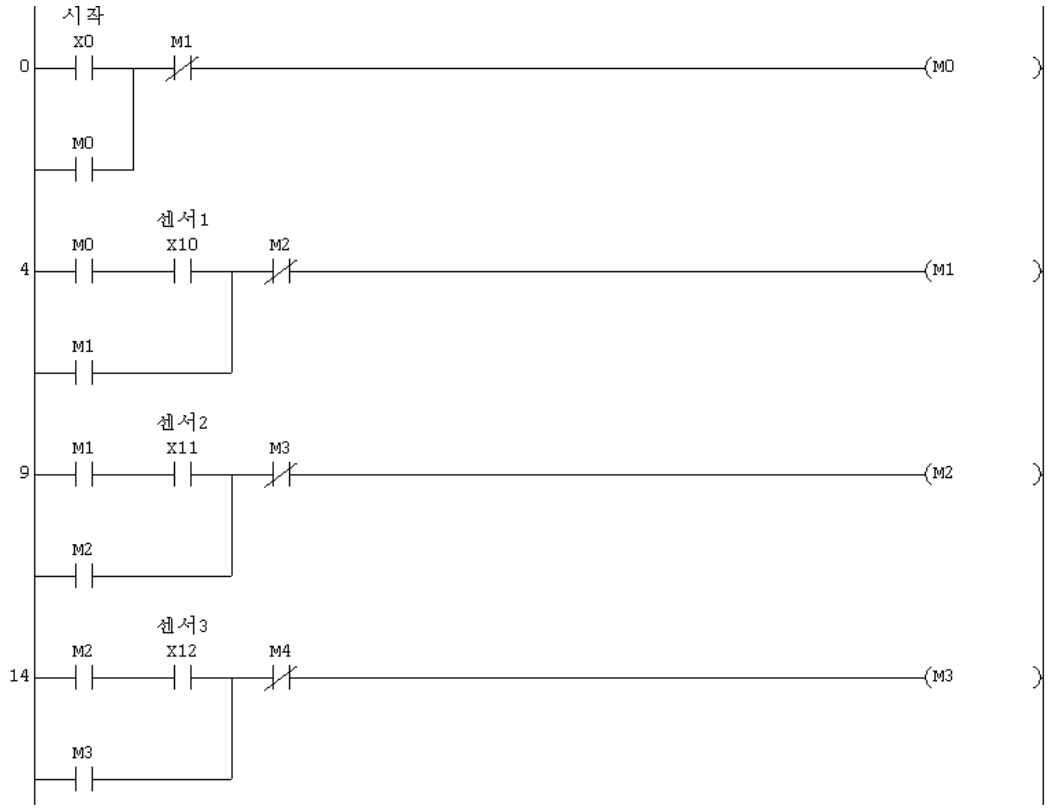
3. LD 프로그램의 작성

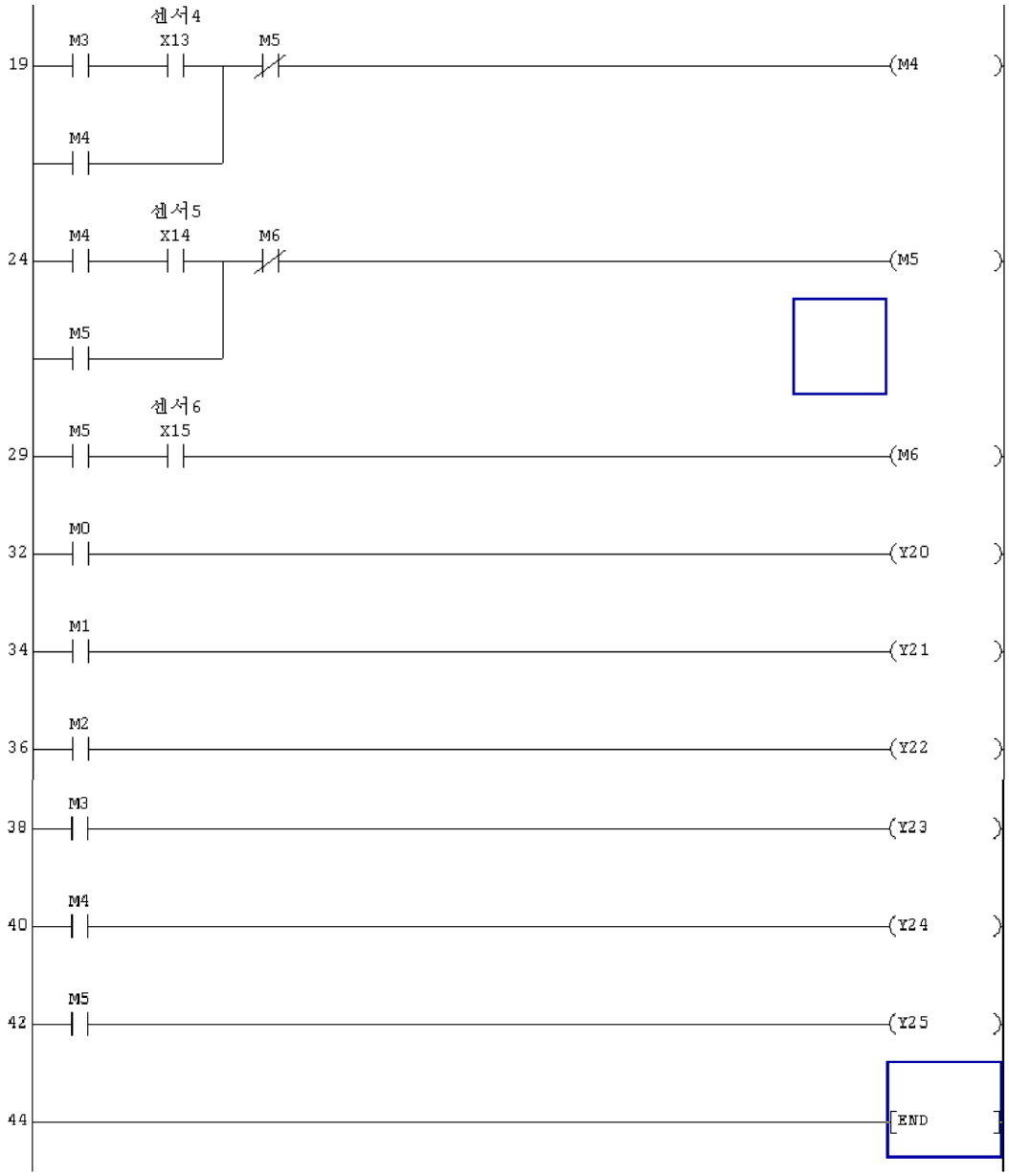
3.1 기본적인 LD프로그램의 작성

3.1.1 기본적인 시퀀스 LD 프로그램

Timing chart →						
Y20	■					
Y21		■				
Y22			■			
Y23				■		
Y24					■	
Y25						■

