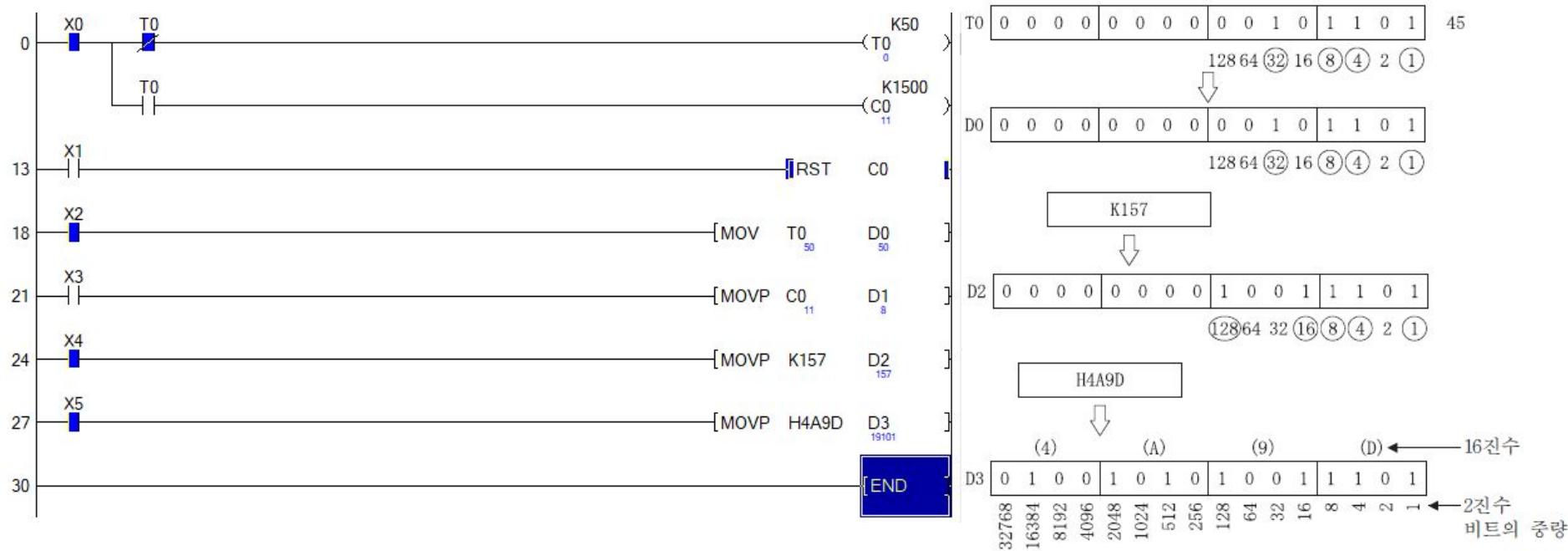
전송명령



- MOV (16 Bit 데이터 전송)
입력 조건이 ON 일 때,
데이터를 지정한 디바이스 혹은 데이터 레지스터에 전송한다
- MOV와 MOVP의 차이점
변화하는 데이터를 계속 전송 때는 MOV 명령을 사용
조건신호 ON하는 순간 한번만 데이터를 전송할 때 MOVP⁶² 사용

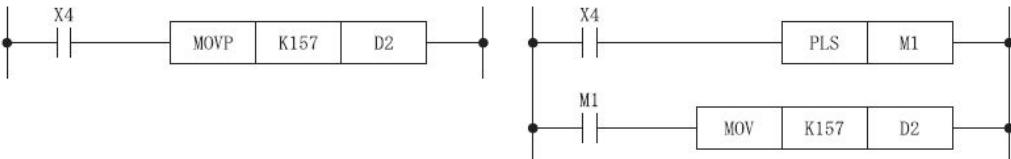
제4장 : 응용명령 사용하기

전송명령



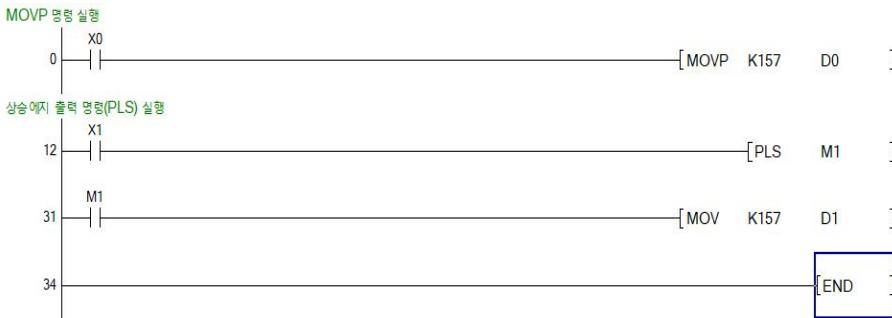
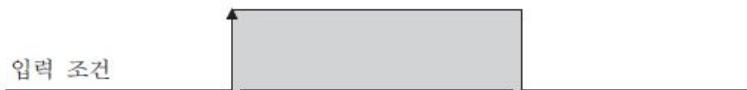
제4장 : 응용명령 사용하기

전송명령



MOV와 MOVP의 차이

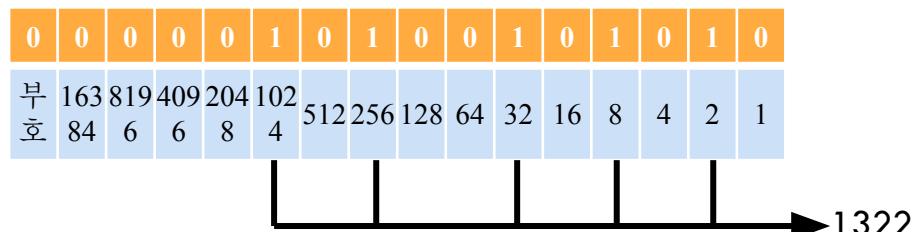
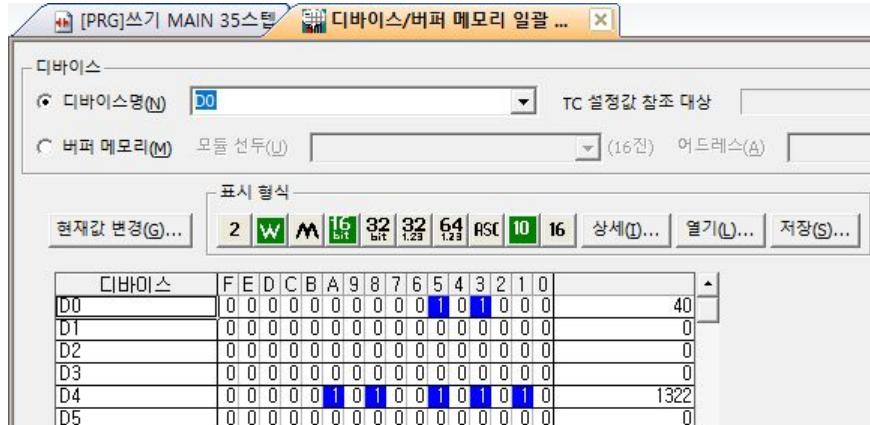
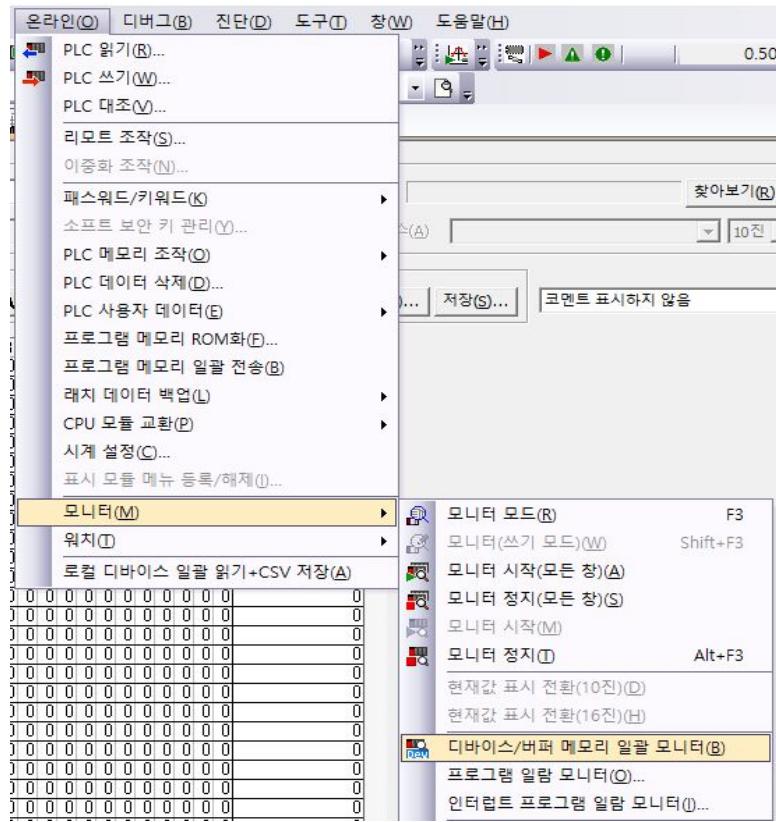
MOVP의 P는 펄스(pulse)의 앞글자



- 변경되는 데이터를 항상 전송할 때는 **MOV** 명령을 사용합니다.
이에 비해, 설정, 이상 시 데이터를 순간적으로 전송할 때는 **MOVP** 명령을 사용합니다.
- 다음 2개의 래더는 같은 기능을 합니다.

제4장 : 응용명령 사용하기

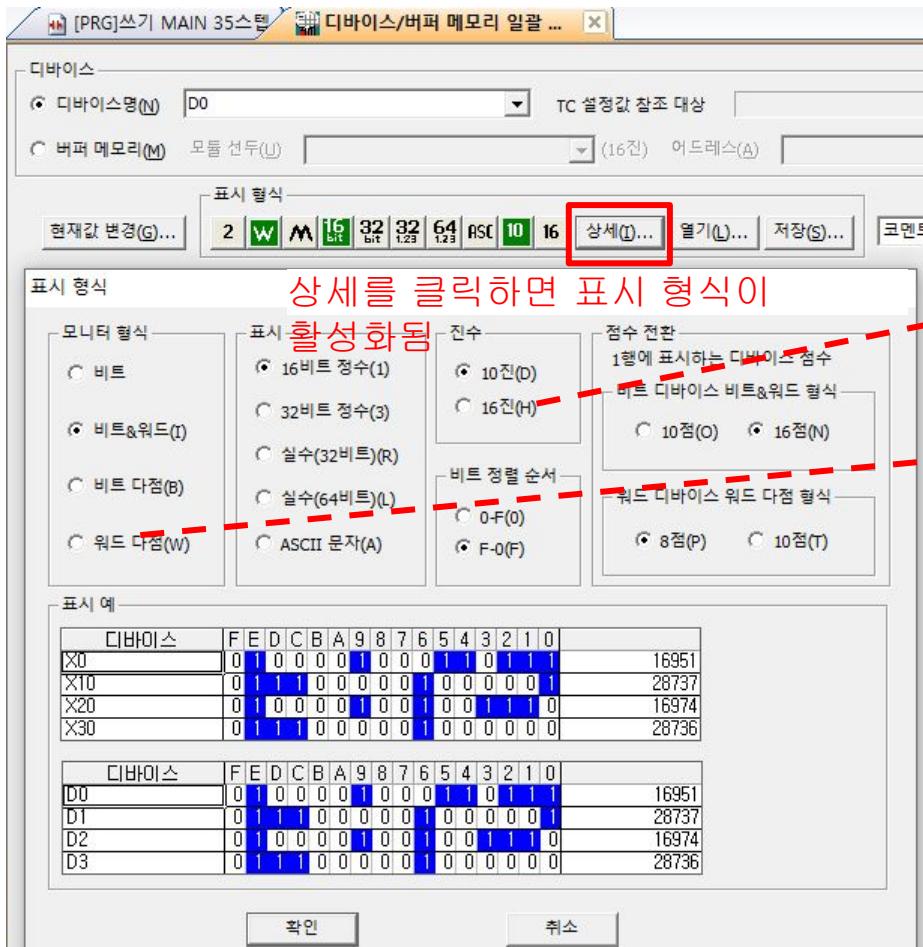
디바이스/버퍼 메모리 일괄 모니터



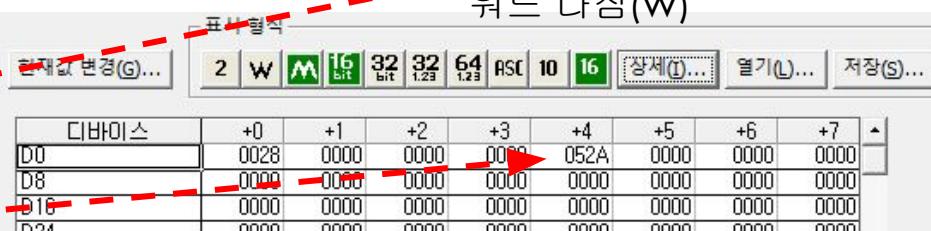
제4장 : 응용명령 사용하기

디바이스/버퍼 메모리 일괄 모니터

2025.1. 기계공정기초 - 한국공학대학교



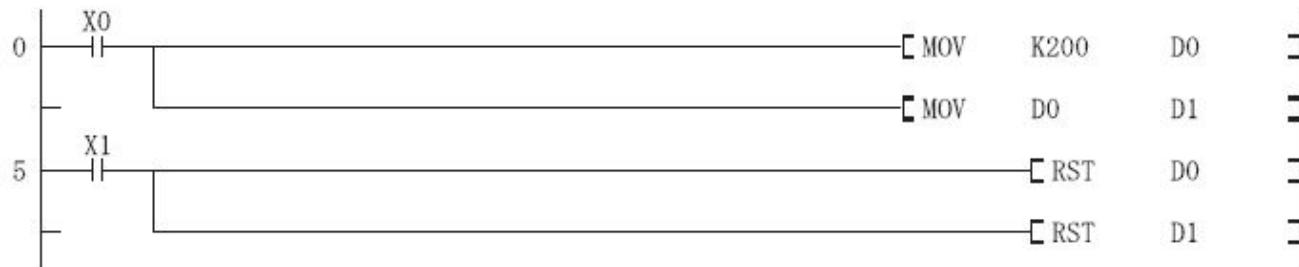
워드 다점(W)



제4장 : 응용명령 사용하기

래더 예

아래의 래더를 GX Works2에서 작성하여 실습기에 쓴 후 MOV 명령의 실행을 확인하십시오.



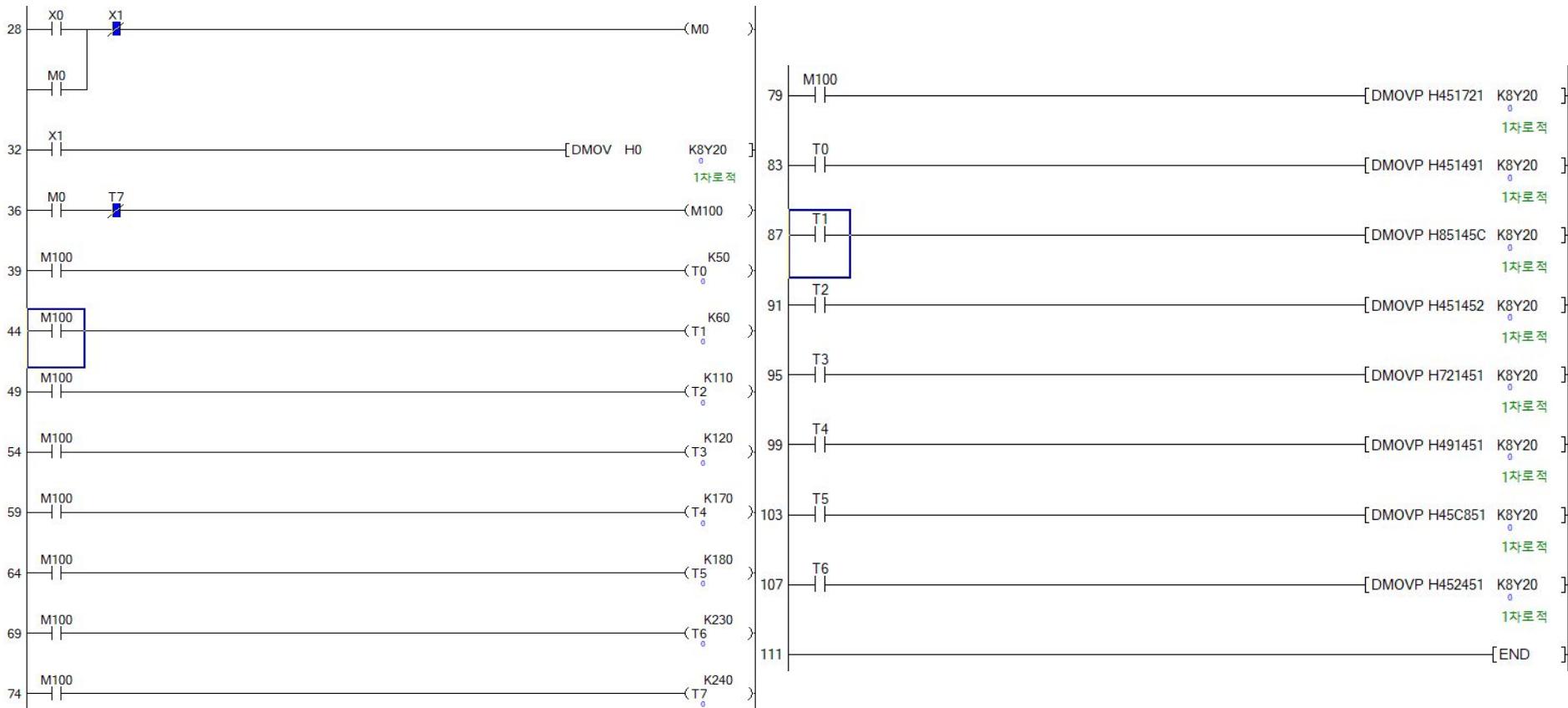
제4장 : 응용명령 사용하기

4거리 신호등 제어기

차로	구분	신호등색	5초	1초	5초	1초	5초	1초	5초	1초	겹점
1차로	자동차 신호등	적	●	●	●	●	●	●			Y20
		주								●	Y21
		녹							●		Y22
		좌회전							●		Y23
	보행자 신호등	적		●	●	●	●	●	●	●	Y24
		녹	●								Y25
2차로	자동차 신호등	적			●	●	●	●	●	●	Y26
		주		●							Y27
		녹	●								Y28
		좌회전	●								Y29
	보행자 신호등	적	●	●		●	●	●	●	●	Y2A
		녹			●						Y2B
3차로	자동차 신호등	적	●	●			●	●	●	●	Y2C
		주				●					Y2D
		녹			●						Y2E
		좌회전			●						Y2F
	보행자 신호등	적	●	●	●	●		●	●	●	Y30
		녹					●				Y31
4차로	자동차 신호등	적	●	●	●	●			●	●	Y32
		주						●			Y33
		녹					●				Y34
		좌회전					●				Y35
	보행자 신호등	적	●	●	●	●	●	●		●	Y36
		녹							●		Y37