< Timing chart >





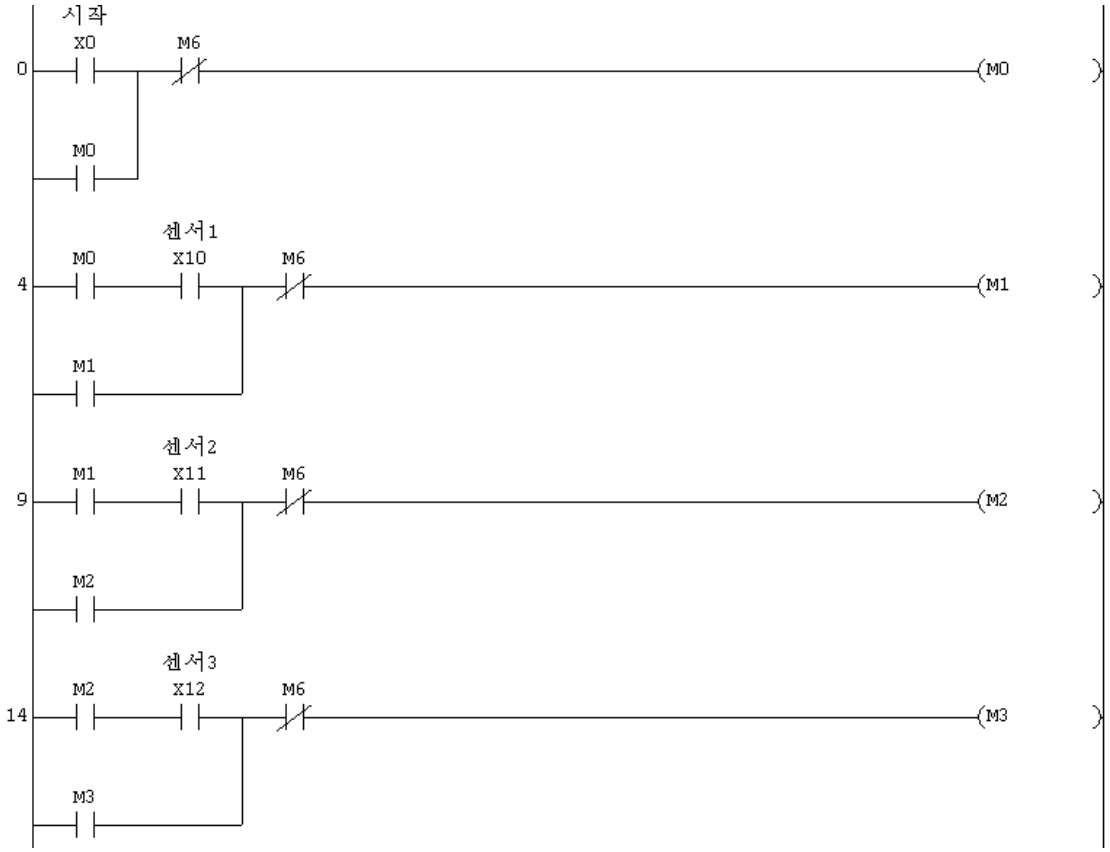
< 기본적인 시퀀스 프로그램 >

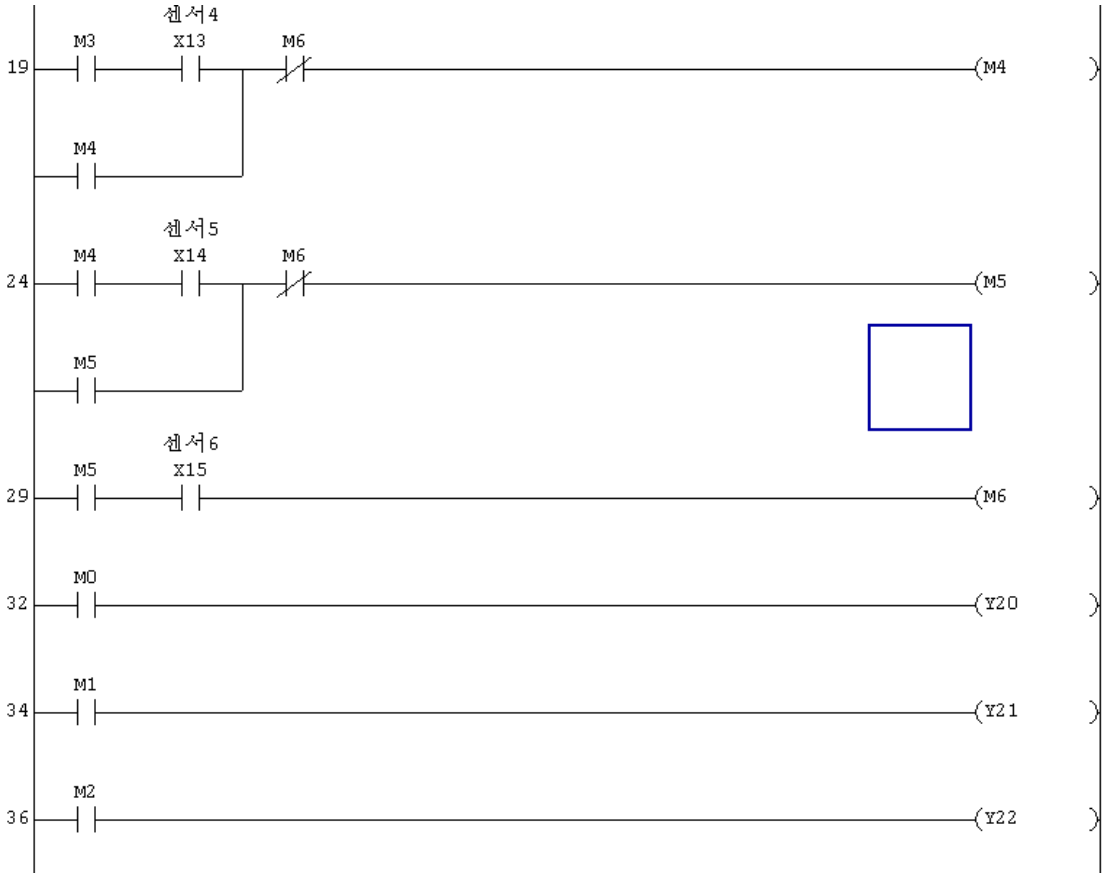
위 프로그램은 시퀀스 프로그램의 전형으로 M0부터 M5까지 순차적으로 ON-OFF하는 프로그램입니다. 보조릴레이 M을 이용하여 출력모듈의 Y20부터 Y25까지 순차적으로 값을 내보냄으로써 타이밍 챠트와 동일한 동작을 하도록 프로그램 하였습니다.

위 동작과는 다르게 출력값(Y20~Y25)을 유지한채 시퀀스 동작의 맨 마지막에서 OFF 할 수 있습니다.

Timing chart →						
Y20	■	■	■	■	■	■
Y21		■	■	■	■	■
Y22			■	■	■	■
Y23				■	■	■
Y24					■	■
Y25						■

< Timing chart >

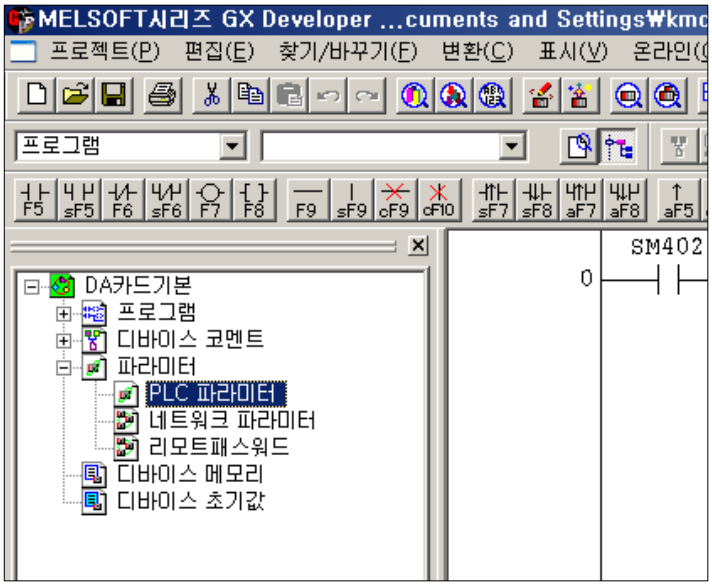




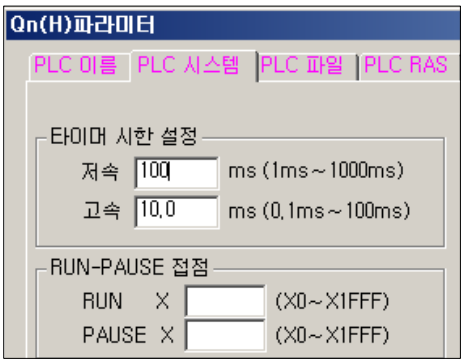
< 기본적인 시퀀스 프로그램 >

3.2 특수 디바이스를 이용한 순차 프로그램

3.2.1 타이머 디바이스를 이용한 순차 프로그램



< 타이머 디바이스 설정 변경 >



< 저속 및 고속 타이머 시간 설정 >

파라미터에서 타이머 시한 설정을 어떤 값을 하느냐에 따라 타이머의 값이 달라집니다. 저속타이머 (Txx ~) 는 1ms~1000ms 까지 설정이 가능하며, 고속타이머 (Hxx ~)는 0.1ms~100ms 까지 설정이 가능합니다. 여기서 K 는 정수를 의미합니다.

Timing chart →												
Y20		500 ms		500 ms		500 ms		500 ms		500 ms		
Y21												
Y22												
Y23												
Y24												
Y25												

< Timing chart >

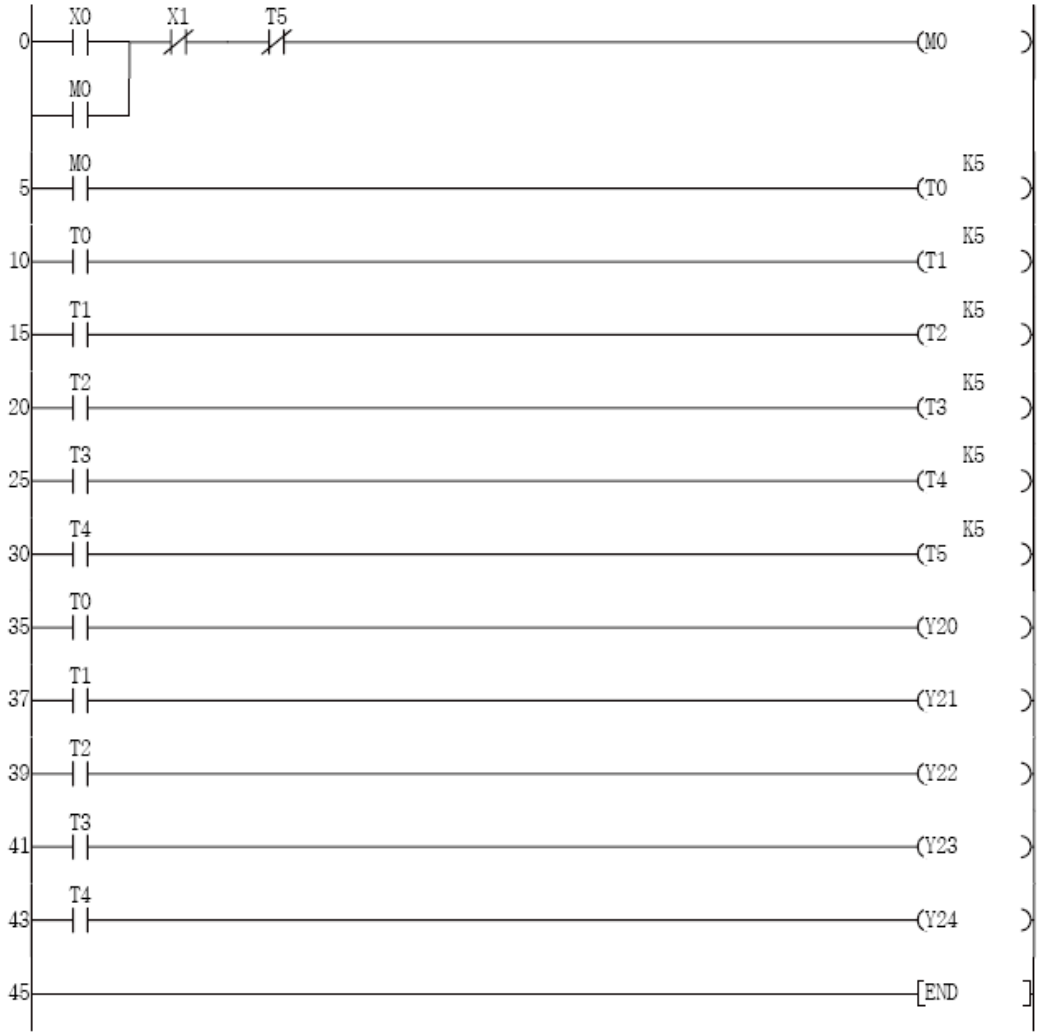
위 동작은 출력이 나간 후 0.5초 후에 다음 출력을 내보내는 시퀀스 프로그램을 작성하기 위한 타이밍 챠트이다. 타이머의 종류는 앞서 설명한 것과 같이 저속과 고속 타이머로 나뉘며 입력 방법은 아래와 같다.