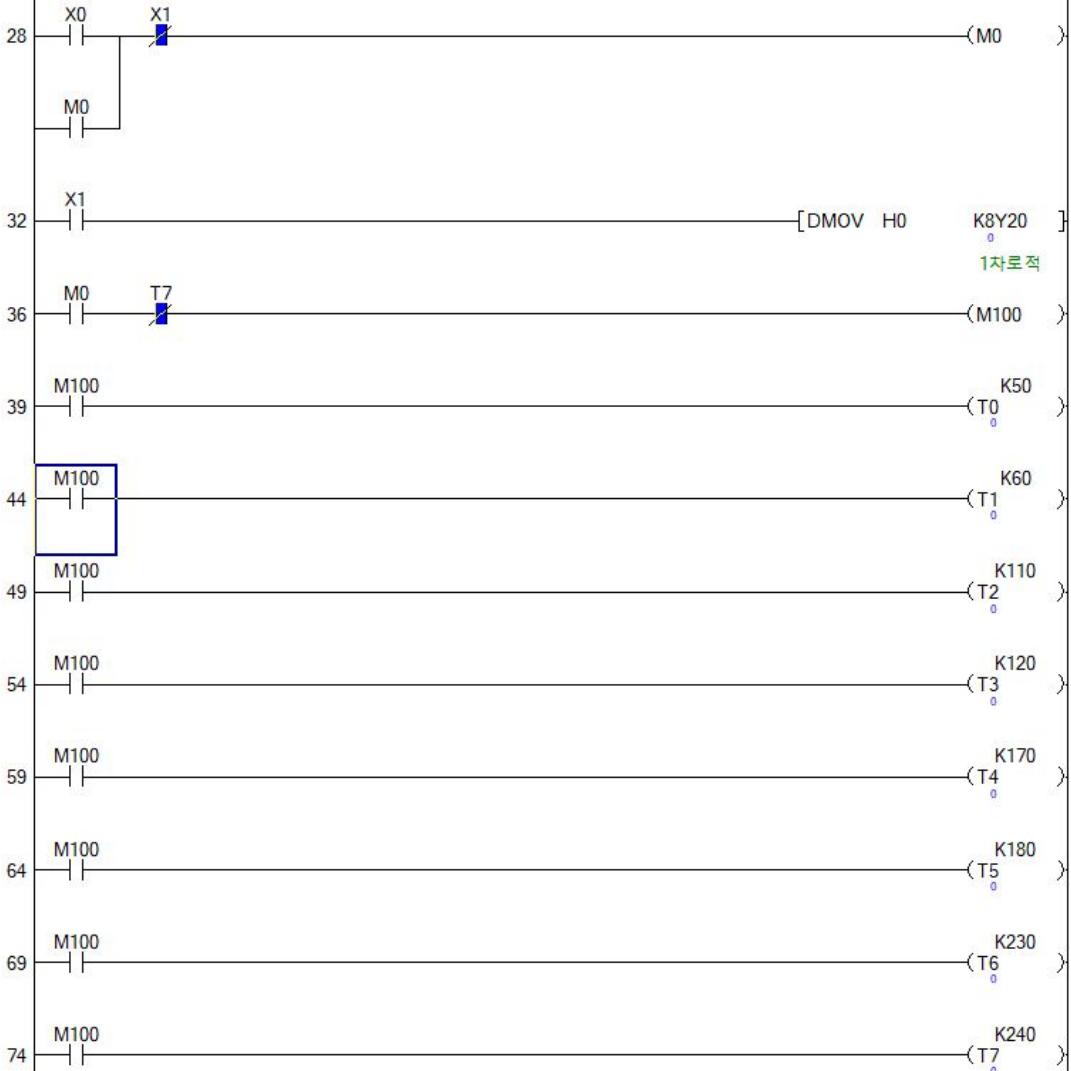
제4장 : 응용명령 사용하기

4거리 신호등 래더



제4장 : 응용명령 사용하기

4거리 신호등 래더



제4장 : 응용명령 사용하기

4거리 신호등 래더



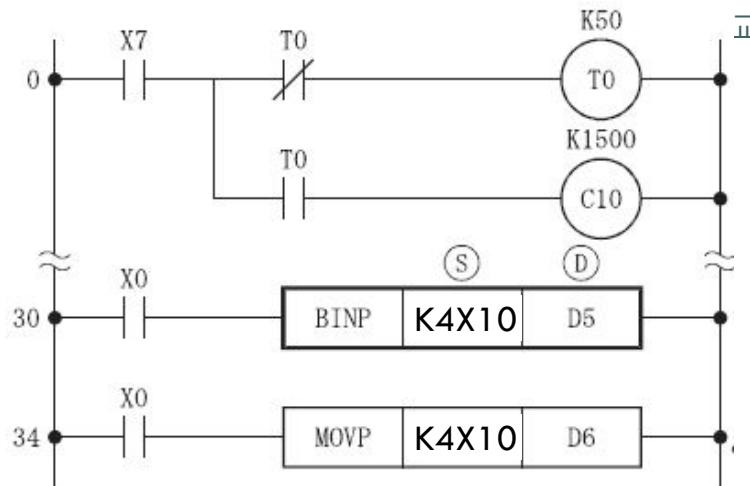
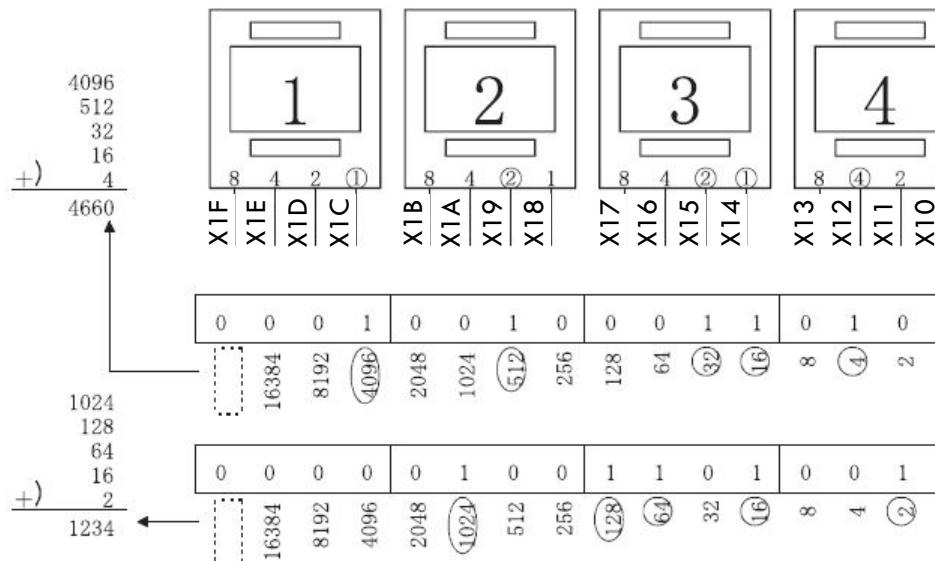
코드 변환 : BIN / BCD



- BIN(P)
BCD코드 (2진화 10진수) >> BIN코드 (2진수) c변환 명령
- BCD(P)
BIN(2진수) >> BCD코드 (2진화 10진수) 데이터 변환 명령
- 일반적인 디지털 스위치는 BCD코드를 사용
따라서 디지털 스위치의 데이터를 PLC로 전달할 때, BIN명령을 사용
- BCD명령에서는 데이터(BIN >> BCD의 데이터가 0~9,999일 때 가능
(9,999를 넘는 값을 표시할 때는 DBCD명령을 사용. 99,999,999 까지)

제4장 : 응용명령 사용하기

코드 변환 : BIN / BCD



디지털 스위치

K4X20

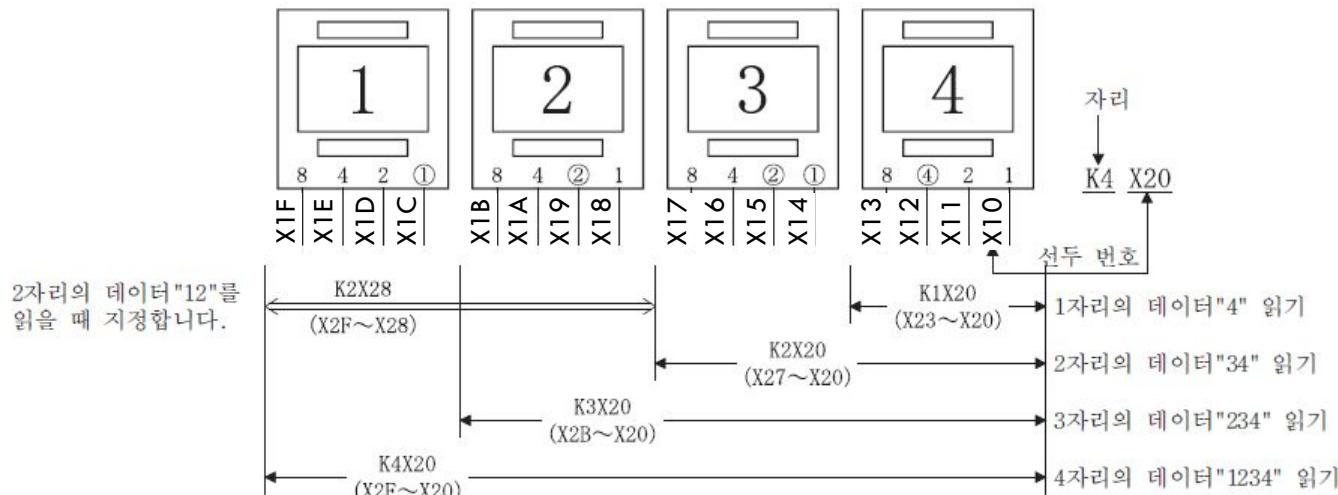
D6
BCD 그대로
입력한 경우

D5
BIN으로 변환하여
입력한 경우

코드 변환 : BIN / BCD

K4X20이란

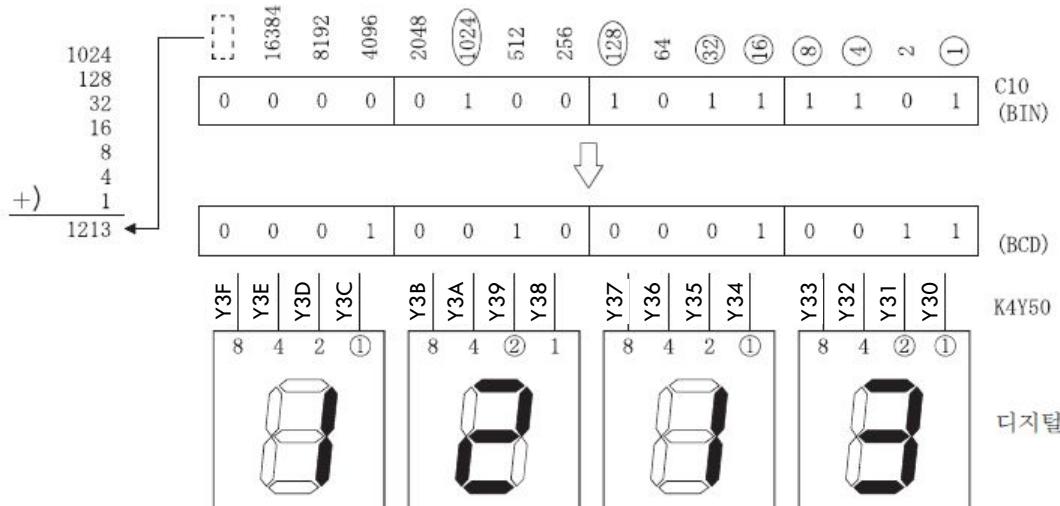
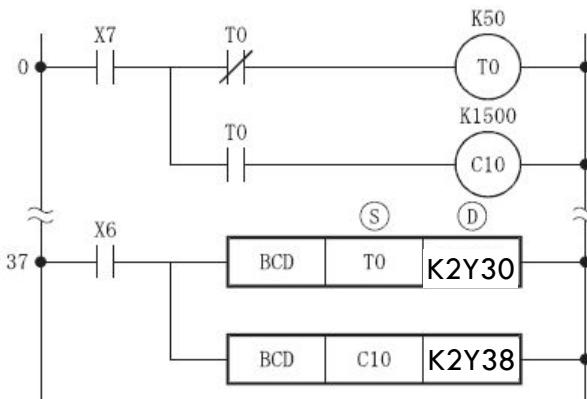
- 워드 디바이스 D(데이터 레지스터), T(타이머 현재값), C(카운터 현재값) 등은 1개가 16비트(1워드)로 구성되며, 원칙적으로 1개의 디바이스 간에 데이터의 전송을 합니다.
- 비트 디바이스(X, Y, M 등)도 16점이 모이면, 워드 디바이스와 같은 크기의 데이터를 취급할 수 있습니다. 다만 디바이스 번호가 연속된 16점이어야 합니다.
- 비트 디바이스의 경우, 4점 단위로 데이터를 취급할 수 있습니다.



4점 단위로 연속된 디바이스라면, 선두 디바이스는 어디라도 관계없습니다.

제4장 : 응용명령 사용하기

코드 변환 : BIN / BCD

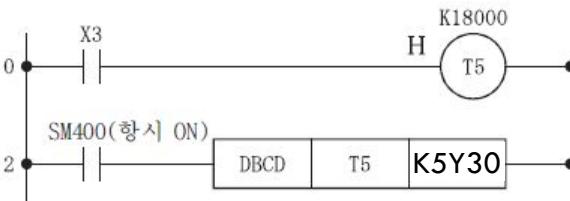
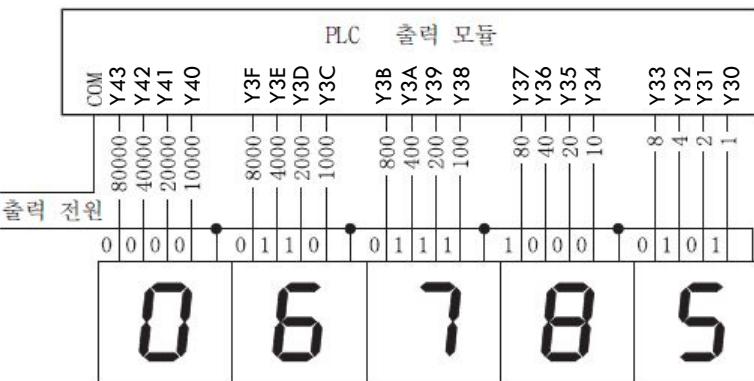


제4장 : 응용명령 사용하기

코드 변환 : BIN / BCD

BCD 명령을 사용하여 표시할 수 있는 범위

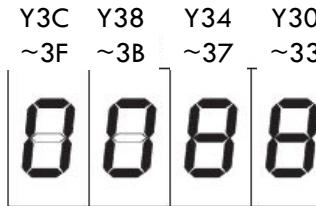
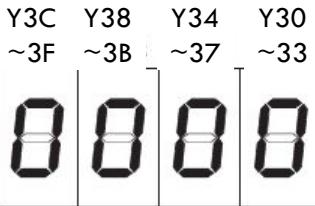
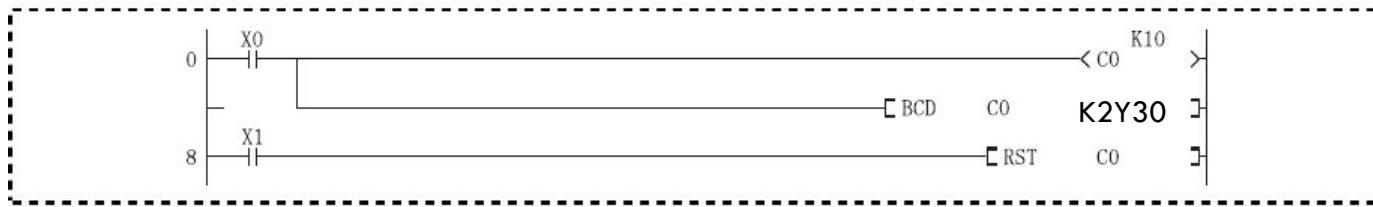
- BCD 명령은 표시하고자 하는 데이터(BIN→BCD 코드로 변환하고자 하는 데이터)가 0~9999일 때 사용이 가능합니다. 이외의 값에서는 에러가 됩니다.
(에러 코드 4100 : OPERATION ERROR)
- 예를 들어, 설정값이 9,999를 초과하는 타이머의 현재값을 표시하고 싶을 때는 DBCD 명령을 사용합니다. 이 때는 99,999,999까지의 8자리의 숫자를 취급할 수 있습니다.



코드 변환 : BIN / BCD

래더 예

아래의 래더를 GX Works2에서 작성하여 실습기에 쓴 후 BCD 명령의 실행을 확인하십시오.



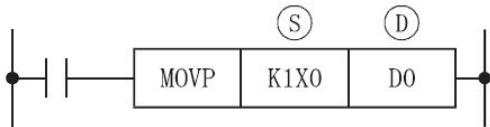
0~10
C0의 값을 표시

BCD 디지털 표시기

제4장 : 응용명령 사용하기

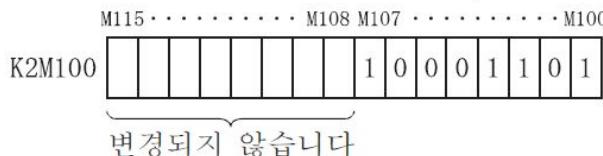
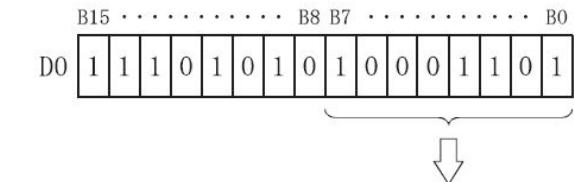
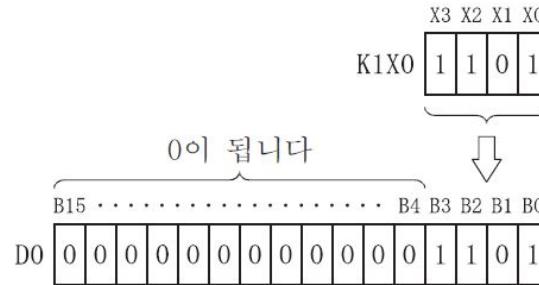
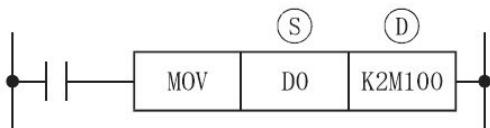
비트를 워드로, 워드를 비트로 전송

데스티네이션(D) 데이터가 워드 디바이스일 때



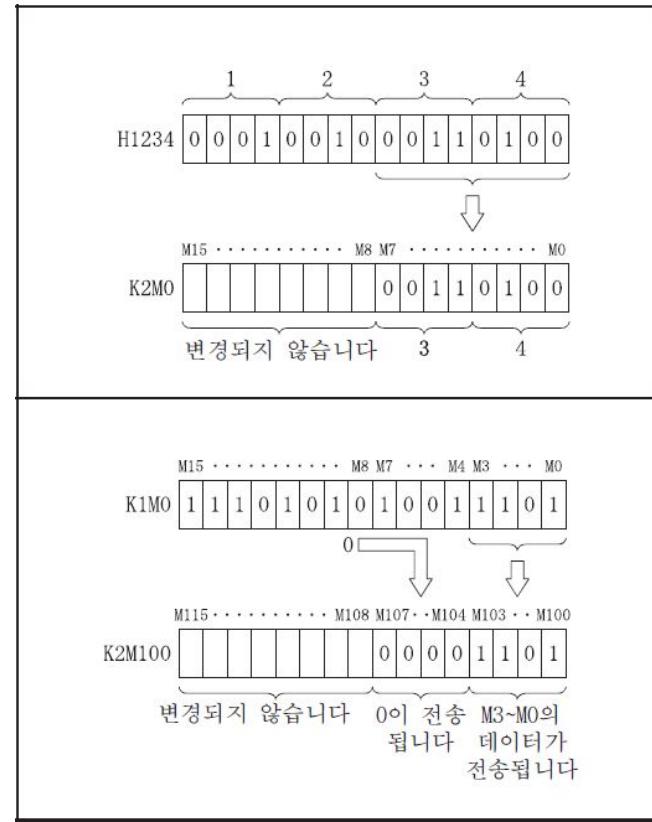
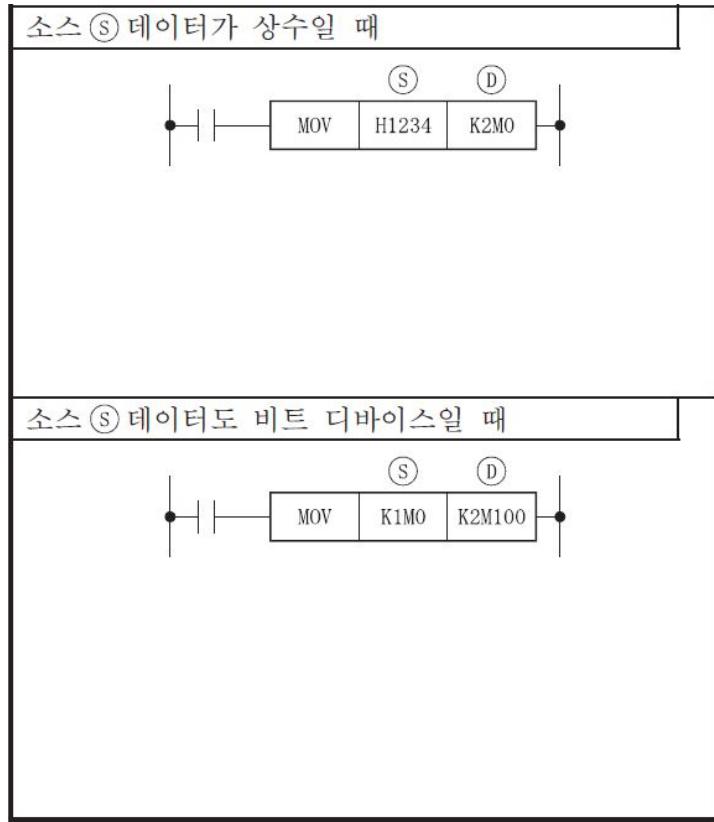
- 소스 : 전송 소스 디바이스
- 데스티네이션 : 전송 상대 디바이스

소스(S) 데이터가 워드 디바이스일 때



제4장 : 응용명령 사용하기

데이터를 비트로, 비트를 비트로 전송



제4장 : 응용명령 사용하기

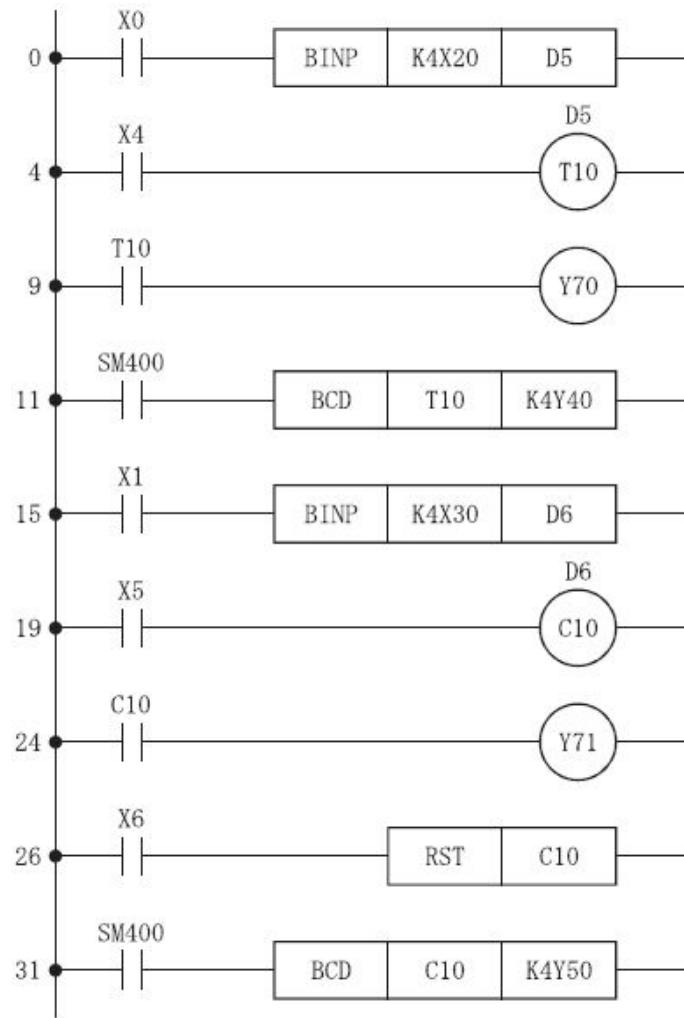
코드 변환 : BIN / BCD



제4장 : 응용명령 사용하기

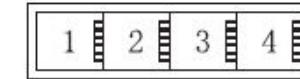
코드 변환 : BIN / BCD

타이머, 카운터 설정값의
외부 설정과 현재값의 외부 표시



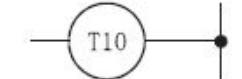
디지털 스위치

X2F-X20



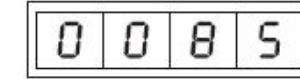
D5 [1 2 3 4]

D5



디지털 표시기

Y4F~Y40

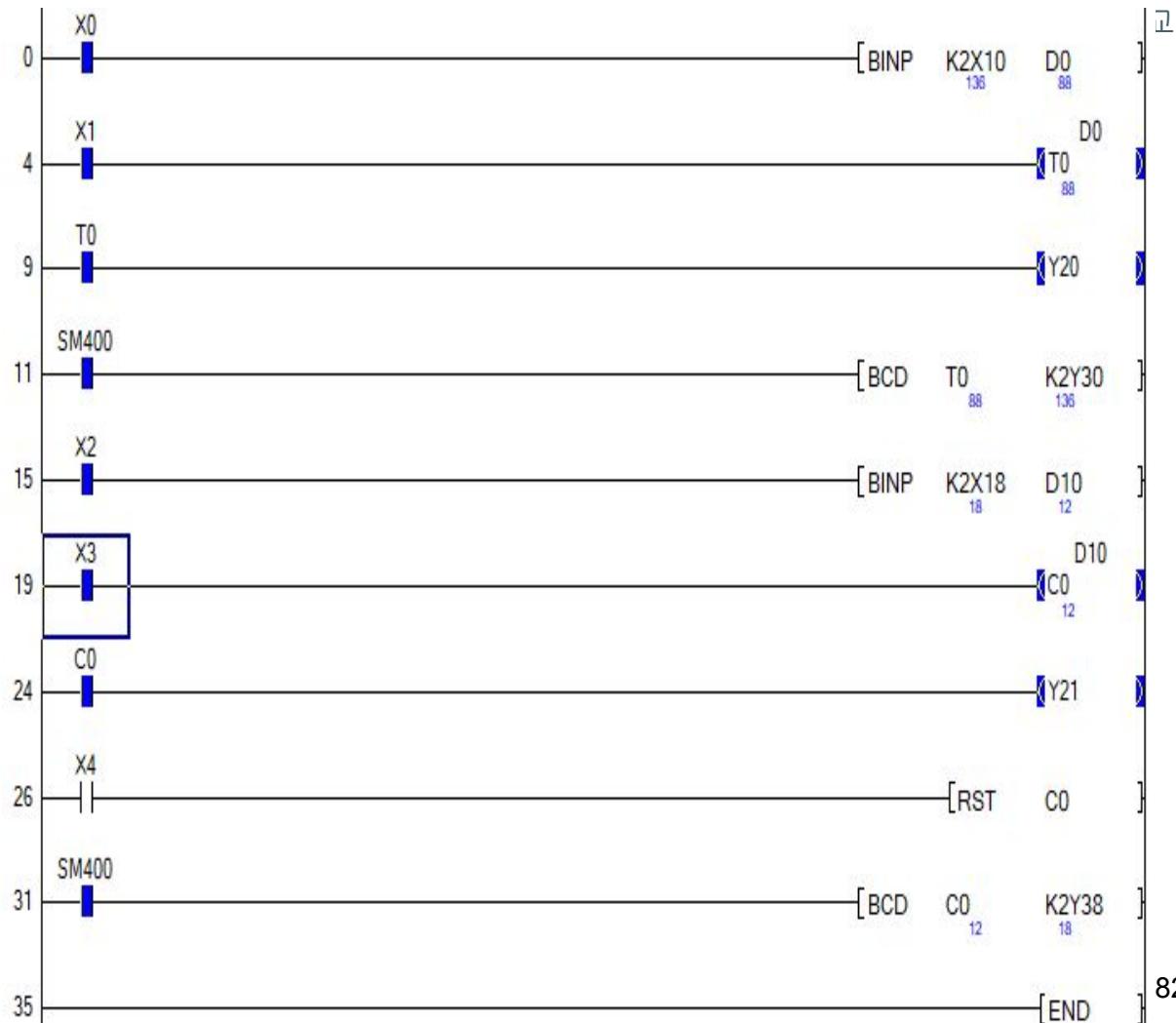


T10의 현재값을 표시

제4장 : 응용명령 사용하기

코드 변환 : BIN / BCD

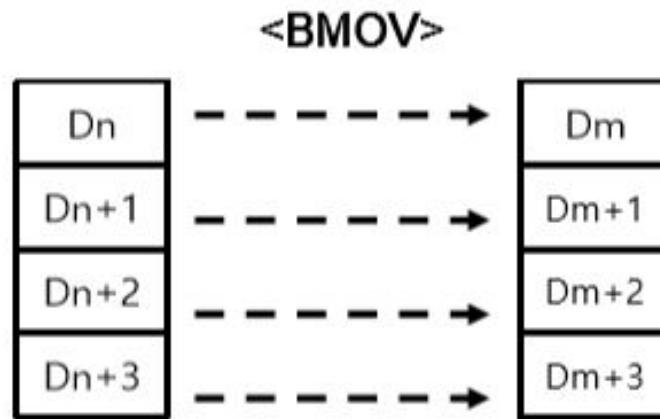
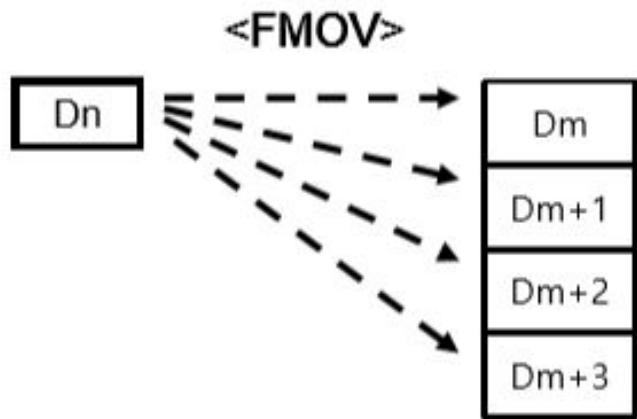
타이머, 카운터 설정값의
외부 설정과 현재값의 외부 표시



FMOV / BMOV

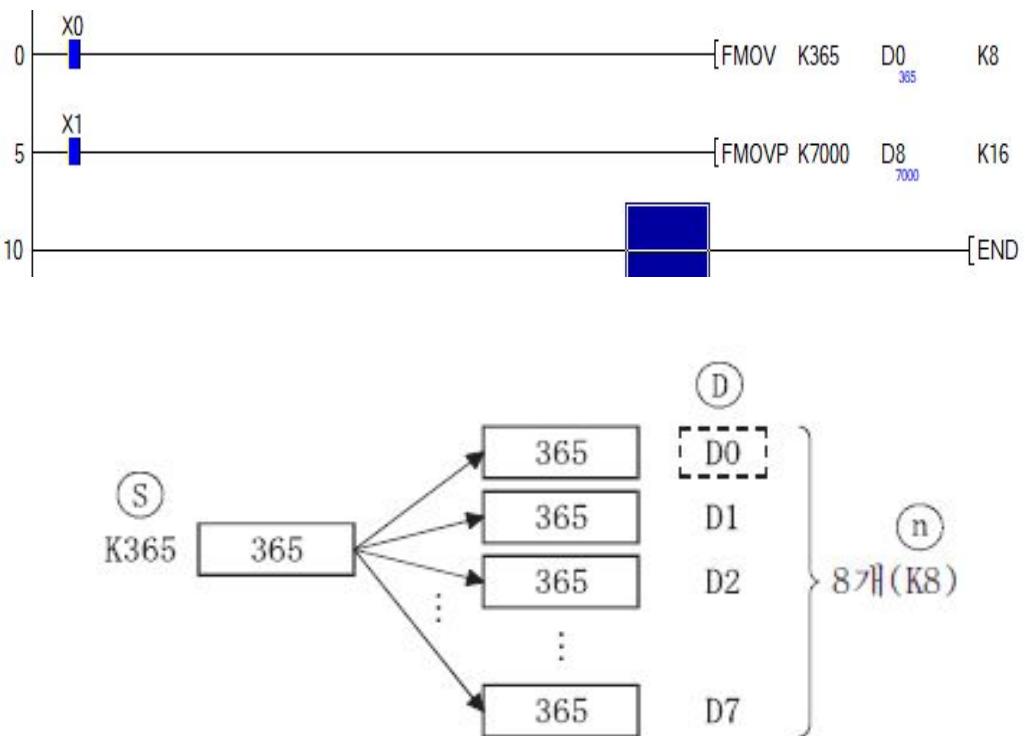
X0 데이터 레지스터데이터
[FMOV K0 D0 K10]

X0 데이터 레지스터데이터
[BMOV D0 D100 K4]



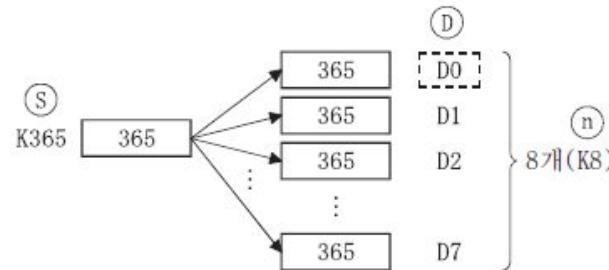
제4장 : 응용명령 사용하기

FMOV : 동일 데이터 일괄 전송



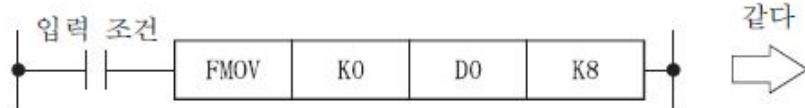
디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	0	365
D1	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D2	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D3	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D4	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D5	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D6	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D7	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D8	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D9	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D10	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D11	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D12	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D13	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D14	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D15	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D16	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D17	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D18	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D19	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D20	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D21	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D22	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D23	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMOV : 동일 데이터 일괄 전송

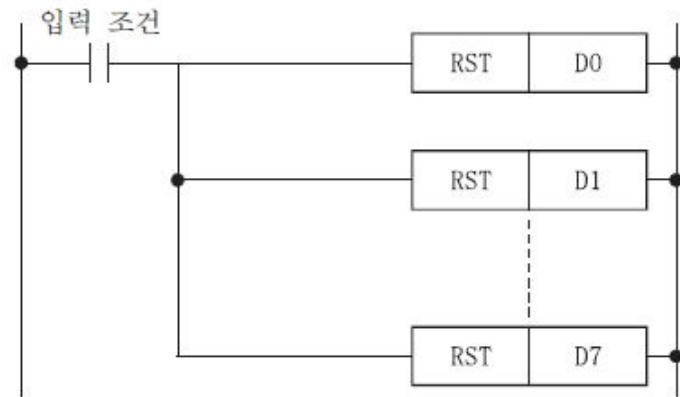


- FMOV 명령은 많은 데이터를 한꺼번에 클리어하는 경우에 사용합니다.