< 타이머 디바이스 입력 방법, 코일임에 주의 할 것 >

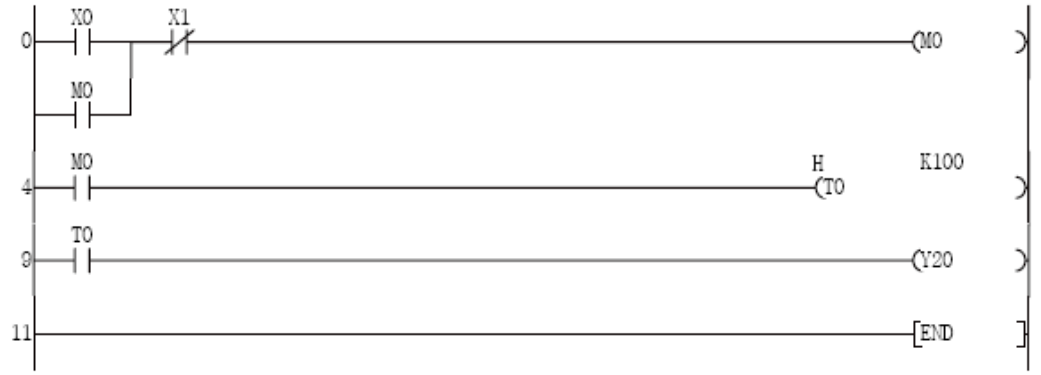
타이머는 출력 코일임에 주의(함수가 아님)해야 하며, T0~T2047까지 사용이 가능하다.

< 고속타이머 입력 방법 >

고속타이머의 경우는 타이머 앞에 'H'가 붙는다. 이렇게 입력을 함으로써 저속타이머와 구분을 지을 수 있다. 고속타이머를 사용하더라도 타이머는 T2047까지 사용이 가능합니다.



< 타이머를 이용한 LD프로그램 >

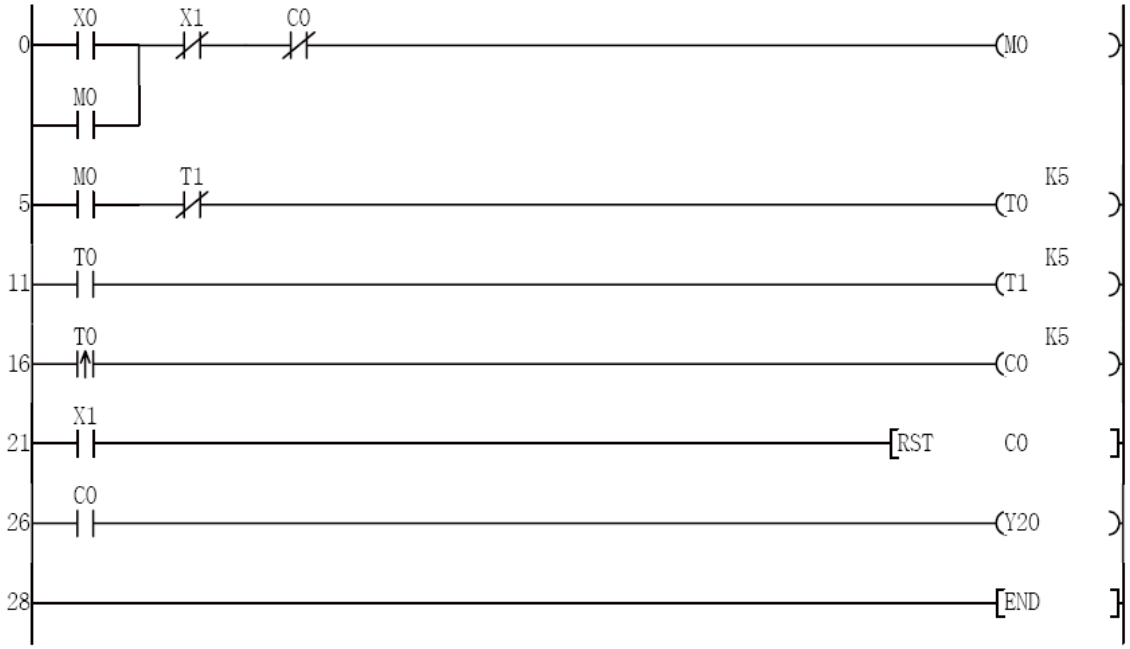


< 고속타이머의 사용 예 >

X0 입력 1초 후 (100 * 10ms = 1sec) Y20이 ON 됩니다.
(파라미터에서 고속타이머의 설정값을 10ms로 했음을 기억하라)

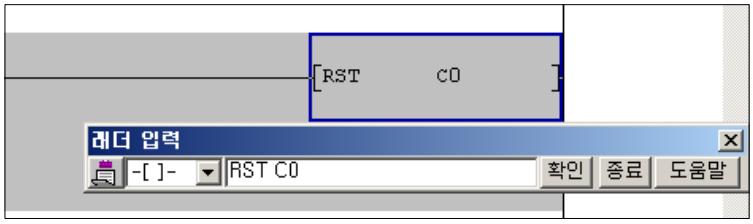
3.2.2 카운터 디바이스를 이용한 순차 프로그램

일반 카운터(C) 는 1024개 까지 사용이 가능합니다. 즉, C0 ~ C1023까지 카운터로 사용할 수 있습니다.

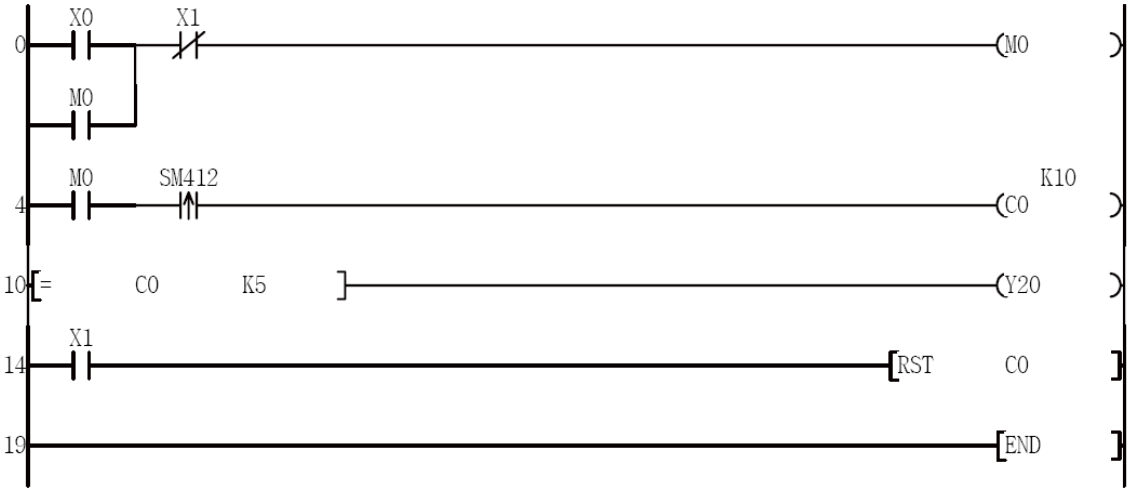


< 카운터 사용예 >

위 프로그램은 타이머를 이용하여 1초마다 자동 카운트를 하고, 카운트 횟수가 5회가 되면 Y20으로 출력을 내보내는 프로그램입니다. 단, 카운터(C)는 리셋으로 초기화하기 전까지는 접점 및 현재 카운팅 값을 유지합니다. 아래는 카운터를 리셋하기 위해 RST 함수 ([ ]) 사용하는 방법을 나타내었습니다.



< RST 함수 사용 예 >



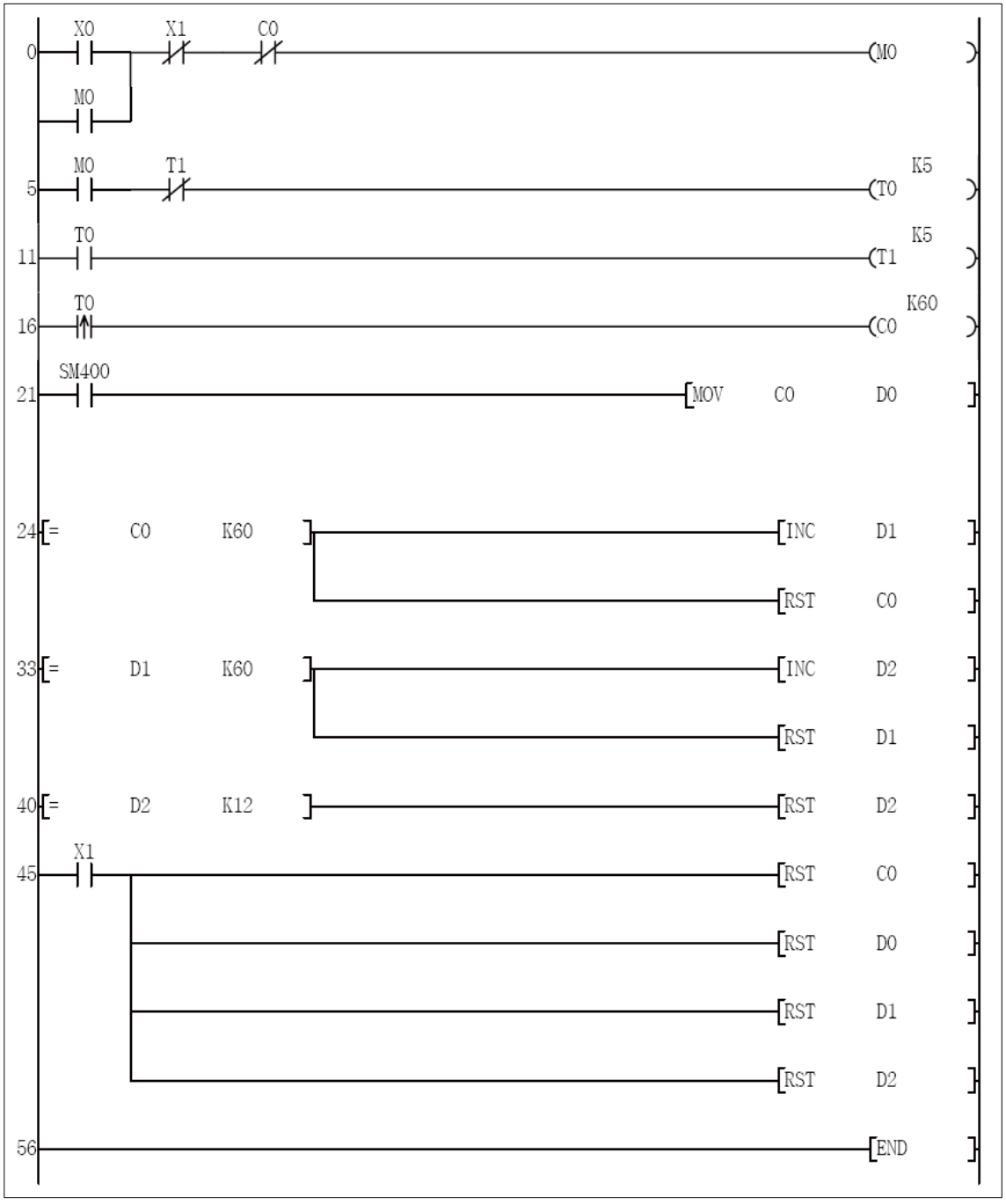
< 특수 릴레이를이용한 내부 카운터 >

타이머를 사용하지 않고 위와 같이 특수릴레이(SM412, 1초 클럭)를 사용하여 카운팅을 할 수 있습니다. 특수 릴레이는 아래와 같으며 더 자세한 사항은 미쯔비시 공통명령편을 이용하시기 바랍니다.

SM409	0.01초 클록	0.005초 0.005초	• 5ms마다 ON/OFF를 반복한다. • 전원 OFF 또는 리세트 시에는 OFF에서 시작한다.
SM410	0.1초 클록	0.05초 0.05초	• 지정 시간마다 ON/OFF를 반복한다. • 전원 OFF 또는 리세트 시에는 OFF에서 시작한다. * 프로그램 실행 도중에도 지정 시간이 되면 ON/OFF 상태가 변하므로 주의하십시오.
SM411	0.2초 클록	0.1초 0.1초	
SM412	1초 클록	0.5초 0.5초	
SM413	2초 클록	1초 1초	
SM414	2n초 클록	n초 n초	• SD414에서 지정한 시간(초)에 따라 ON/OFF를 반복한다.
SM415	2n(ms) 클록	n(ms) n(ms)	• SD415에서 지정한 시간(ms)에 따라 ON/OFF를 반복한다.

< 특수 릴레이, 클럭발생 특수릴레이 >

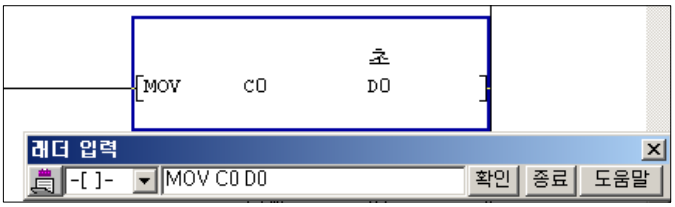
3.2.3 카운터, 타이머, 비교명령어를 이용한 전자시계



< 전자시계 프로그래밍 >

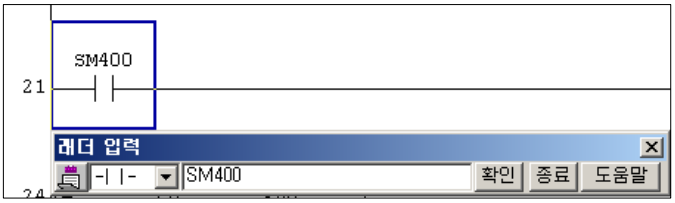
위 프로그램은 D0, D1, D2에 각각 초, 분, 시를 저장하는 프로그램입니다. X0를 이용하여 시계를 구동하며, X1을 이용하여 데이터를 리셋합니다.

위 프로그램을 코딩중 사용된 함수를 사용하는 방법입니다.

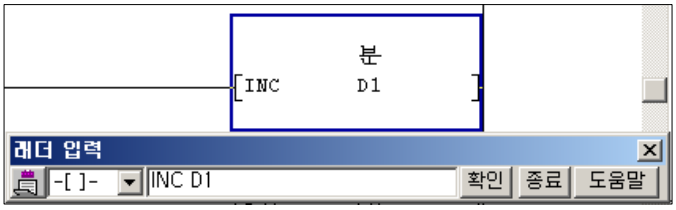


< 데이터 전송명령 C0 → D0 >

카운터 C0의 카운팅 값을 데이터레지스터 D0에 저장합니다.

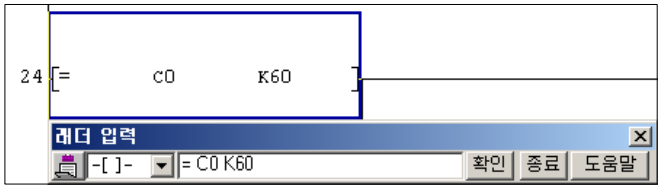


< 특수레지스터, 항상 ON 접점 >



< 1 증가 명령, Increase >

D1의 데이터를 1씩 증가 시킵니다.



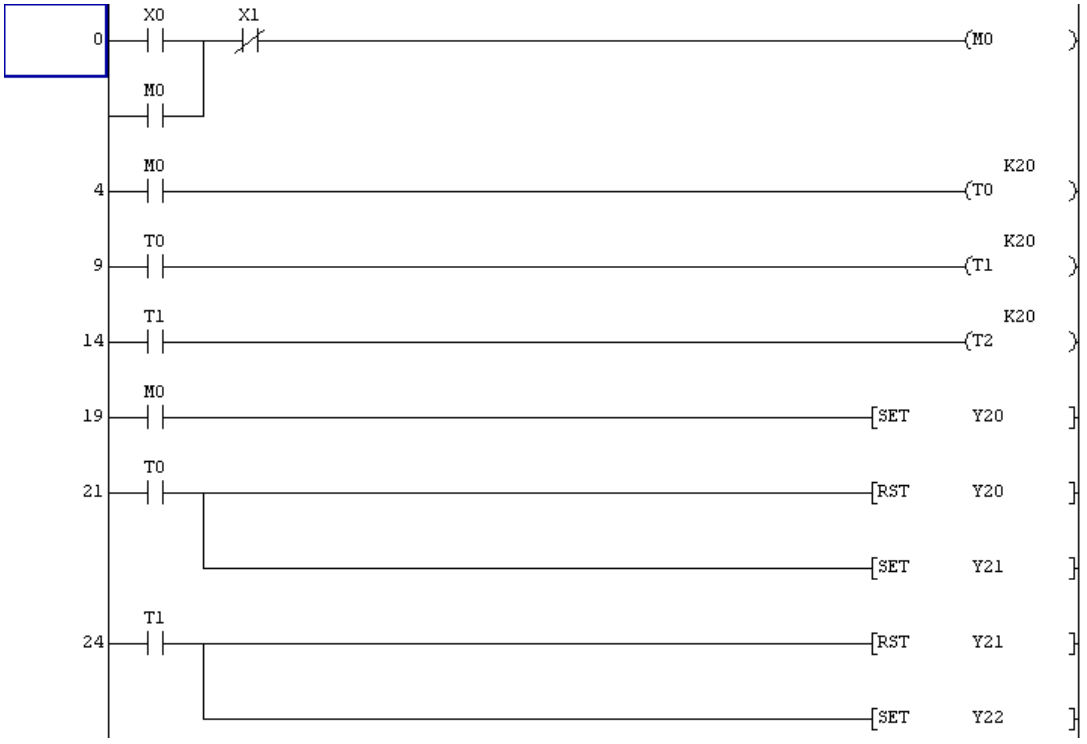
< 비교명령 '같다' >

C0의 값과 십진수(K)60이 같으면 출력으로 1을 내보냅니다.

3.2.4 SET, RESET 명령어를 이용한 순차프로그램

<SET RST명령 사용>

X0를 ON하면 Y20이 계속 ON출력을 내보냅니다. X1을 ON하면 Y20이 OFF됩니다. 별도의 자기유지가 필요 없습니다.



<타이머와 SET, RST 명령을 함께 사용된 예제>

X0가 ON되면 바로 Y20이 ON됩니다.
2초후에 Y20은 OFF되며, Y21이 ON됩니다.
2초후에 Y21은 OFF되며, Y22가 ON됩니다.
2초후에 Y22는 OFF됩니다.
X1을 누르면 Y20~Y22까지 OFF 됩니다.

3.3 신호등 프로그램 작성

3.3.1 신호등 상태값 연산

우선 신호등 프로그램을 작성하기 위해 아래와 같은 상태값을 정의합니다. 신호등은 변환되는 순서가 이미 정해져 있기 때문에, 계속 연산하여 결과값을 처리하는 것 보다는 처리된 결과값을 기억하고 있다가 출력하는 것이 유리합니다.

그렇기 때문에 아래와 같이 일반적인 사거리에서 일어 날 수 있는 상황을 구성하여 출력어드레스에 매칭하여 출력하도록 하겠습니다.