예



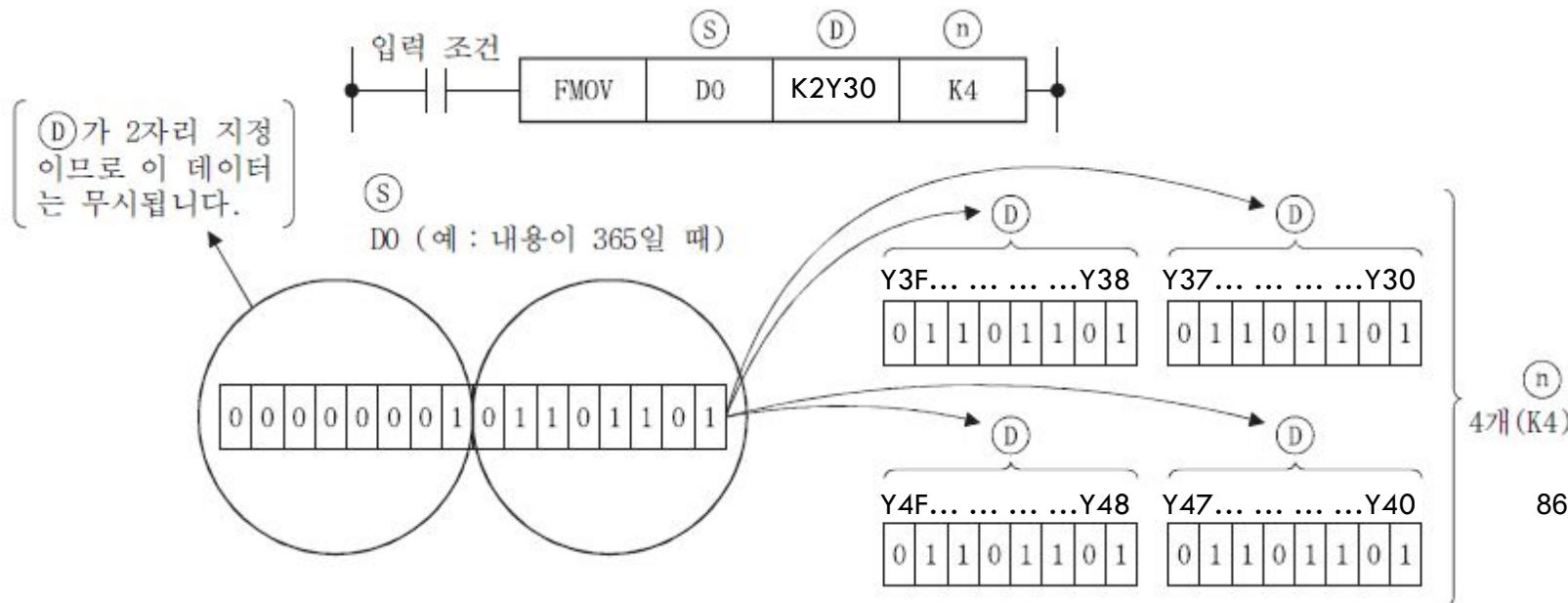
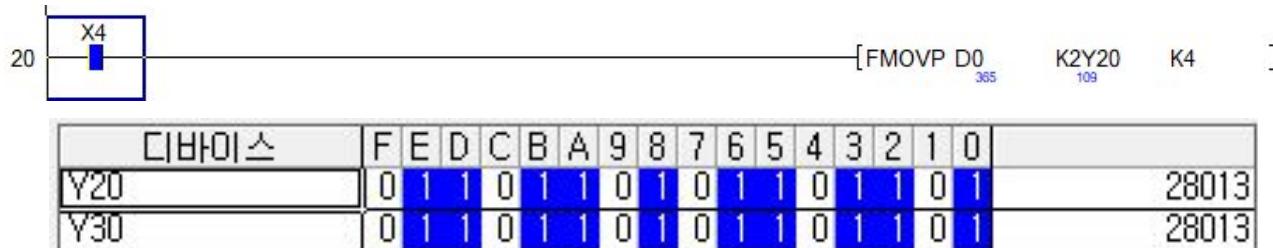
같다



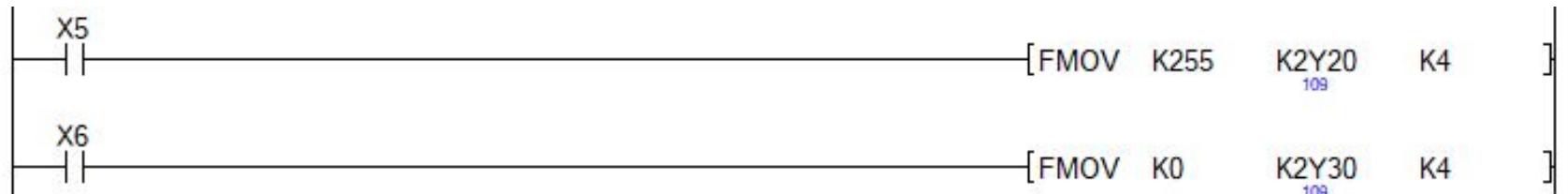
FMOV 명령을 사용하면 상기 오른쪽 그림과 같이 많은 RST 명령을 사용할 필요가 없습니다.

제4장 : 응용명령 사용하기

비트 디바이스일 때 FMOV 명령



비트 디바이스일 때 FMOV 명령



K255의 비트 패턴

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
Y20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1
Y30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

· 비트 디바이스를 4점 단위로

16개까지 OFF 하고 싶을 때는 \rightarrow MOV 명령 예

MOV	K0	K4M0
-----	----	------

32개까지 OFF 하고 싶을 때는 \rightarrow DMOV 명령 예

DMOV	K0	K8M0
------	----	------

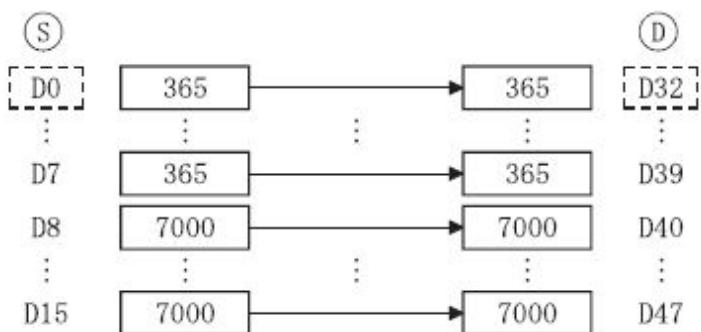
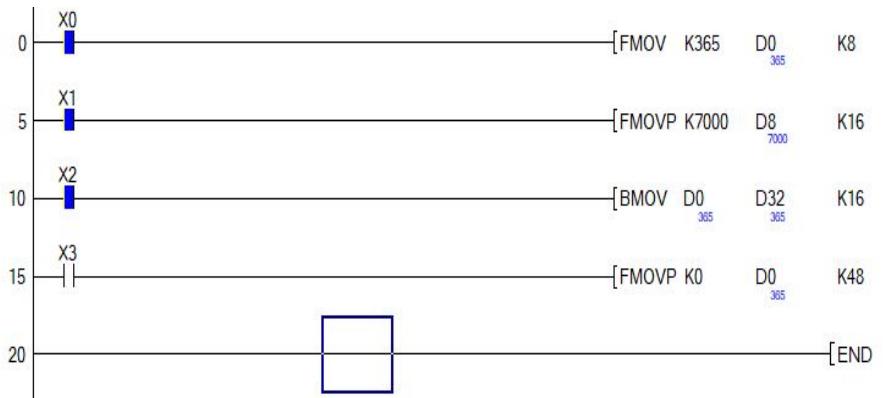
32개를 초과하였을 때는 \rightarrow FMOV 명령 예

FMOV	K0	K4M0	K4
------	----	------	----

 (64개를 OFF 합니다.)

제4장 : 응용명령 사용하기

BMOV : 블록 데이터 일괄 전송



디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D32	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1		365
D33	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D34	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D35	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D36	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D37	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D38	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D39	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	365
D40	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D41	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D42	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D43	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D44	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D45	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D46	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D47	0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	0	7000
D48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

디바이스	+0	+1	+2	+3	+4	+5	+6	+7
D0	365	365	365	365	365	365	365	365
D8	7000	7000	7000	7000	7000	7000	7000	7000
D16	7000	7000	7000	7000	7000	7000	7000	7000
D24	0	0	0	0	0	0	0	0
D32	365	365	365	365	365	365	365	365
D40	7000	7000	7000	7000	7000	7000	7000	7000

비트 디바이스일 때 BMOV 명령



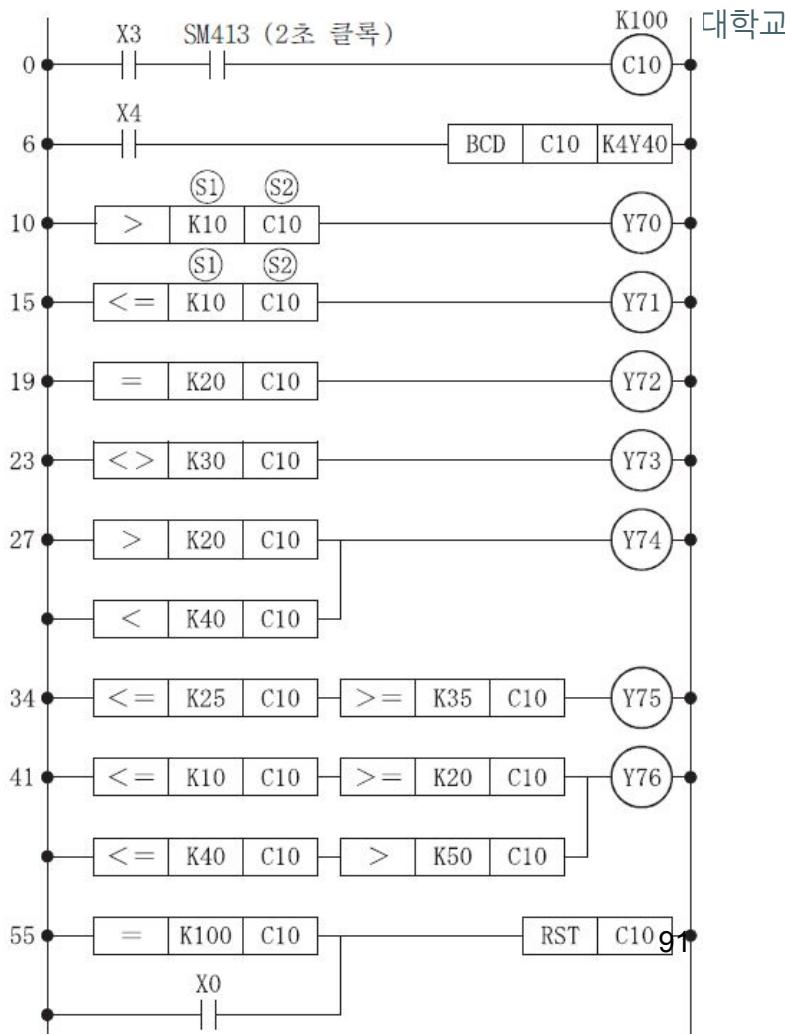
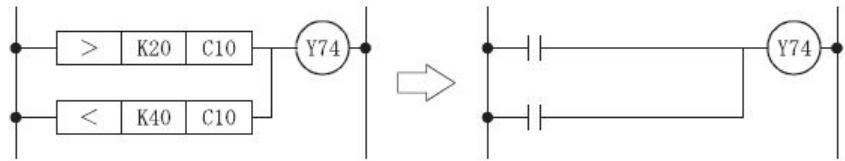
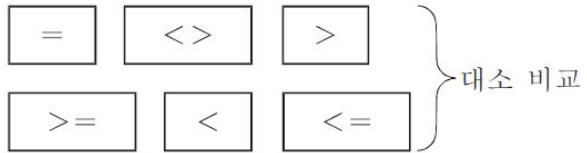
디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Y20	0	1	1	0	1	1	0	1	0	1	1	0	1	1	0	1
Y30	0	1	1	0	1	1	0	1	0	1	1	0	1	1	0	1

BMOV : 블록 데이터 일괄 전송

- BMOV 명령은 다음과 같은 용도에 사용하면 편리합니다.
 - 로깅 데이터의 파일
 - 귀중한 데이터(예 : 자동 운전용 데이터, 측정 데이터 등)의 래치 영역에 대한 임시 저장 (예를 들어, 파라미터에서 정전 유지용으로 설정된 데이터 레지스터에 임시 저장하여, 뜻하지 않은 정전에 의해 데이터가 소멸되는 것을 방지할 수 있습니다.)

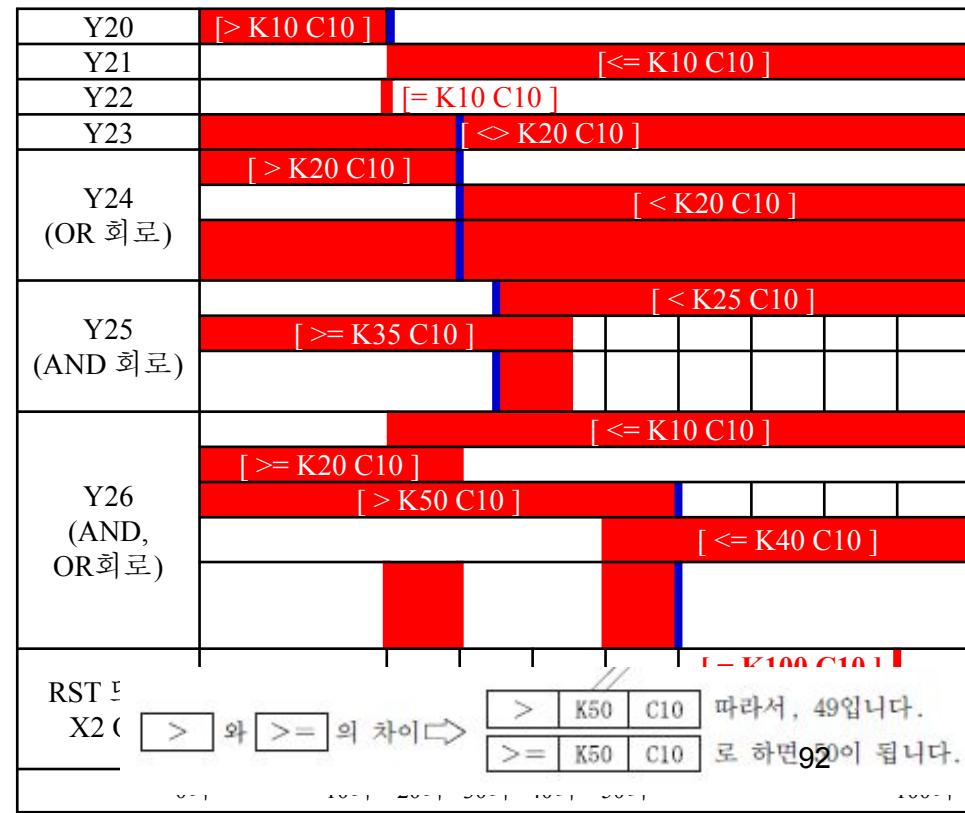
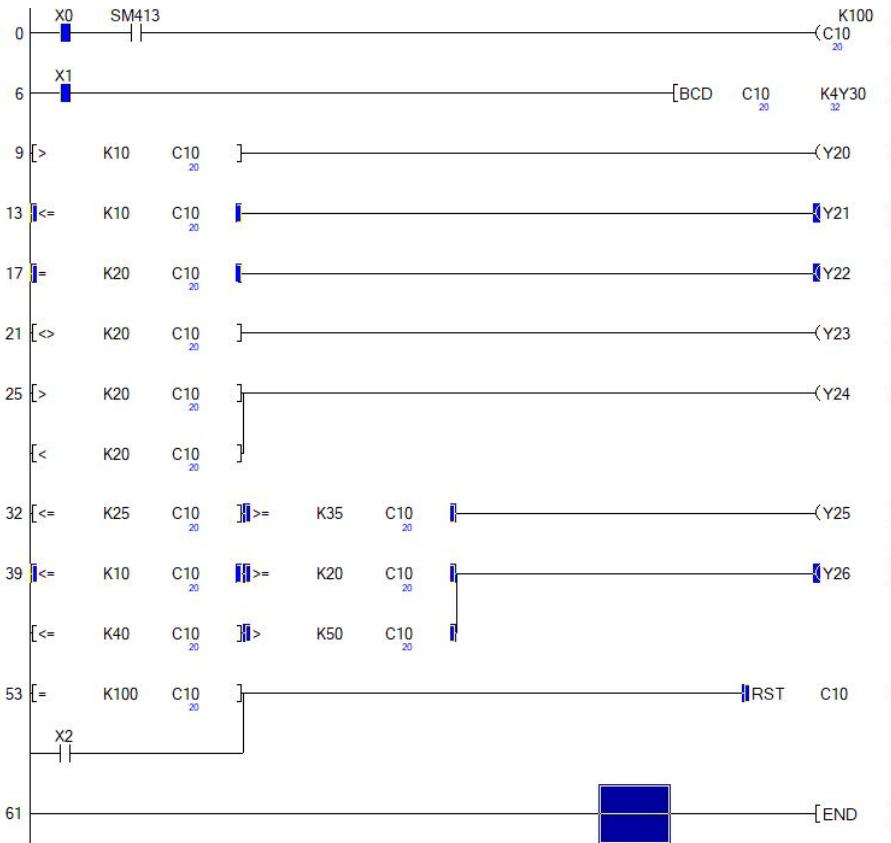
제4장 : 응용명령 사용하기

비교연산명령



제4장 : 응용명령 사용하기

비교연산명령

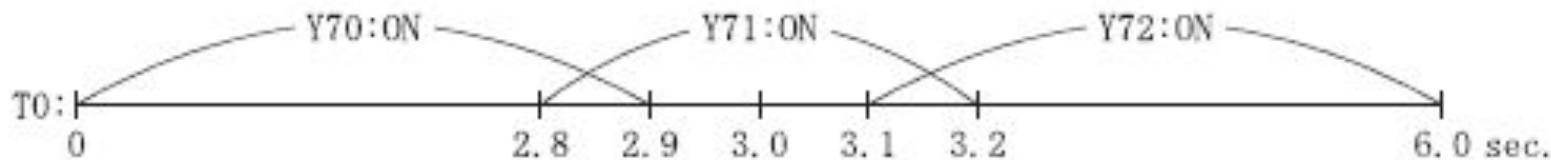


비교연산명령

래더 예

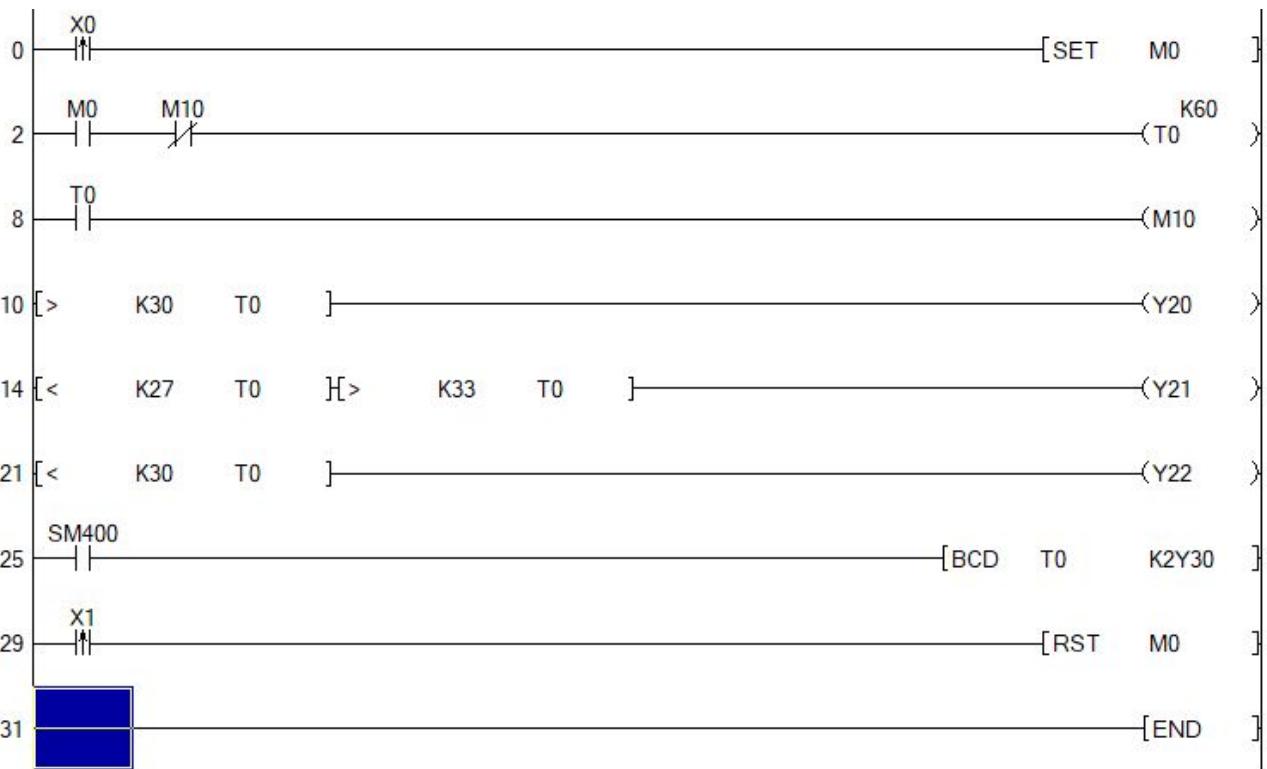
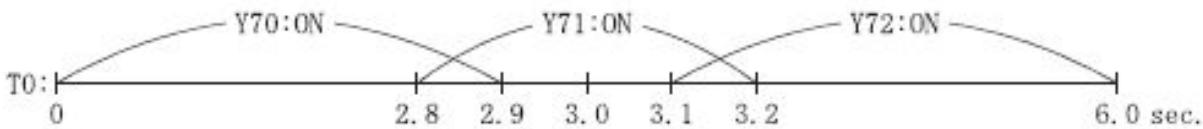
아래의 래더를 읽고, 실습기에 쓴 후 >, < 명령의 실행을 확인하십시오.

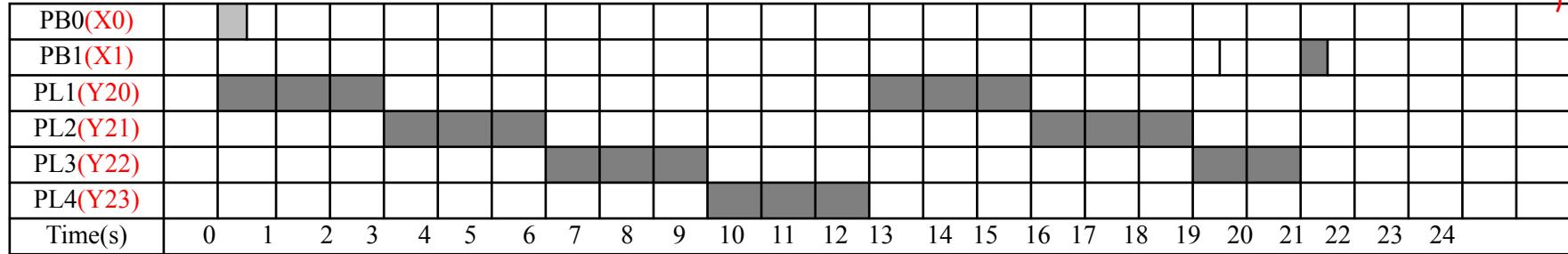
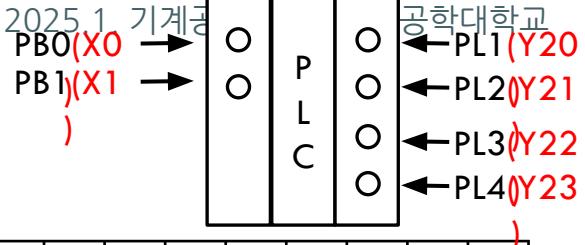
$0\text{sec} \leq T0 < 3\text{sec} \rightarrow Y70 : \text{ON}$, $2.7\text{sec} < T0 < 3.3\text{sec} \rightarrow Y71 : \text{ON}$, $3\text{sec} < T0 \leq 6\text{sec} \rightarrow Y72 : \text{ON}$

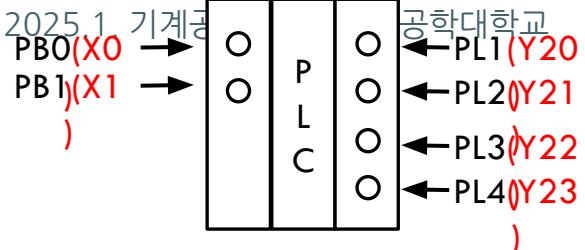


제4장 : 응용명령 사용하기

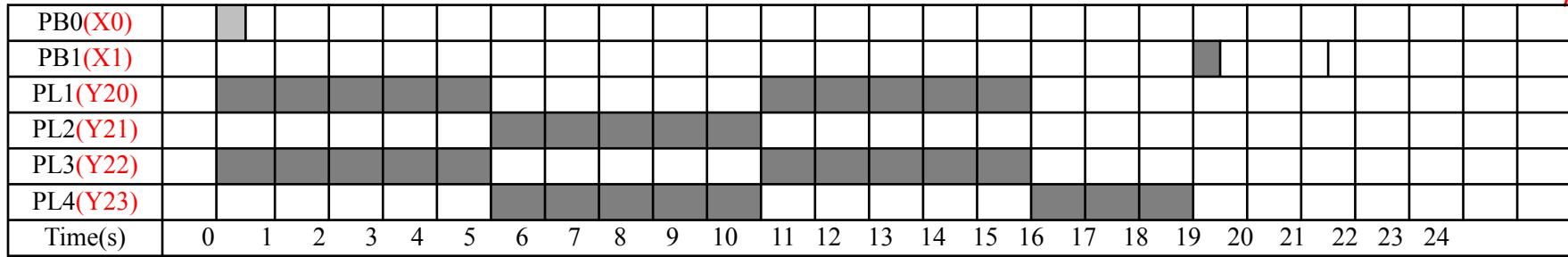
비교연산명령

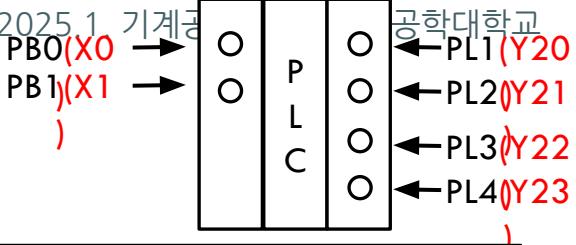




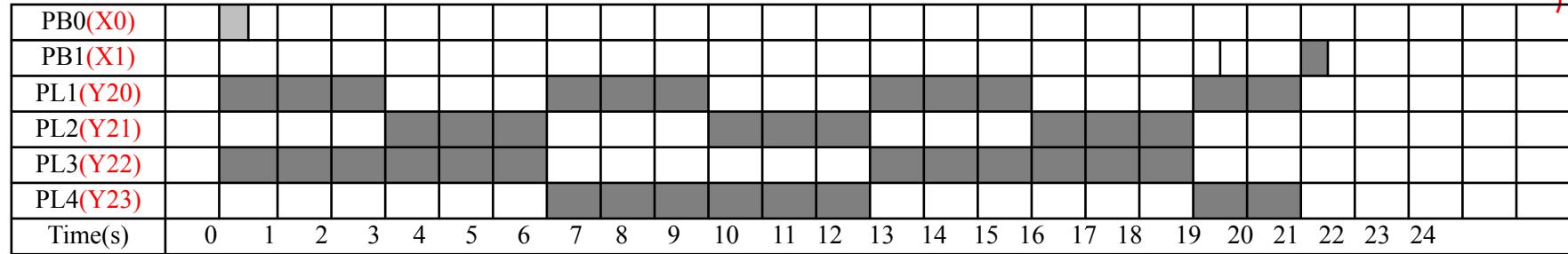


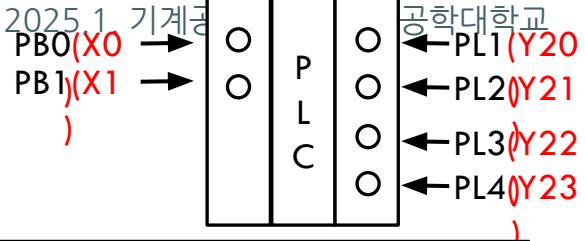
프로그램 연습2 ; 비교연산명령



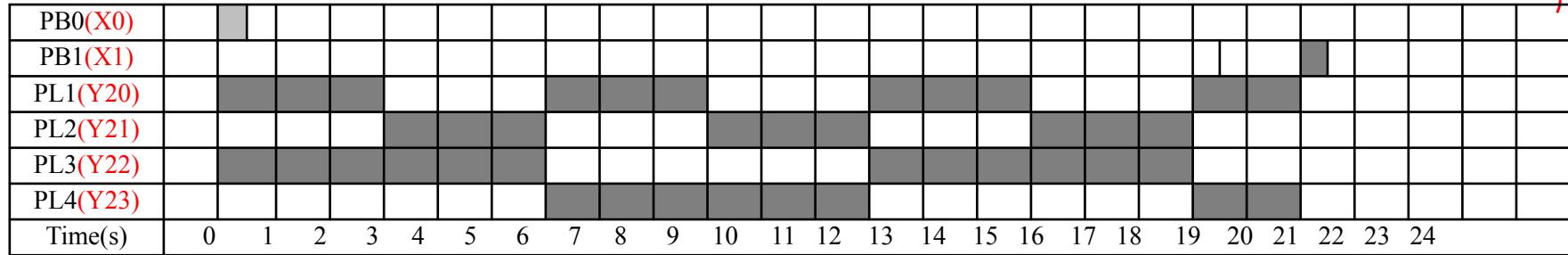


프로그램 연습3 ; 비교연산명령





프로그램 연습4 ; 비교연산명령

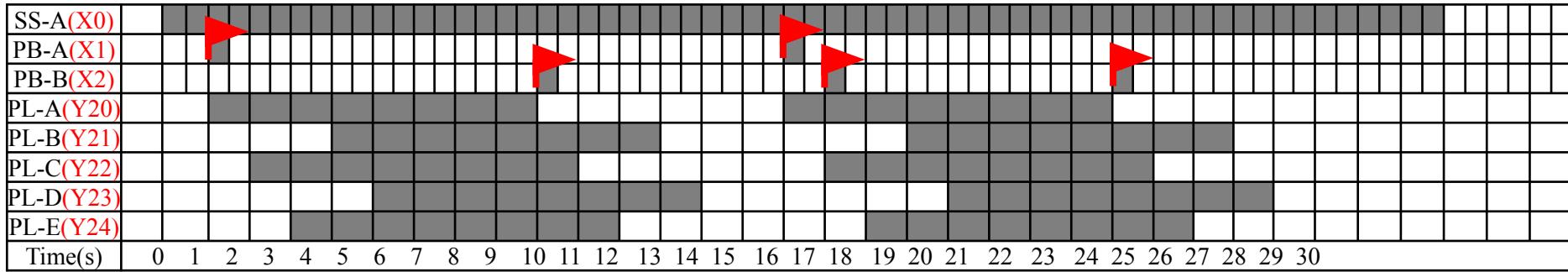
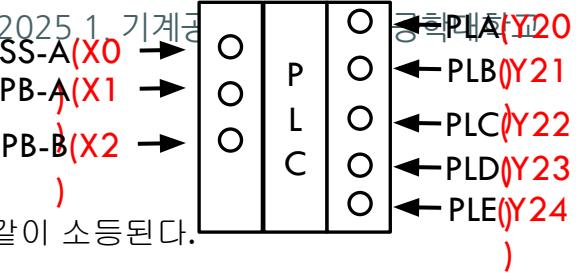


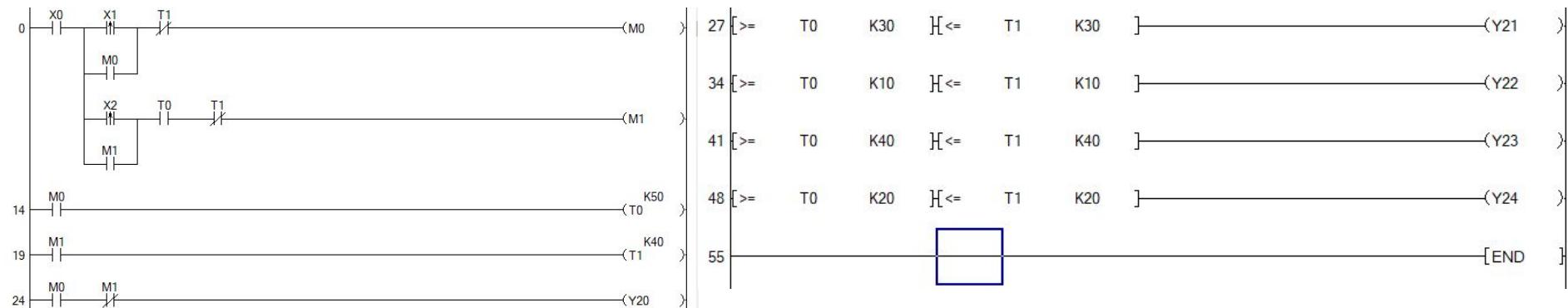
제4장 : 응용명령 사용하기

프로그램 연습5 ; 비교연산명령

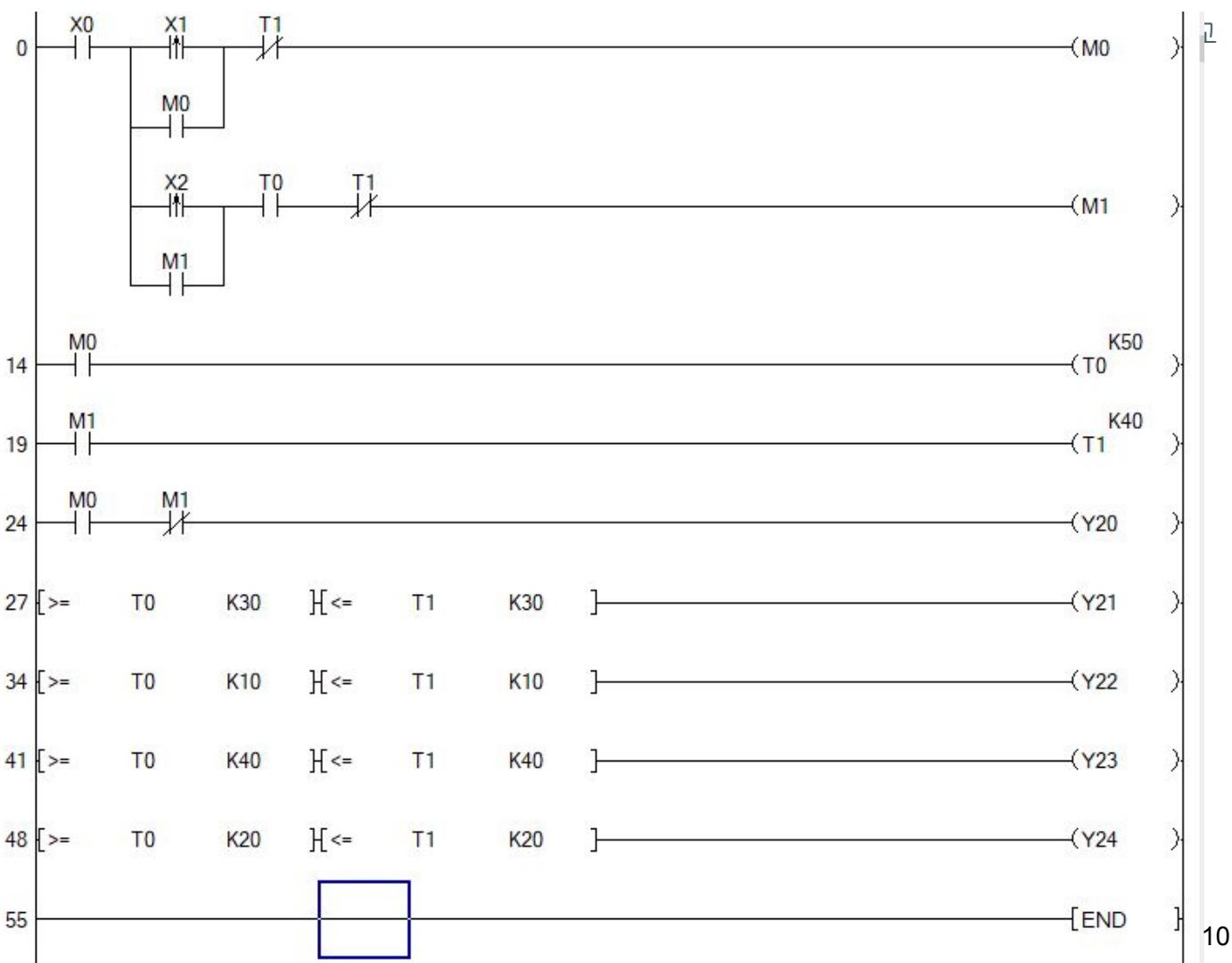
요구사항

SS-A가 입력된 상태에서 PB-A가 입력되면 아래와 같이 점등되며, PB-B가 입력되면 아래와 같이 소등된다.
단, PB-A가 입력되고 5초간 입력되지 못한다.