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

<십자 교차로에 대한 신호등 상태값>

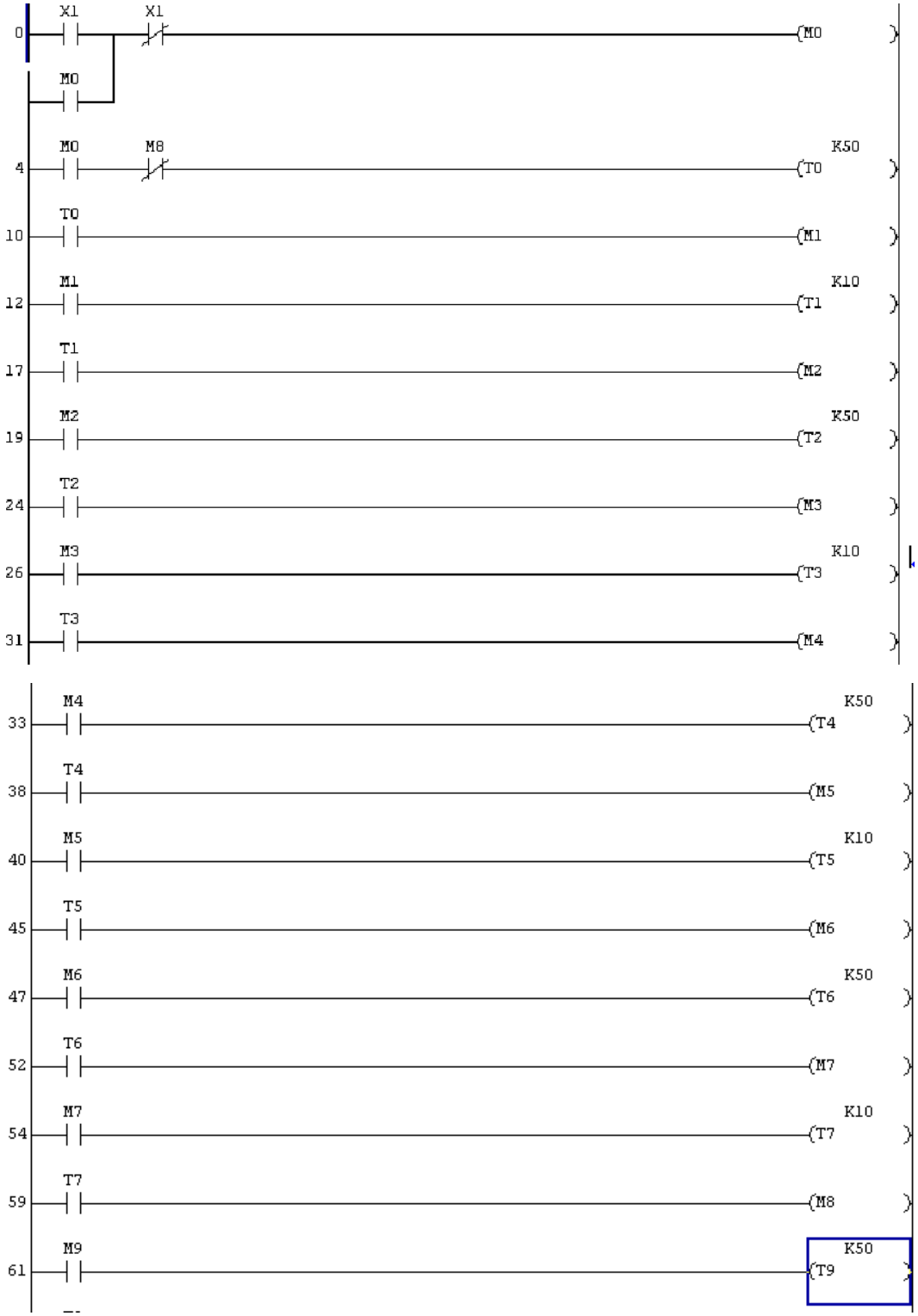
위 표에서 처음 5초간은 1차로의 보행자가 녹색이며, 2차로의 자동차만이 주행하비다. 모든 차로는 직좌동시 신호이며, 5초는 주행 및 보행 시간, 1초는 주황 변경시간에 해당합니다.(시간은 실제 상황과 상이합니다.)
맨 첫줄 (회색) 의 경우를 비트값으로 계산하면(아래서 위 방향으로), 아래와 같이 되며 16진수 변환값이 그 옆에 표기되어 있습니다.

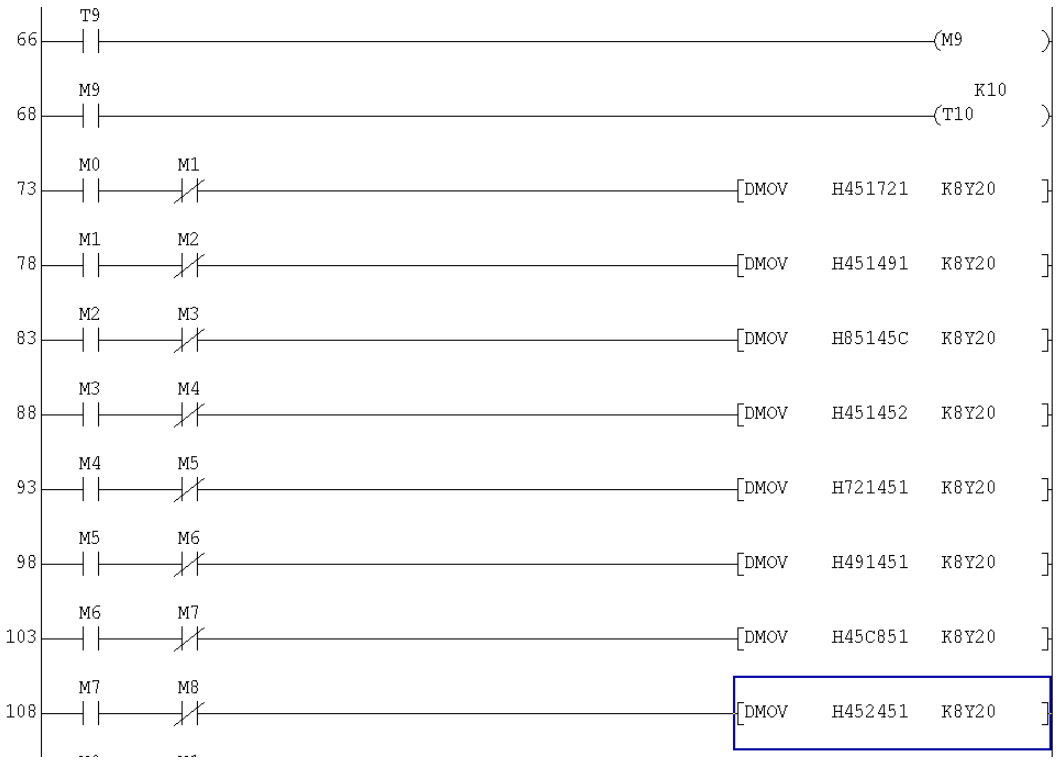
- 각 시점에서의 신호등 출력값

D0	010001010001011100100001	H451721
D1	010001010001010010010001	H451491
D2	100001010001010001011100	H85145C
D3	010001010001010001010010	H451452
D4	011100100001010001010001	H721451
D5	010010010001010001010001	H491451
D6	010001011100100001010001	H45C851
D7	010001010010010001010001	H452451

이제 위와 같이 출력값의 상태가 정해지면 각 시간에 따라 위 값을 출력하면 신호등 시스템이 완성됩니다. 다음과 같이 LD 프로그램으로 구성할 수 있습니다.

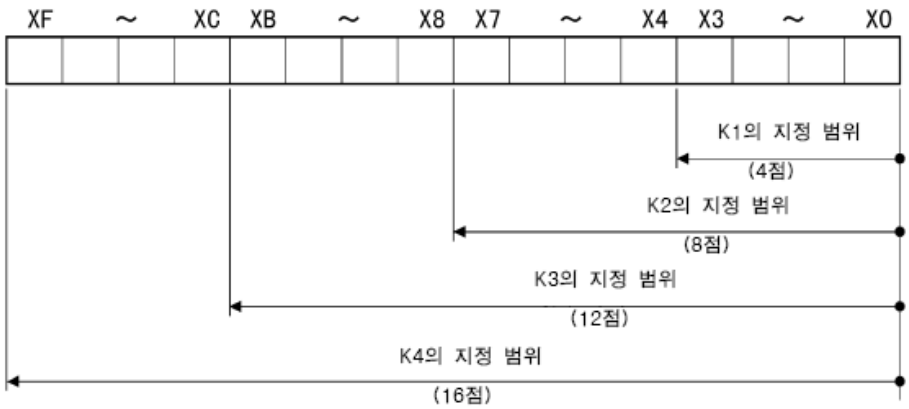
3.3.2 신호등 LD 프로그램





< 사거리 신호등시스템 >

위에서 K8Y20은 출력모듈 Y20부터 니블(4비트) 단위의 출력을 의미합니다. 즉 Y20 ~ Y3f까지 출력을 의미합니다.



<16비트 명령시의 자리 지정 설정범위>

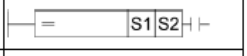
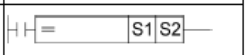
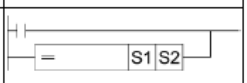
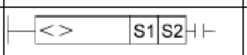
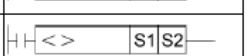
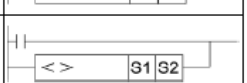
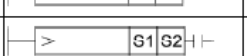
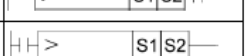
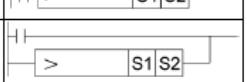
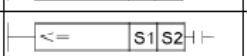
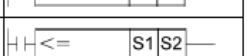
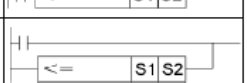
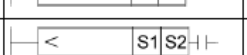
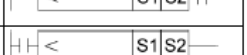
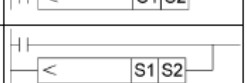
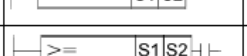
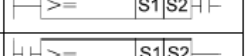
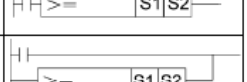
지정 자리 수	16비트 명령시
K1(4점)	0~15
K2(8점)	0~255
K3(12점)	0~4095
K4(16점)	-32768~32767

<자리 지정으로 취급할 수 있는 수치의 일람>

3.4 기타 명령어 사용법

데스티네이션(D, destination)…연산 후의 데이터의 목적지를 표시합니다.
소스(S, source)……연산 전의 데이터를 저장합니다.