제4장 : 응용명령 사용하기

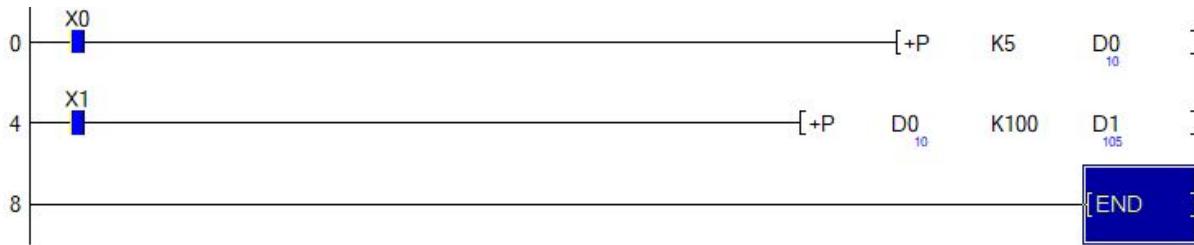
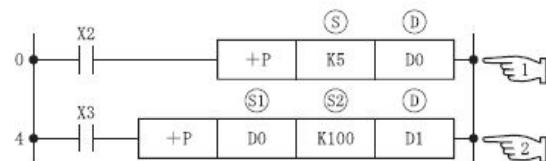


제4장 : 응용명령 사용하기

사칙연산 명령

+ (P) BIN16비트 데이터의 덧셈

- (P) BIN16비트 데이터의 뺄셈



- ☞ 입력 조건이 "ON" 할 때마다 ①로 지정된 디바이스의 내용에 ③로 지정된 디바이스의 내용을 더하여 결과를 ④의 디바이스에 저장합니다.

(입력 조건)	①	+	③	→	④
1번째 ON	0 (가정)	+	5	→	5
2번째 ON	5	+	5	→	10
3번째 ON	10	+	5	→	15

D0의 내용이 바뀐다

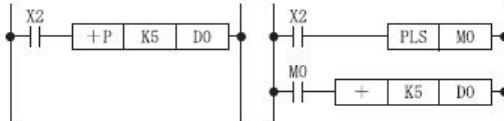
- ☞ 입력 조건이 "ON"되면, ⑤로 지정된 디바이스의 내용과 ⑥로 지정된 디바이스의 내용을 더하여 결과를 ⑦의 디바이스에 저장합니다.

(입력 조건)	⑤	+	⑥	→	⑦
ON	15 (가정)	+	100	→	115

D0의 내용은 덧셈의 전후에 바뀌지 않는다.

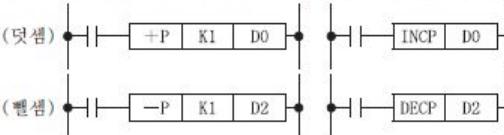
주의

- 덧셈/뺄셈 명령은 반드시 **[+P]** **[-P]** 를 사용하십시오.
- 뺄셈 명령을 사용하지 않으면 매스킹 덧셈/뺄셈을 실행합니다. 다만 미리 펠스화한 조건이면 사용이 가능합니다.



참고

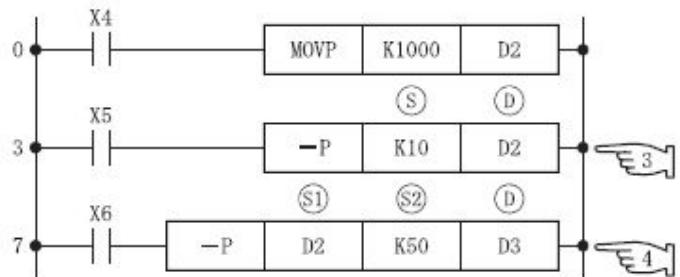
- 덧셈/뺄셈 처리에서 다음의 명령은 모두 같은 기능을 합니다.



제4장 : 응용명령 사용하기

사칙연산 명령

- + (P) BIN16비트 데이터의 덧셈
- (P) BIN16비트 데이터의 뺄셈



• 입력 조건이 "ON"할 때마다 ①로 지정된 디바이스의 내용에서 ⑤로 지정된 디바이스의 내용을 빼서, 결과를 ②의 디바이스에 저장합니다.

(입력 조건)	D2 ①	-	(S) ⑤	→	D2 ②
1번째 ON	1000 (가정)	-	10	→	990
2번째 ON	990	-	10	→	980
3번째 ON	980	-	10	→	970

D2의 내용이 바뀐다

• 입력 조건이 "ON"되면, ③로 지정된 디바이스의 내용에서 ④로 지정된 디바이스의 내용을 빼서, 결과를 ⑤의 디바이스에 저장합니다.

(입력 조건)	D2 ③	-	(S) ④	→	D3 ⑤
ON	970 (가정)	-	50	→	920

D2의 내용은 뺄셈의 전후에 바뀌지 않는다.

제4장 : 응용명령 사용하기

사칙연산 명령

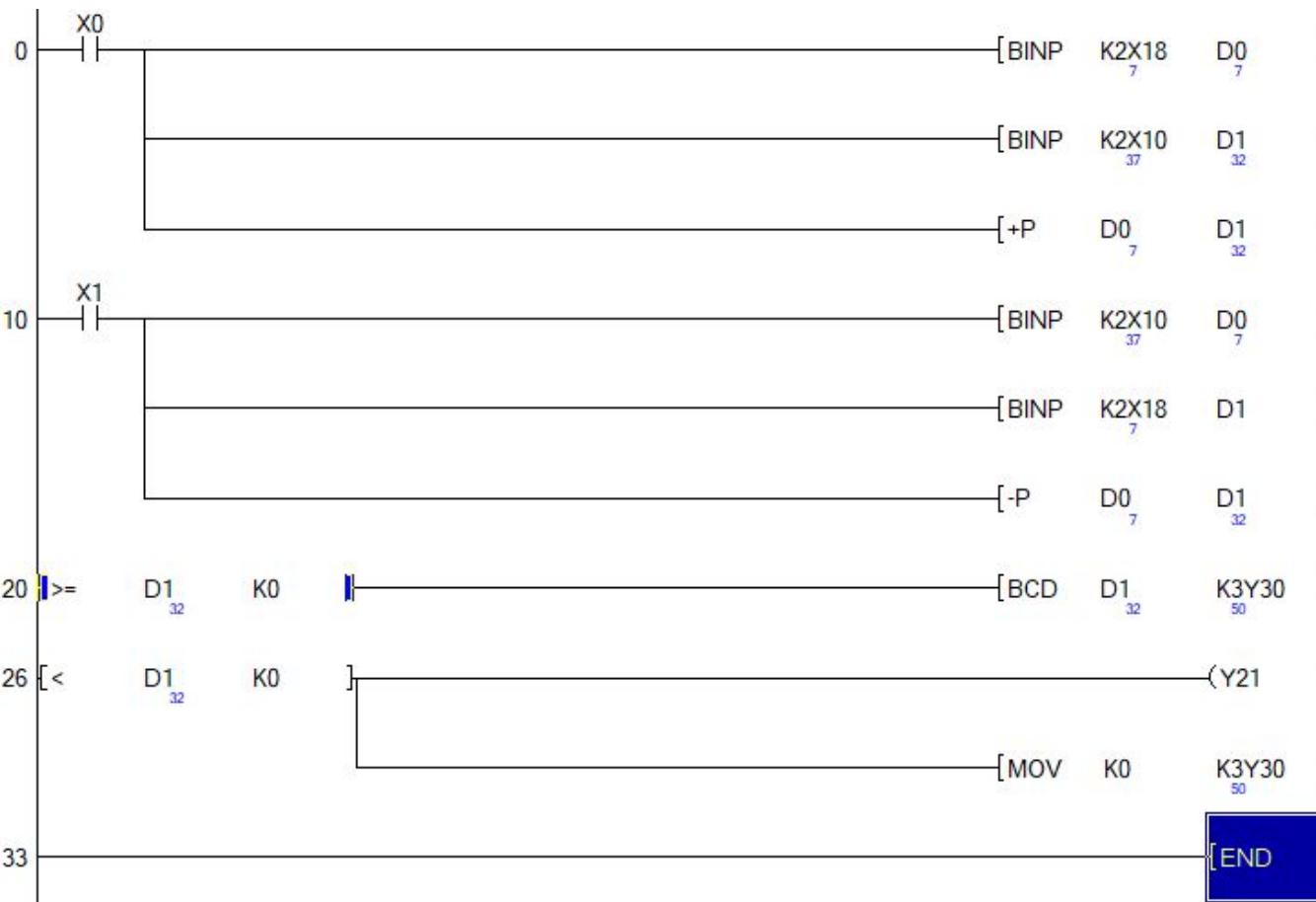
+ (P) BIN16비트 데이터의 덧셈
- (P) BIN16비트 데이터의 뺄셈



제4장 : 응용명령 사용하기

사칙연산 명령

- + (P) BIN16비트 데이터의 덧셈
- (P) BIN16비트 데이터의 뺄셈

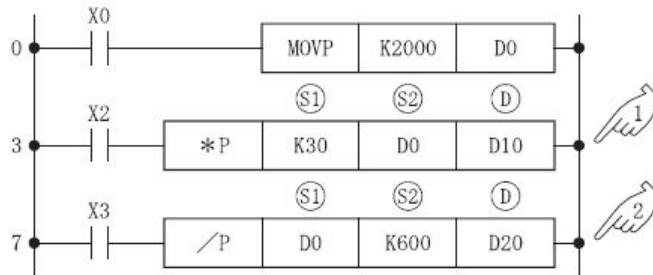


제4장 : 응용명령 사용하기

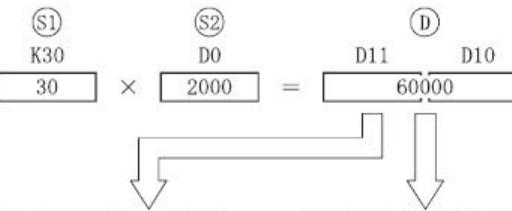
사칙연산 명령

* (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈



· 입력 조건이 ON 되면 ①로 지정된 디바이스의 내용과 ②로 지정된 디바이스의 내용의 곱셈을 실행하여, 그 결과를 ③로 지정된 디바이스에 저장합니다.



16비트 데이터 \times 16비트 데이터의 응답을 저장하기 위해서는 16비트(1워드)로는 불충분합니다.
따라서 프로그램으로 지정한 D10과 다음 디바이스 번호의 D11이 응답의 디바이스로 사용됩니다.

응답의 디바이스는 32비트의 레지스터로 취급하므로, D10의 왼쪽의 비트(B15)는 양/음 판정 비트가 아니라 데이터부의 1부로 취급됩니다.

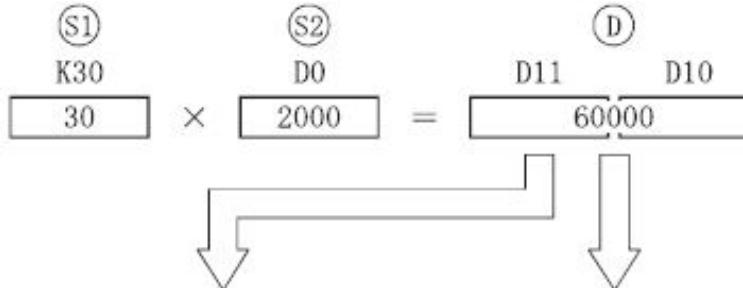
*(P) 명령으로 얻은 연산 결과를 이용하여 프로그래밍 할 때는 32비트 취급 명령을 사용해야 합니다.(예를 들어, DMOV 명령, DBCD 명령 등입니다.)

제4장 : 응용명령 사용하기

사칙연산 명령

★ (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈



16비트 데이터 \times 16비트 데이터의 응답을 저장하기 위해서는 16비트(1워드)로는 불충분합니다.

따라서 프로그램으로 지정한 D10과 다음 디바이스 번호의 D11이 응답의 디바이스로 사용됩니다.

응답의 디바이스는 32비트의 레지스터로 취급하므로, D10의 왼쪽의 비트(B15)는 양/음 판정 비트가 아니라 데이터부의 1부로 취급됩니다.

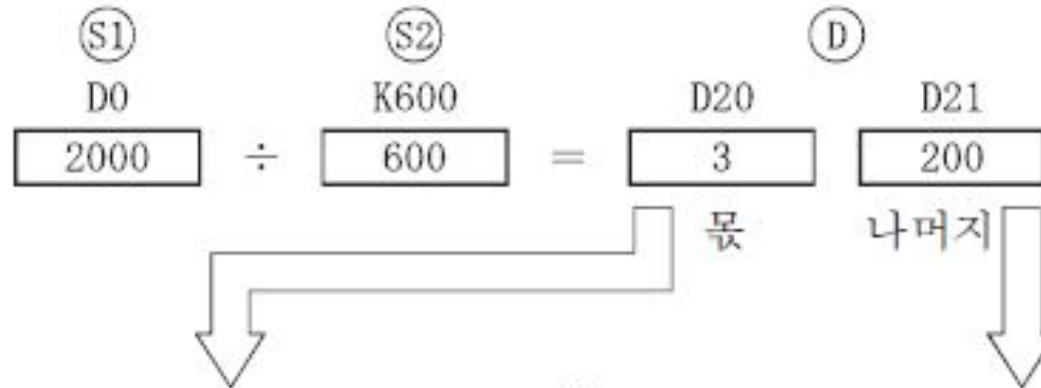
*(P) 명령으로 얻은 연산 결과를 이용하여 프로그래밍 할 때는 32비트 취급 명령을 사용해야 합니다.(예를 들어, DMOV 명령, DBCD 명령 등입니다.)

제4장 : 응용명령 사용하기

사직연산 명령

* (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈



프로그램에서 지정한 D20에는 뭇이 들어갑니다.

그리고
→

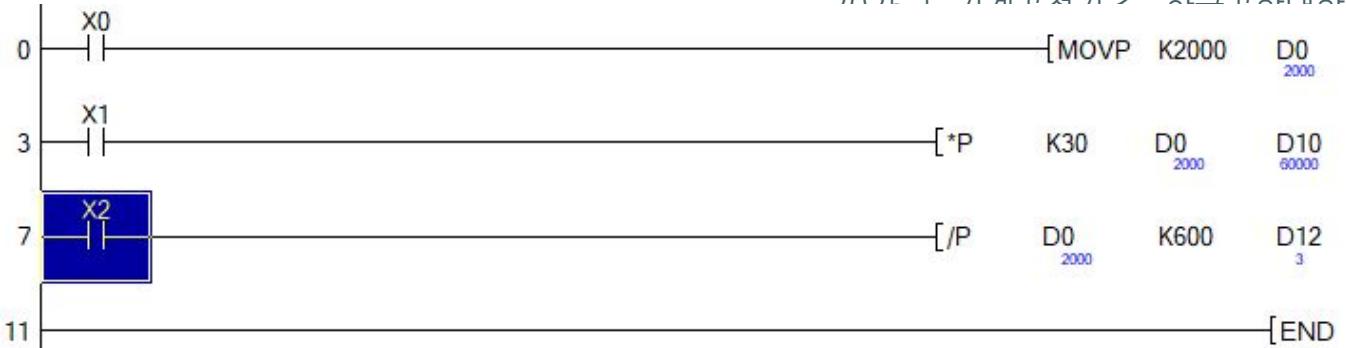
이 다음 디바이스 번호의 D21에는
나머지가 들어갑니다.

제4장 : 응용명령 사용하기

사칙연산 명령

* (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈



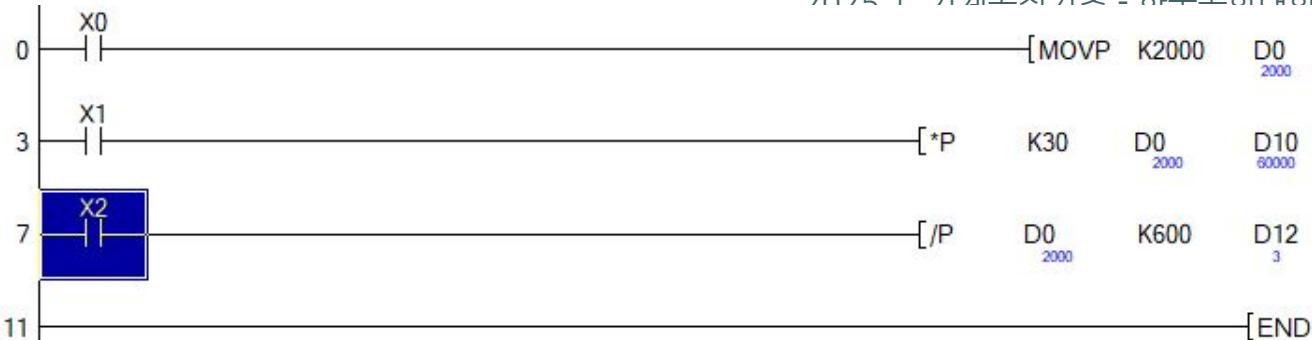
디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	2000
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D10	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0	60000
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	13107203
D13	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	
D14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

제4장 : 응용명령 사용하기

사칙연산 명령

* (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈



나눗셈의 연산 결과로 소수점 이하의 수치는 취급하지 않습니다.

- (D)에 비트 디바이스를 지정한 경우, 몫은 저장되지만 나머지는 저장되지 않습니다.
- 음의 값을 취급하면, 다음과 같이 됩니다.

예 $-5 \div (-3) = 1$, 나머지 -2

$$5 \div (-3) = -1, \text{ 나머지 } 2$$

- 0으로 나누거나 0을 나누면 다음과 같이 됩니다.

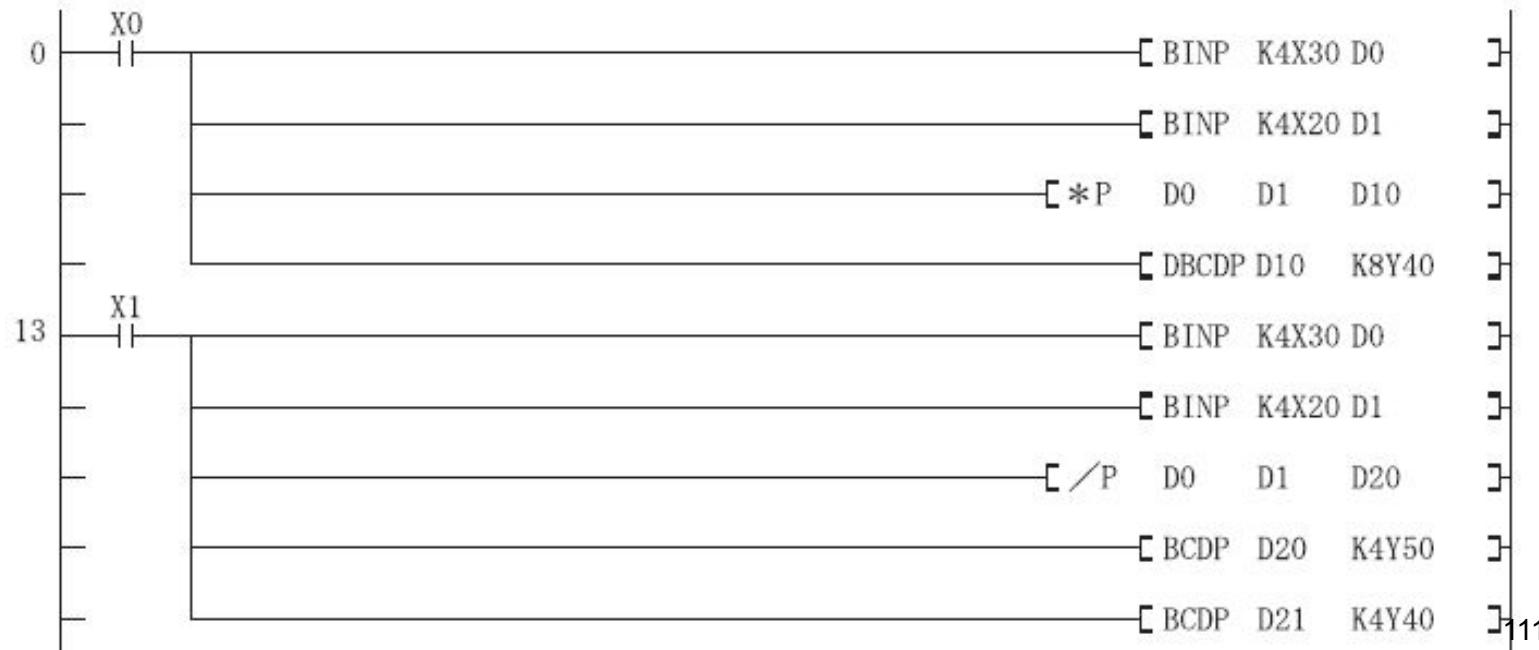
예 $\left. \begin{array}{l} 0 \div 0 \\ 1 \div 0 \end{array} \right\}$ 에러 "OPERATION ERROR"

$$0 \div 1 \quad \text{몫, 나머지 모두 } 0$$

제4장 : 응용명령 사용하기

사칙연산 명령

* (P) BIN16비트의 곱셈
/ (P) BIN16비트의 나눗셈



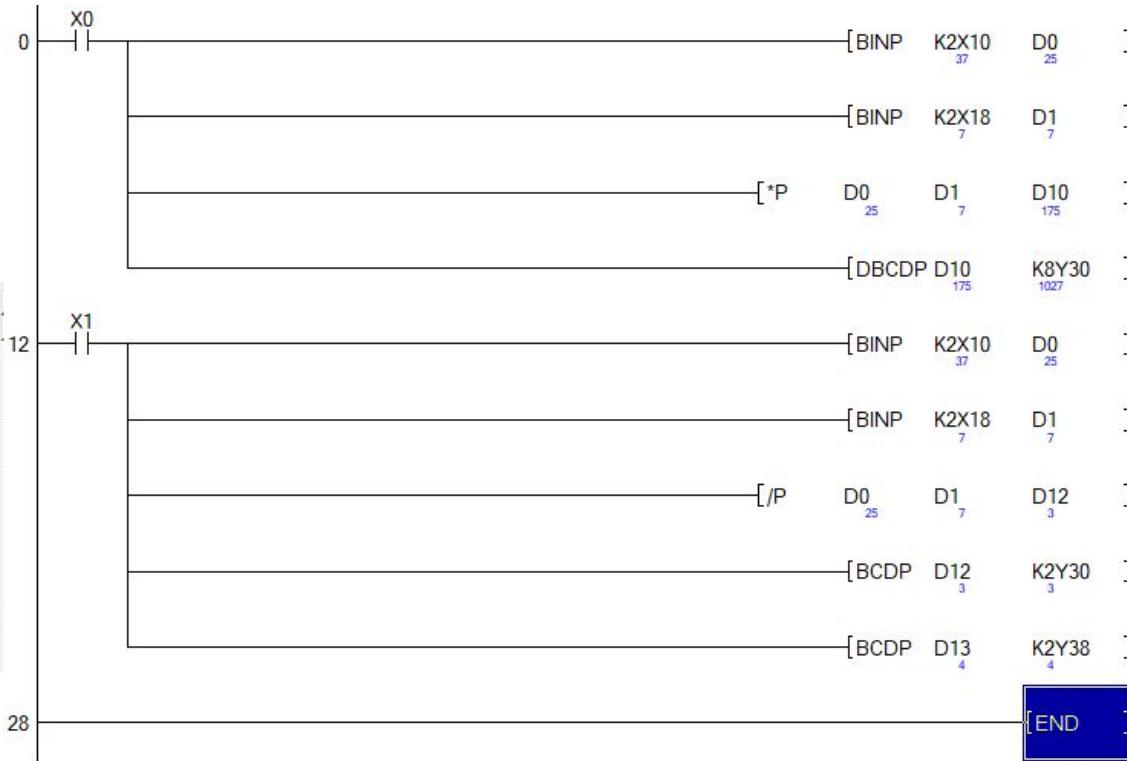
제4장 : 응용명령 사용하기

사칙연산 명령

* (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈

디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1		25
D1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1		7
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D10	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1		175
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D12	0	0	0	0	0	0	0	0	0	0	0	1	1	1			3
D13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		4
D14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
D15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0



제4장 : 응용명령 사용하기

32비트 데이터 명령과 그 필요성

사칙연산 명령

★ (P) BIN16비트의 곱셈

/ (P) BIN16비트의 나눗셈

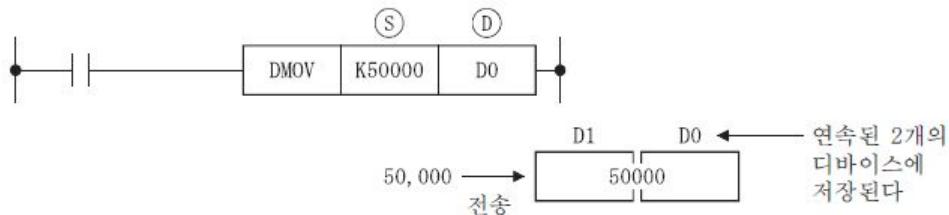
데이터 명령	1워드 16비트	2워드 32비트
전송	MOV (P)	DMOV (P)
	BIN (P)	DBIN (P)
	BCD (P)	DBCD (P)
비교	<, >, <=, >=, =, <>	D<, D>, D<=, D>=, D=, D<>
사칙 연산	+ (P)	D+ (P)
	- (P)	D- (P)
	* (P)	D* (P)
	/ (P)	D/ (P)
취급할 수 있는 수치 범위	-32,768 ↓ 32,767 () 안은 BIN(P), BCD(P) 명령 시	0 ↓ 9,999 -2,147,483,648 ↓ 2,147,483,647 0 ↓ 99,999,999 () 안은 DBIN(P), DBCD(P) 명령 시
자리 지정 가능 범위	K1~K4	K1~K8

사칙연산 명령

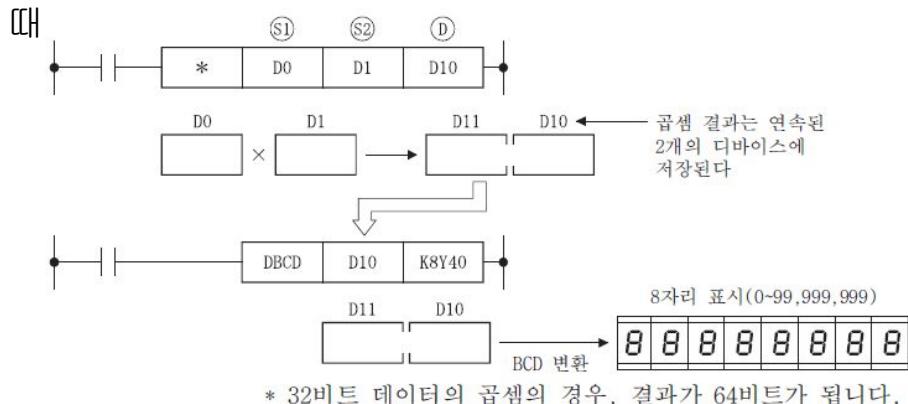
32비트 데이터 명령과 그 필요성

- 2워드(32비트) 취급으로 할지 여부는 취급하는 데이터의 크기에 의해 결정
다음과 같은 경우에는 **2워드 명령으로 할 필요가 있음**

- ① 데이터가 1워드로 취급하여 범위(-32768~32767)를 초과하였을 때.



- ② 16비트 곱셈 명령(1워드 명령)의 결과를 전송할 때



제4장 : 응용명령 사용하기

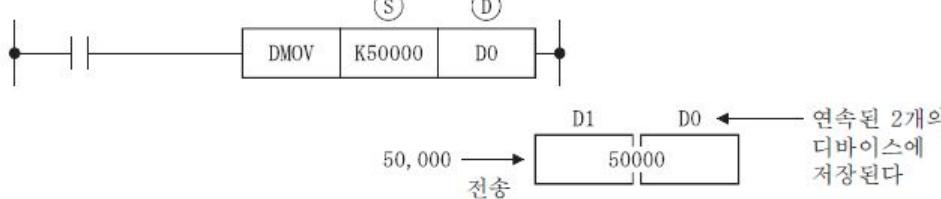
32비트 데이터 명령과 그 필요성

사칙연산 명령

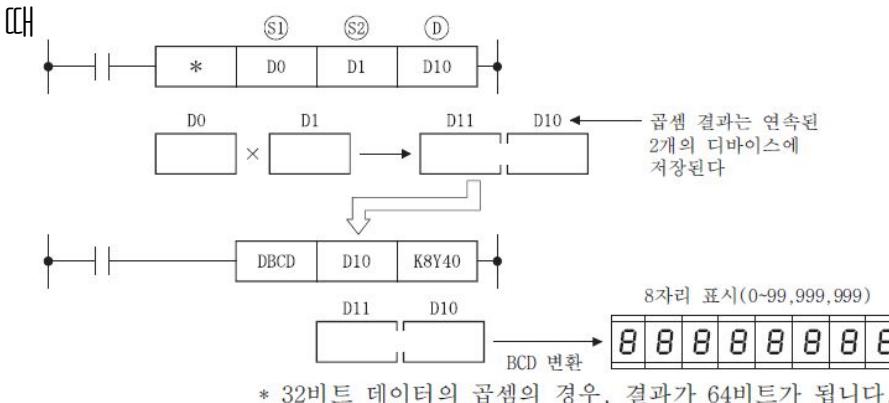
- 2워드(32비트) 취급으로 할지 여부는 취급하는 데이터의 크기에 의해 결정

다음과 같은 경우에는 **2워드 명령으로 할 필요가 있음**

- ① 디바이스가 16비트 친구에서 50,000, 50000을 주기하였을 때.



- ② 16비트 곱셈 명령(1워드 명령)의 결과를 전송할 때

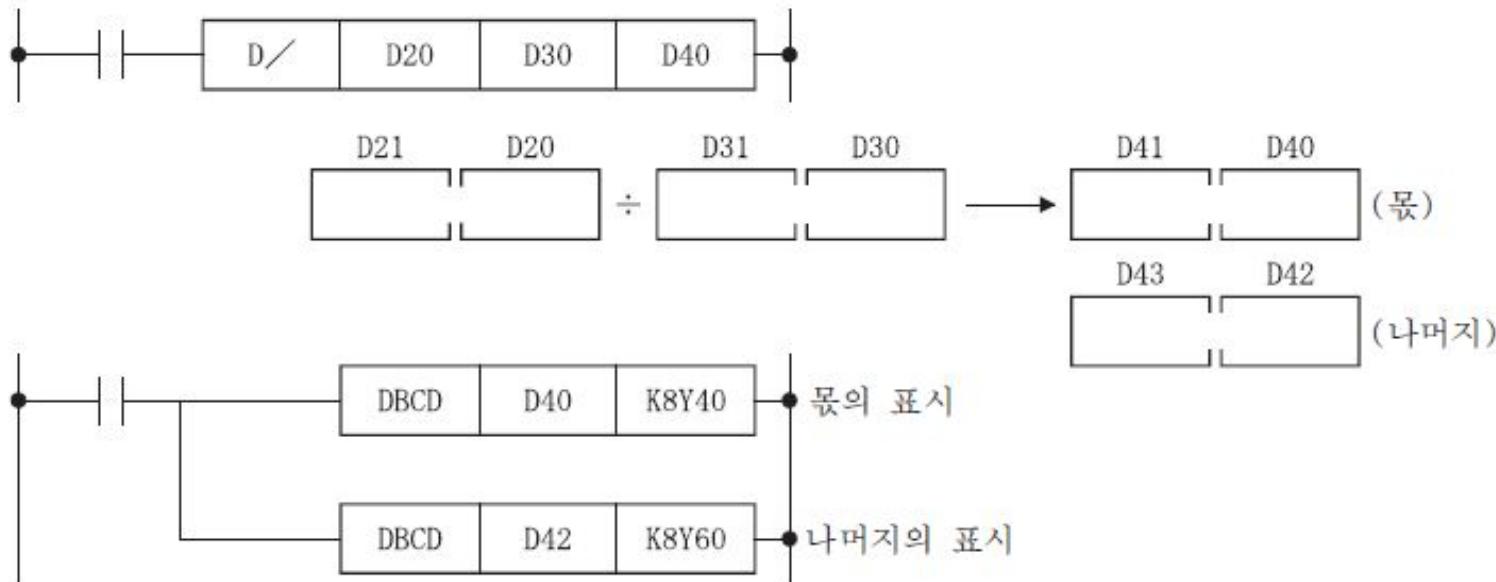


제4장 : 응용명령 사용하기

사칙연산 명령

32비트 데이터 명령과 그 필요성

- ③ 32비트 나눗셈 명령의 결과를 이용할 때.



사직연산 명령

소수점을 포함한 값의 곱셈과 나눗셈 계산 예(사칙 연산 명령의 *, / 를 사용한 경우)

(예 1)의 계산을

예 1 원주를 구하는 계산 예

디지털 스위치의 합 $\times 3.14 \rightarrow$ **정수부 표시** 와 **소수부 표시**

(K4X30) (원주율) (K8Y50) (K2Y48)

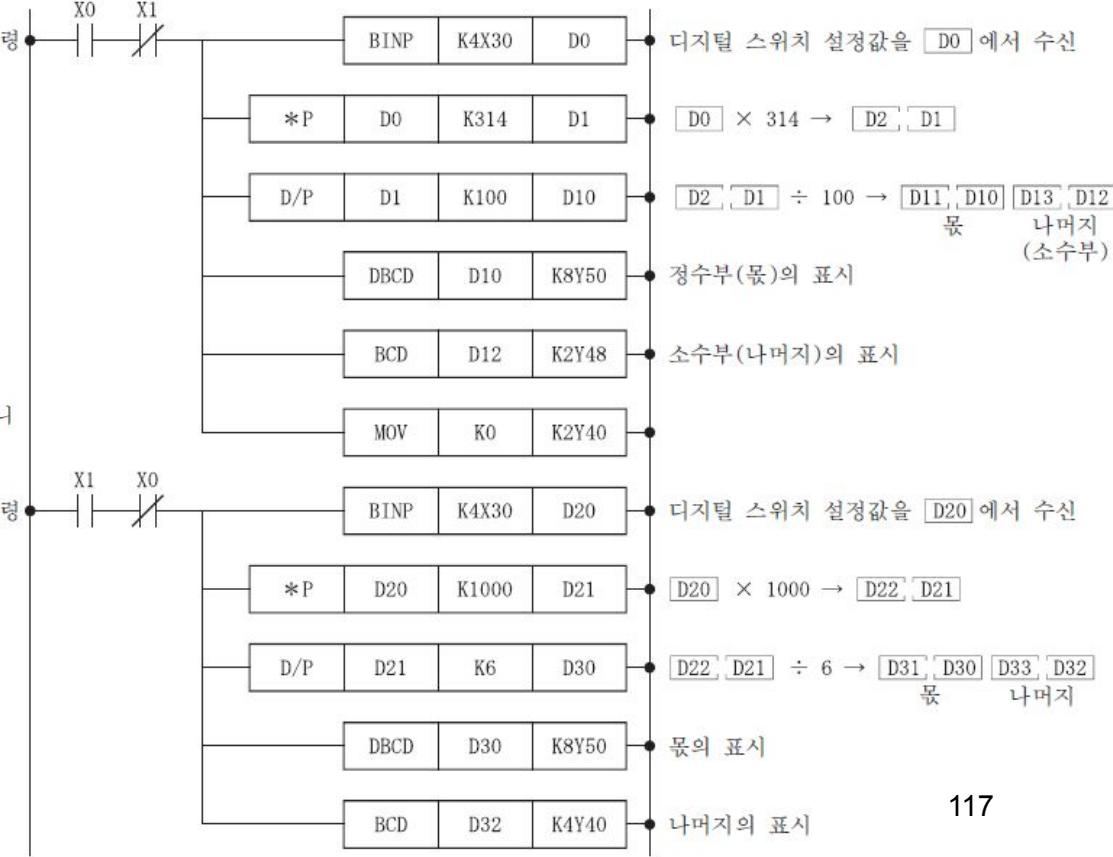
- 프로그래밍 방법
원주율 3.14를 100배의 314로 취급하고, 이후에 100으로 나눕니다.