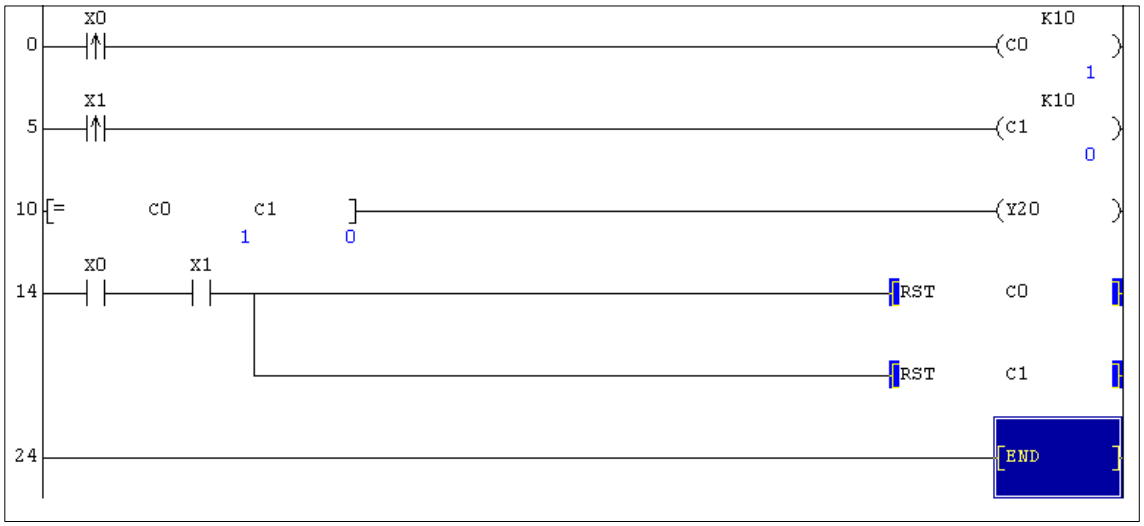
3.4.1 비교명령어

분류	명령 기호	심볼	처리 내용
16비트 데이터 비교	LD=		• (S1)=(S2)일 때 ON 상태 • (S1)≠(S2)일 때 OFF 상태
	AND=		
	OR=		
	LD<>		• (S1)≠(S2)일 때 ON 상태 • (S1)=(S2)일 때 OFF 상태
	AND<>		
	OR<>		
	LD>		• (S1)>(S2)일 때 ON 상태 • (S1)≤(S2)일 때 OFF 상태
	AND>		
	OR>		
	LD<=		• (S1)≤(S2)일 때 ON 상태 • (S1)>(S2)일 때 OFF 상태
	AND<=		
	OR<=		
	LD<		• (S1)<(S2)일 때 ON 상태 • (S1)≥(S2)일 때 OFF 상태
	AND<		
	OR<		
	LD>=		• (S1)≥(S2)일 때 ON 상태 • (S1)<(S2)일 때 OFF 상태
	AND>=		
	OR>=		

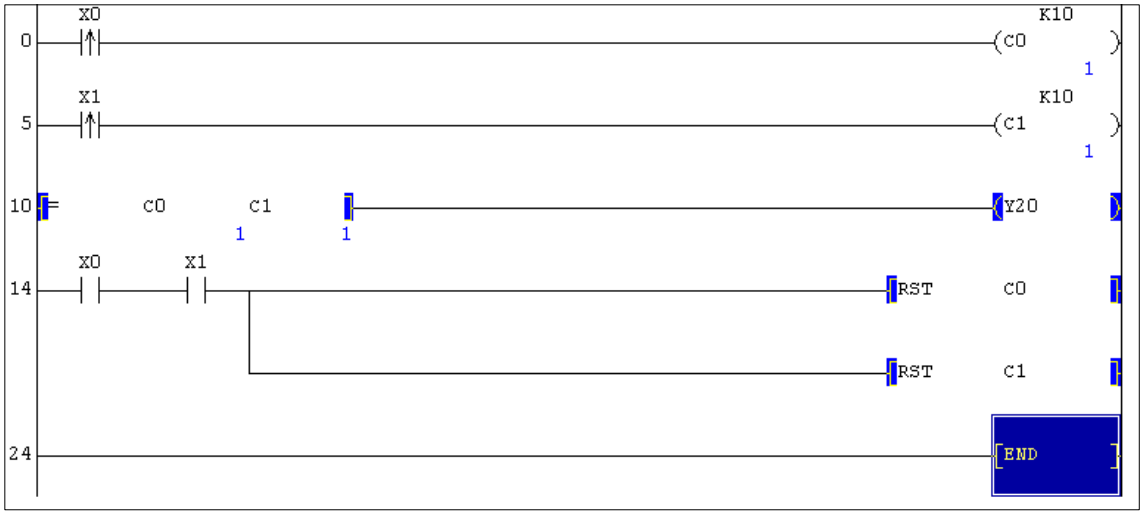
< 비교연산 명령 >

(1) 비교 명령어 '=' (같다)