예 2 소수점 이하의 작은 데이터를 취급하는 계산 예(나누셉의 예)

디지털 스위치의 값 (K4X30)	$\div 0.006 \rightarrow$	몫의 표시 (K8Y50)	와	나머지의 표시 (K4Y40)
-----------------------	--------------------------	------------------	---	--------------------

- 프로그래밍 방법
0.006을 6이라고 하는 상수로 취급하므로, 제수, 피제수 모두 1000배가 됩니다.

(예 2)의 계산을



제4장 : 응용명령 사용하기

사칙연산 명령

소수점을 포함한 값의 곱셈과 나눗셈 계산 예(사칙 연산 명령의 *, / 를 사용한 경우)

예 1 원주를 구하는 계산 예

디지털 스위치의 값	×	3.14	→	정수부 표시	와	소수부 표시
(K4X30)		(원주율)		(K8Y50)		(K2Y48)

• 프로그래밍 방법

원주율 3.14를 100배의 314로 취급하고, 이후에 100으로 나눕니다.

예 2 소수점 이하의 작은 데이터를 취급하는 계산 예(나눗셈의 예)

디지털 스위치의 값	÷	0.006	→	몫의 표시	와	나머지의 표시
(K4X30)		(K8Y50)		(K4Y40)		

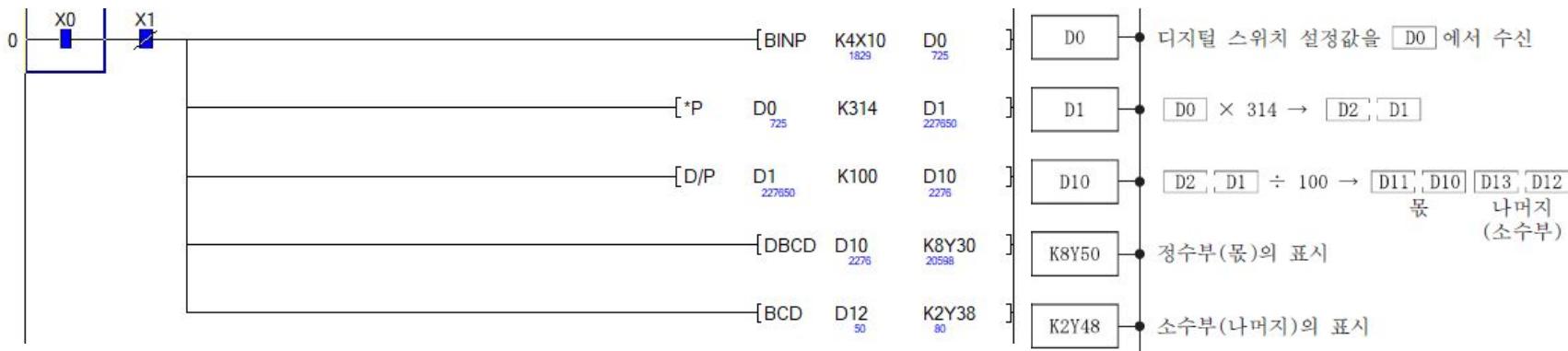
• 프로그래밍 방법

0.006을 6이라고 하는 상수로 취급하므로, 제수, 피제수 모두 1000배가 됩니다.

제4장 : 응용명령 사용하기

사칙연산 명령

소수점을 포함한 값의 곱셈과 나눗셈 계산 예(사칙 연산 명령의 *, / 를 사용한 경우)



디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	2034369237
D1	0	1	1	1	1	0	0	1	0	1	0	0	0	0	1	0	
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D10	0	0	0	0	1	0	0	0	1	1	1	0	0	1	0	0	2276
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D12	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	50

제4장 : 응용명령 사용하기

사칙 연산 명령

소수점을 포함한 값의 곱셈과 나눗셈 계산 예(사칙 연산 명령의 *, / 를 사용한 경우)



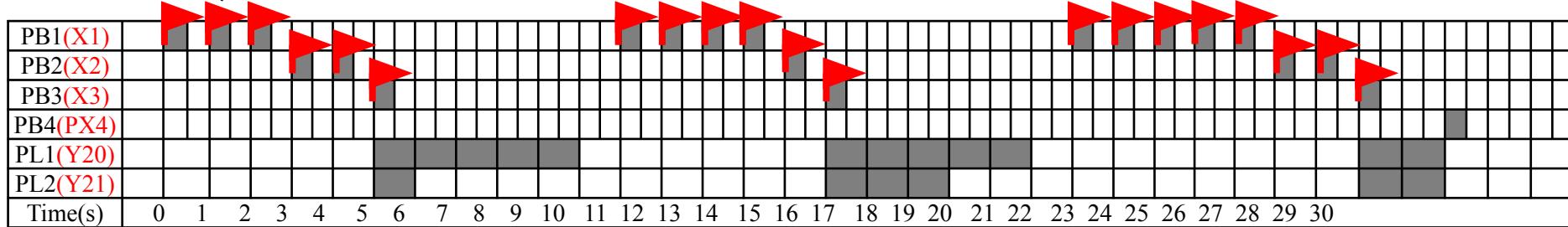
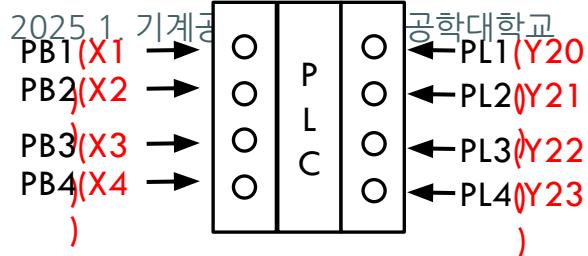
디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D20	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	725
D21	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	4104
D22	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	11
D23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D30	1	1	0	1	1	0	0	0	0	0	0	0	1	-10239			
D31	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1		
D32	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	

디바이스	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D20	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	268960469
D21	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
D22	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	11
D23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
D30	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	120833
D31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
D32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2

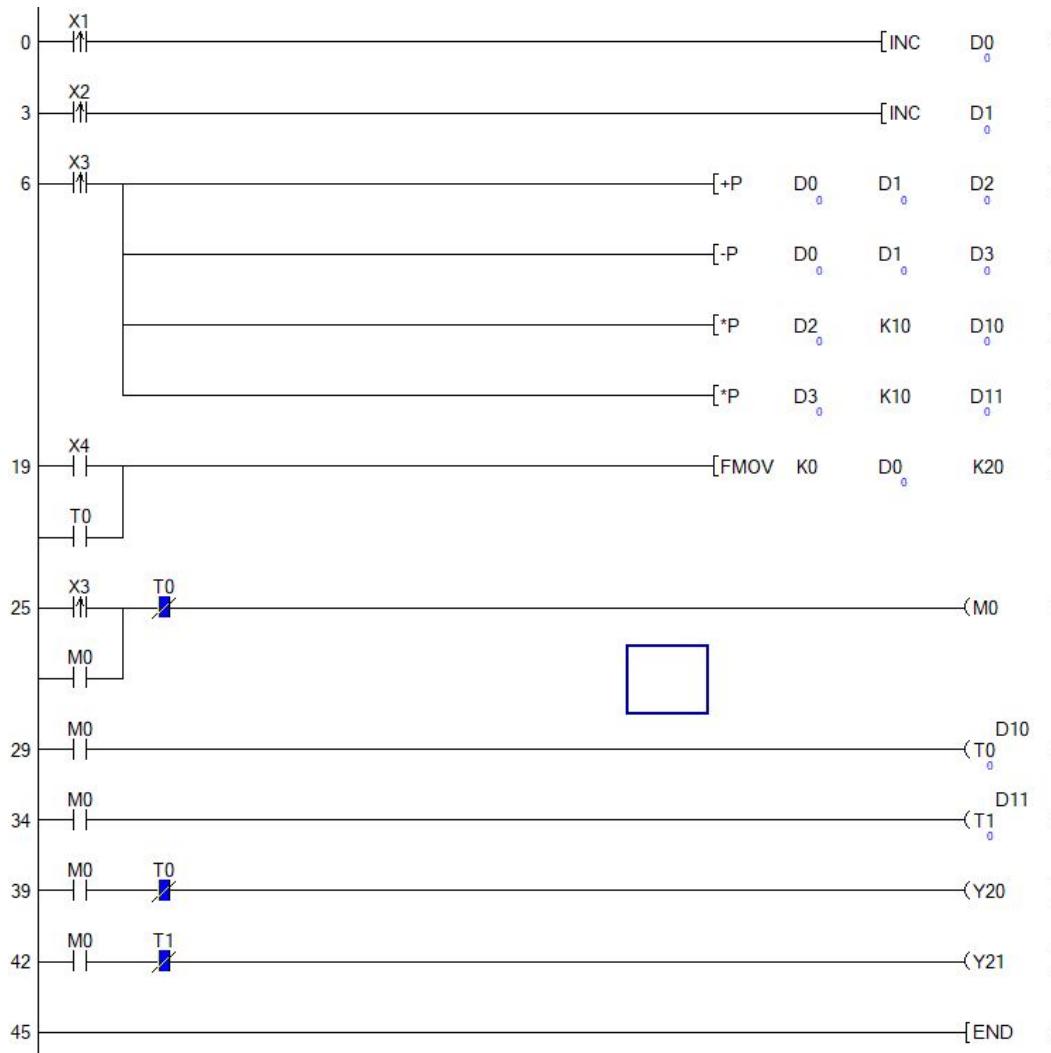
프로그램 연습1 ; 산술연산 명령

요구사항

1. PB1의 입력과 PB2의 입력 카운터는 PB3의 입력에 의해 출력된다.
2. PL1은 PL2의 출력은 PB1, PB2의 입력 횟수에 의해 연산된 값에 의해 결정된다.
3. PB4는 PB1, PB2의 카운터를 리셋 시킨다.



제4장 : 응용명령 사용하기



제4장 : 응용명령 사용하기

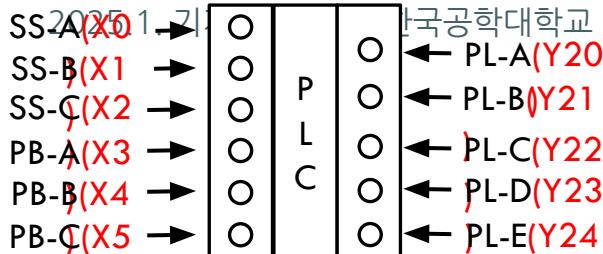
프로그램 연습2 ; 산술연산 명령

요구사항

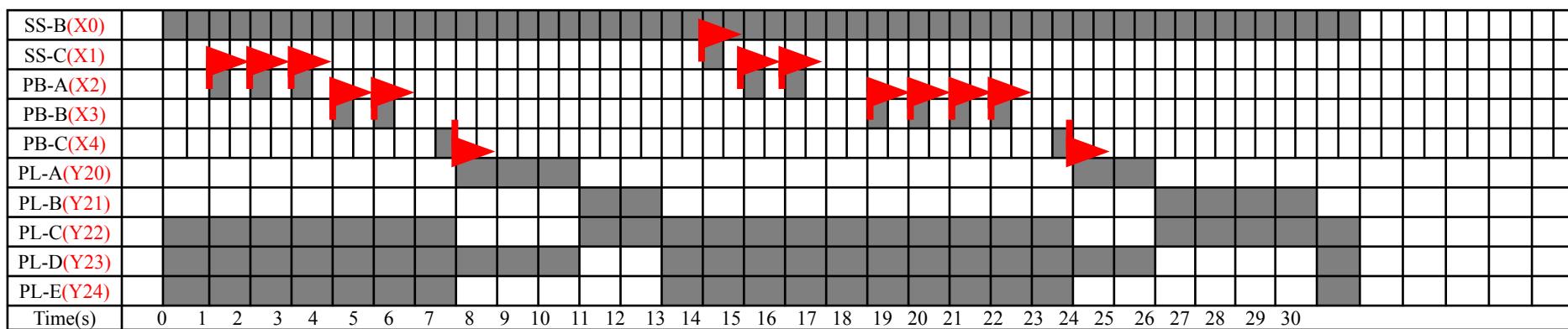
PB-A와 PB-B가 입력된 후 PB-C가 입력되면 아래와 같이 출력된다.

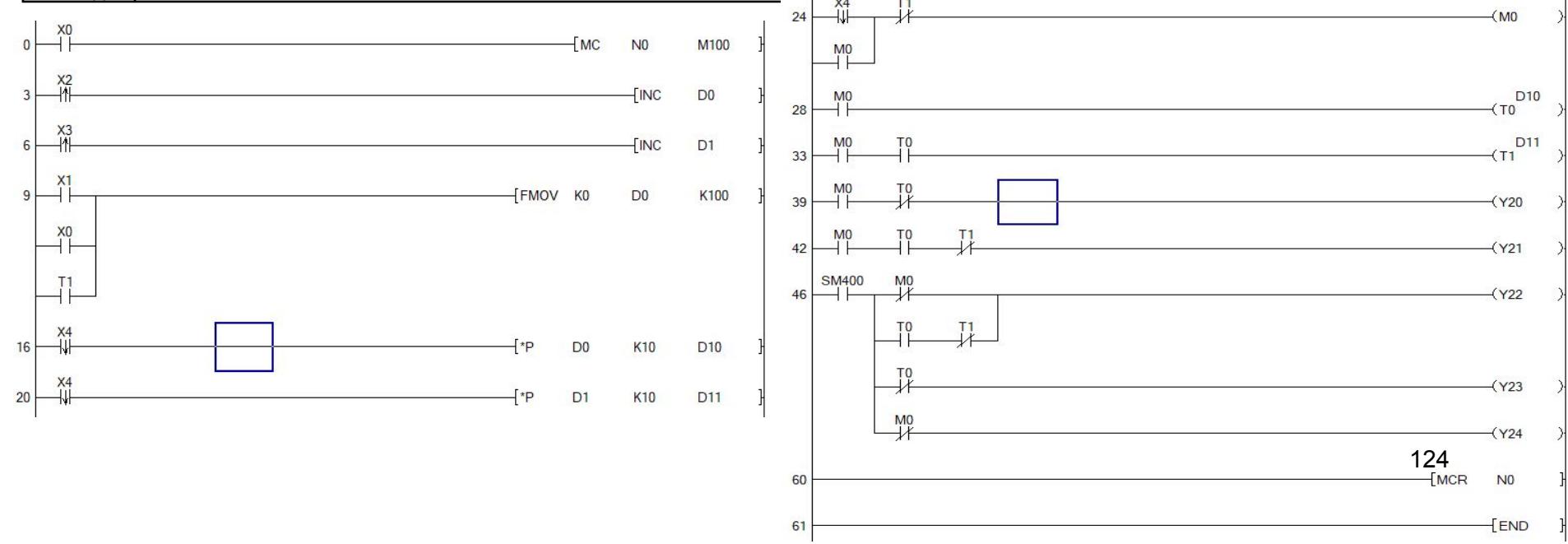
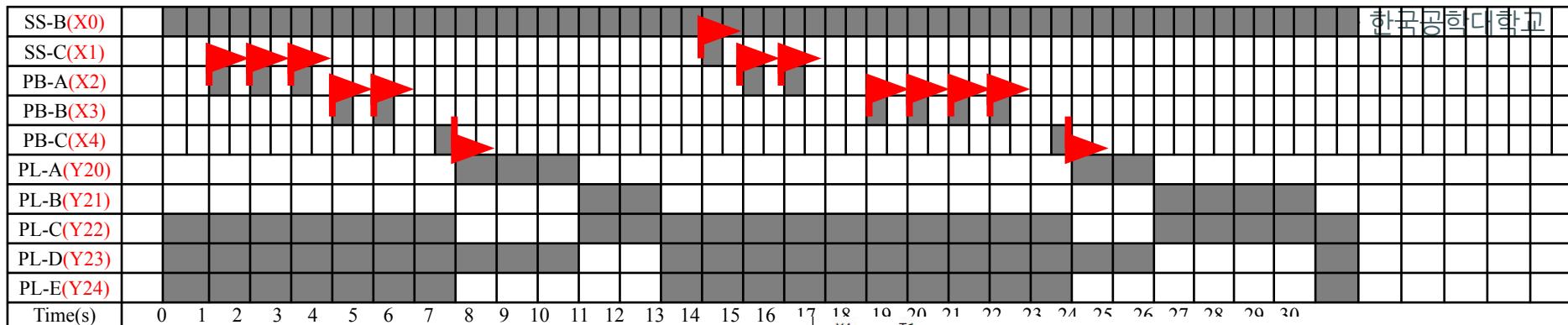
단, PB-A와 PB-B가 입력된 횟수 만큼 PL-A가 시간으로 변경되어 점등되고 이어 PL-B가 점등된다.

입력된 횟수 * 1초 [PL-C는 PL-A], [PL-D는 PL-B], [PL-E는 PLC “PL-D”]와 같이 동작한다. SS-C는 입력 초기화 버튼



)





프로그램 연습3 ; 산술연산 명령

1. SS-A(P0) A모드

2. SS-B(PA) B모드

3. P2 입력시 카운터 값이 증가하여 같은 초로 환산 한다.(카운터값 1= 타이머 1초)

4. 스타트 점등

5. 스타트 소등

6. 리셋조건 : A모드 P0 off 시 리셋, B모드 P1 off 시 리셋, P0, P1 값이 on 되면 리셋