< 함수 입력 방법 >

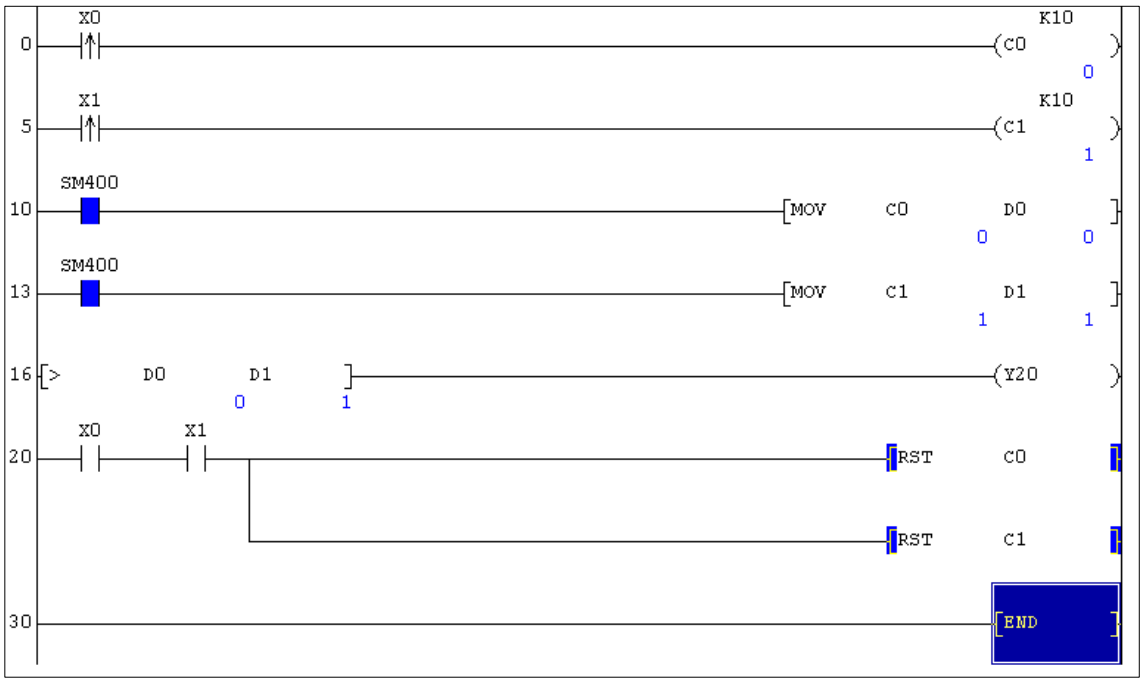


< '=' 같다 비교 예제, 값이 다를 때 >

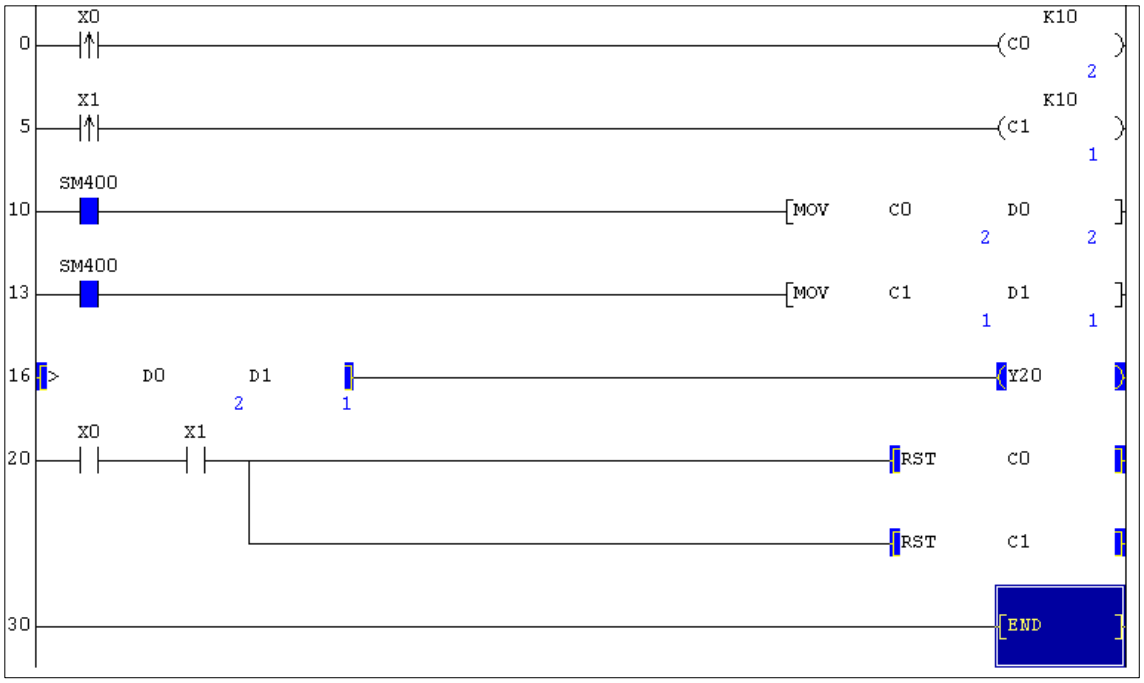


< '=' 같다 비교 예제, 값이 같을 때 >

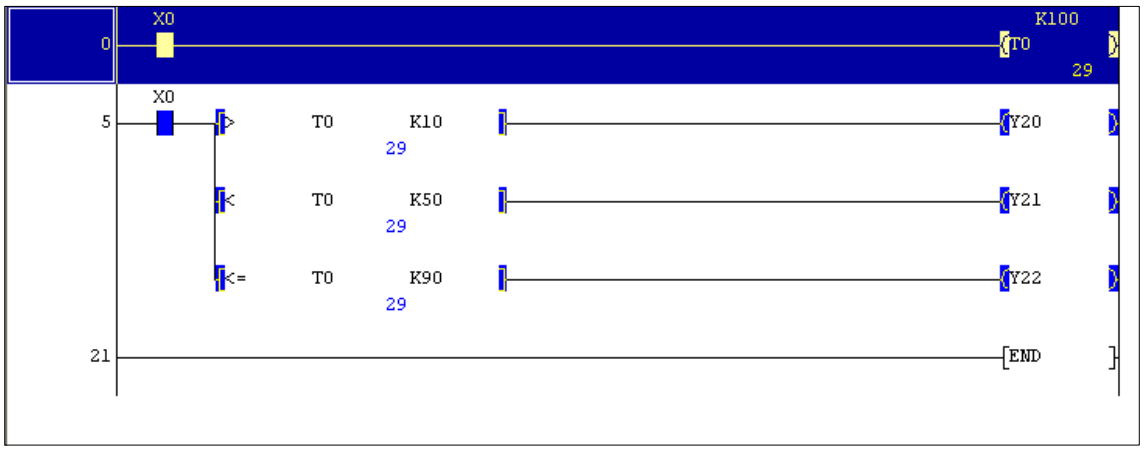
(2) 비교 명령어 '>' (크다)



< '>' 크다 비교 예제, D0가 D1보다 크지 않을 때 >

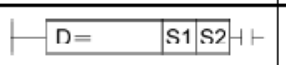
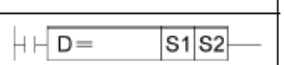
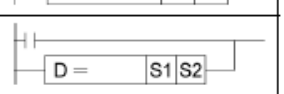
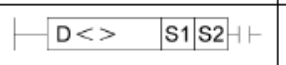
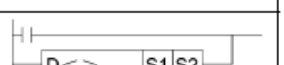
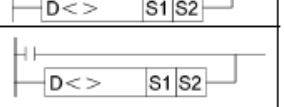
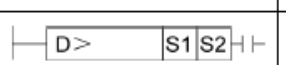
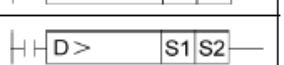
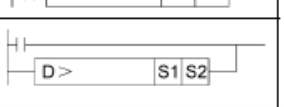
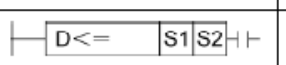
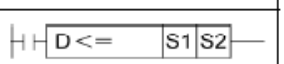
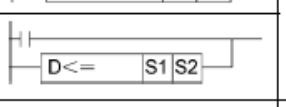
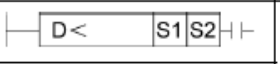
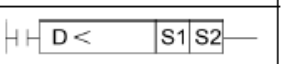
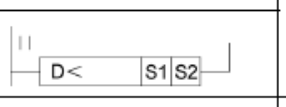
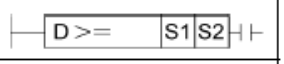
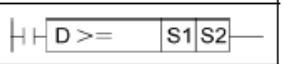
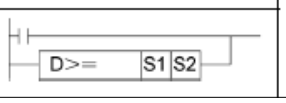


< '>' 크다 비교 예제, D0가 D1보다 클 때 >



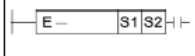
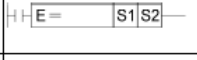
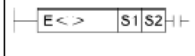
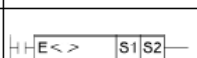
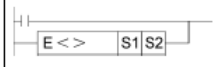
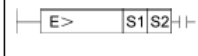
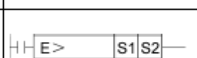
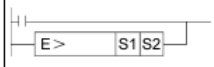
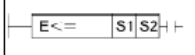
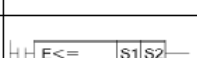
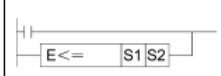
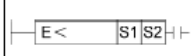
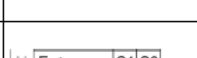
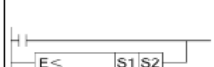
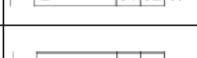
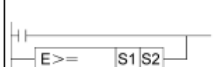
< 비교명령의 사용 >

(3) 32bit 및 실수 데이터의 비교

분류	명령 기호	심볼	처리 내용
32비트 데이터 비교	LDD=		・(S1)=(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)≠(S2+1, S2)일 때 OFF 상태
	ANDD=		
	ORD=		
	LDD<>		・(S1+1, S1)≠(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)=(S2+1, S2)일 때 OFF 상태
	ANDD<>		
	ORD<>		
	LDD>		・(S1+1, S1)>(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)≤(S2+1, S2)일 때 OFF 상태
	ANDD>		
	ORD>		
	LDD<=		・(S1+1, S1)≤(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)>(S2+1, S2)일 때 OFF 상태
	ANDD<=		
	ORD<=		
	LDD<		・(S1+1, S1)<(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)≥(S2+1, S2)일 때 OFF 상태
	ANDD<		
	ORD<		
	LDD>=		・(S1+1, S1)≥(S2+1, S2)일 때 ON 상태 ・(S1+1, S1)<(S2+1, S2)일 때 OFF 상태
	ANDD>=		
	ORD>=		

< 32비트 데이터 비교 함수 >

※ 문자열비교 및 블록데이터 비교는 명령어 집 참고할 것

분류	명령 기호	심볼	처리 내용
실수 데이터 비교	LDE=		• (S1+1, S1)=(S2+1, S2)일 때 ON 상태 • (S1+1, S1)≠(S2+1, S2)일 때 OFF 상태
	ANDE=		
	ORE=		
	LDE<>		• (S1+1, S1)≠(S2+1, S2)일 때 ON 상태 • (S1+1, S1)=(S2+1, S2)일 때 OFF 상태
	ANDE<>		
	ORE<>		
	LDE>		• (S1+1, S1)>(S2+1, S2)일 때 ON 상태 • (S1+1, S1)≤(S2+1, S2)일 때 OFF 상태
	ANDE>		
	ORE>		
	LDE<=		• (S1+1, S1)≤(S2+1, S2)일 때 ON 상태 • (S1+1, S1)>(S2+1, S2)일 때 OFF 상태
	ANDE<=		
	ORE<=		
	LDE<		• (S1+1, S1)<(S2+1, S2)일 때 ON 상태 • (S1+1, S1)≥(S2+1, S2)일 때 OFF 상태
	ANDE<		
	ORE<		
	LDE>=		• (S1+1, S1)≥(S2+1, S2)일 때 ON 상태 • (S1+1, S1)<(S2+1, S2)일 때 OFF 상태
	ANDE>=		
	ORE>=		

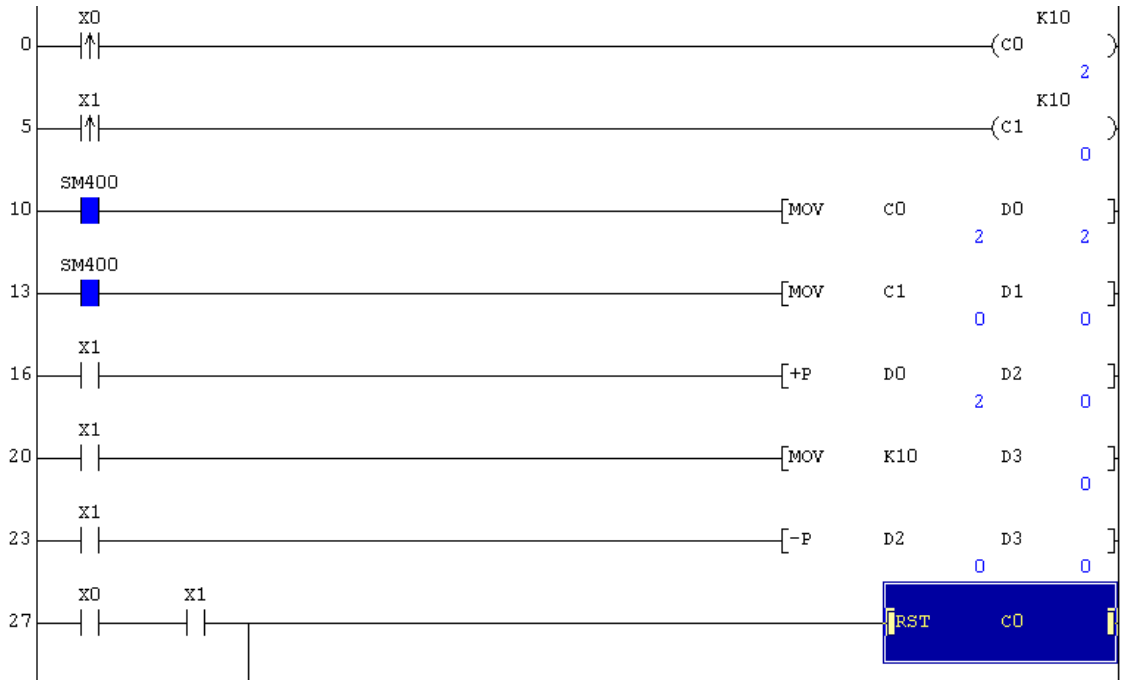
< 실수 데이터 비교 함수 >

3.4.2 산술명령어

분류	명령 기호	심볼	처리 내용
BIN 16비트 가감산	+	+, S, D	• (D)+(S) → (D)
	+P	+P, S, D	
	+	+, S1, S2, D	• (S1)+(S2)→(D)
	+P	+P, S1, S2, D	
	-	−, S, D	• (D)-(S)→(D)
	-P	−P, S, D	
	-	−, S1, S2, D	• (S1)-(S2)→(D)
	-P	−P, S1, S2, D	
BIN 32비트 가감산	D+	D+, S, D	• (D+1,D)+(S+1,S)→(D+1,D)
	D+P	D+P, S, D	
	D+	D+, S1, S2, D	• (S1+1,S1)+(S2+1,S2)→(D+1,D)
	D+P	D+P, S1, S2, D	
	D-	D−, S, D	• (D+1,D)-(S+1,S)→(D+1,D)
	D-P	D−P, S, D	
	D-	D−, S1, S2, D	• (S1+1,S1)-(S2+1,S2)→(D+1,D)
	D-P	D−P, S1, S2, D	
BIN 16비트 승제산	*	*, S1, S2, D	• (S1)*(S2)→(D+1,D)
	*P	*P, S1, S2, D	
	/	/, S1, S2, D	• (S1)/(S2)→몫(D), 나머지(D+1)
	/P	/P, S1, S2, D	
BIN 32비트 승제산	D*	D*, S1, S2, D	• (S1+1,S1)*(S2+1,S2) →(D+3,D+2,D+1,D)
	D*P	D*P, S1, S2, D	
	D/	D/, S1, S2, D	• (S1+1, S1)/(S2+1, S2) →몫(D+1, D), 나머지(D+3, D+2)
	D/P	D/P, S1, S2, D	

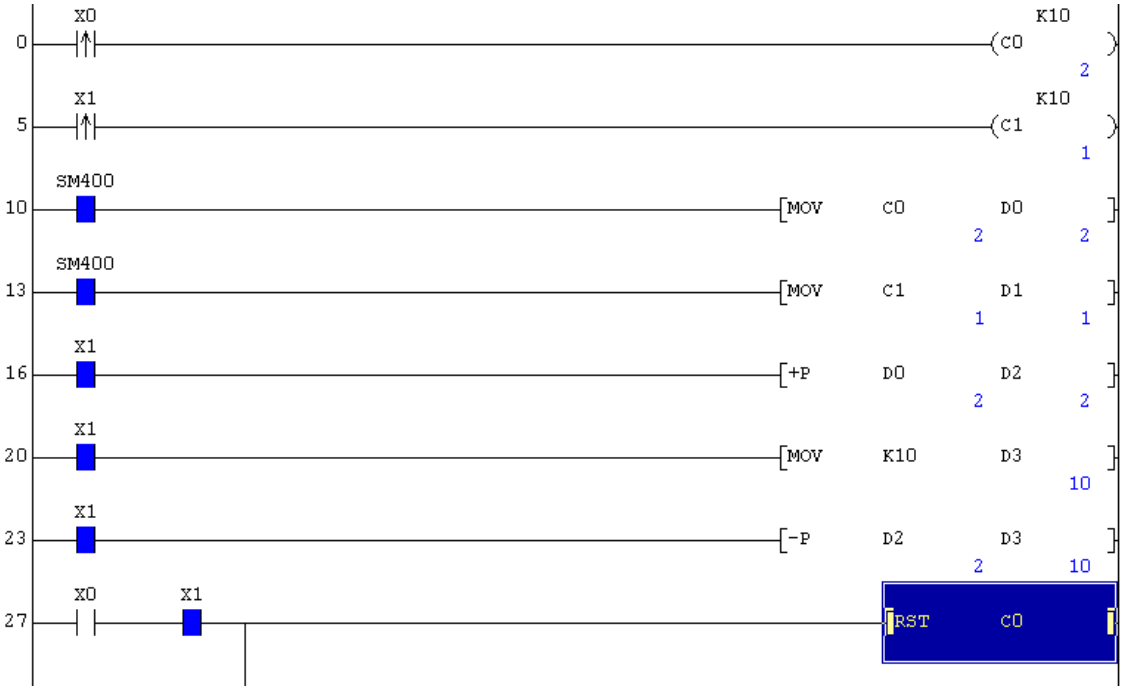
< 산술연산명령 >

(1) 산술명령어 '+ -' (덧셈, 뺄셈)

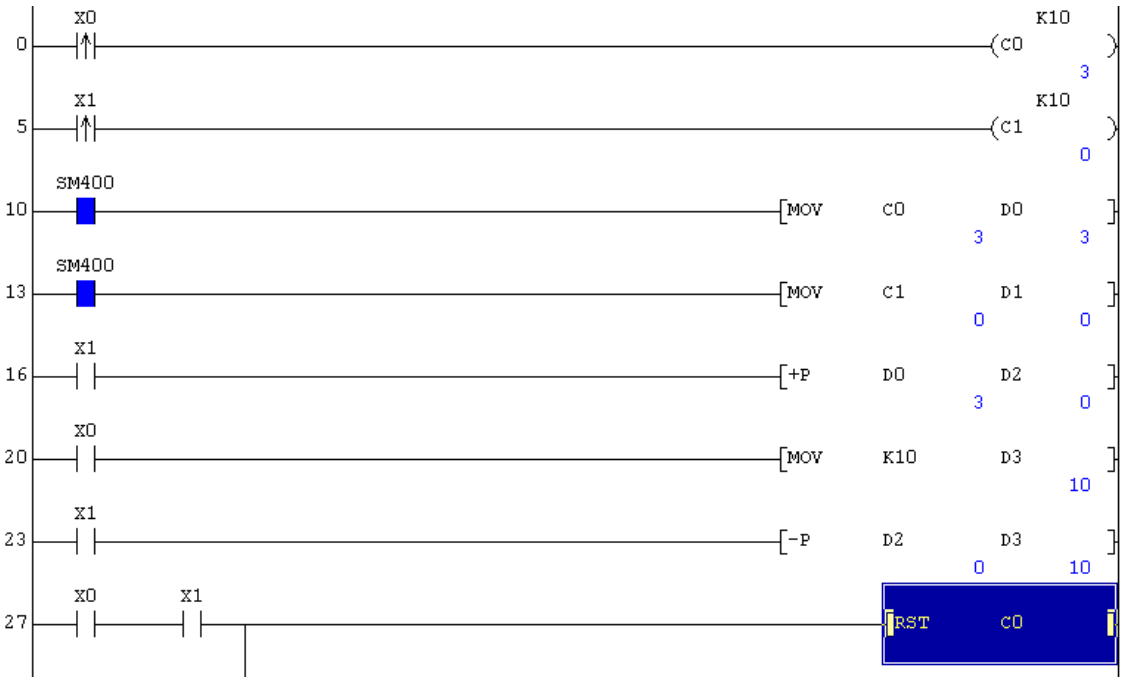


< 덧셈계산1 >

D0에 현재 카운트 값 '2' 가 저장 되어 있으며, D0+D2의 결과 값이 D2에 저장됩니다. (2+0 = 2 (D2))

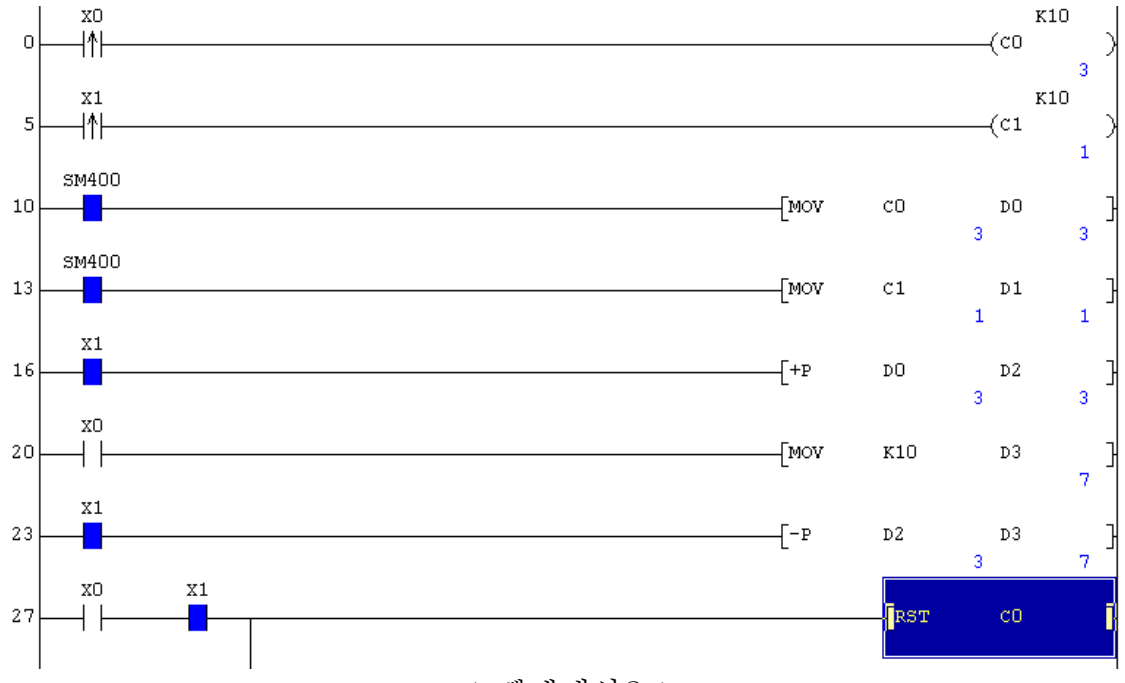


< 덧셈계산2 >



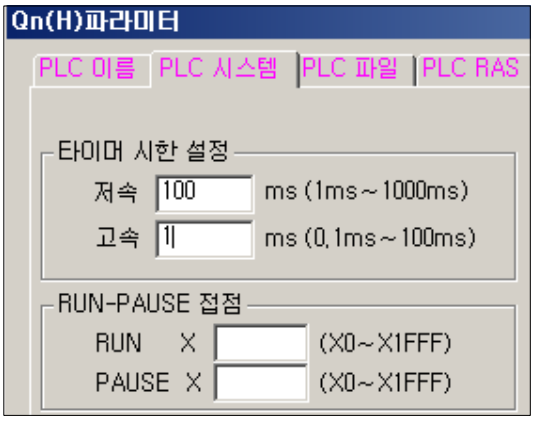
< 뺄셈계산1 >

D3에 현재 10이 저장되어 있고 D2에는 0값이 저장되어 있습니다. 이때 X1을 누르게 되면 D0에 저장되어 있는 3이 D2에 저장되며, D3-D2 = 7이 D3에 저장됩니다.

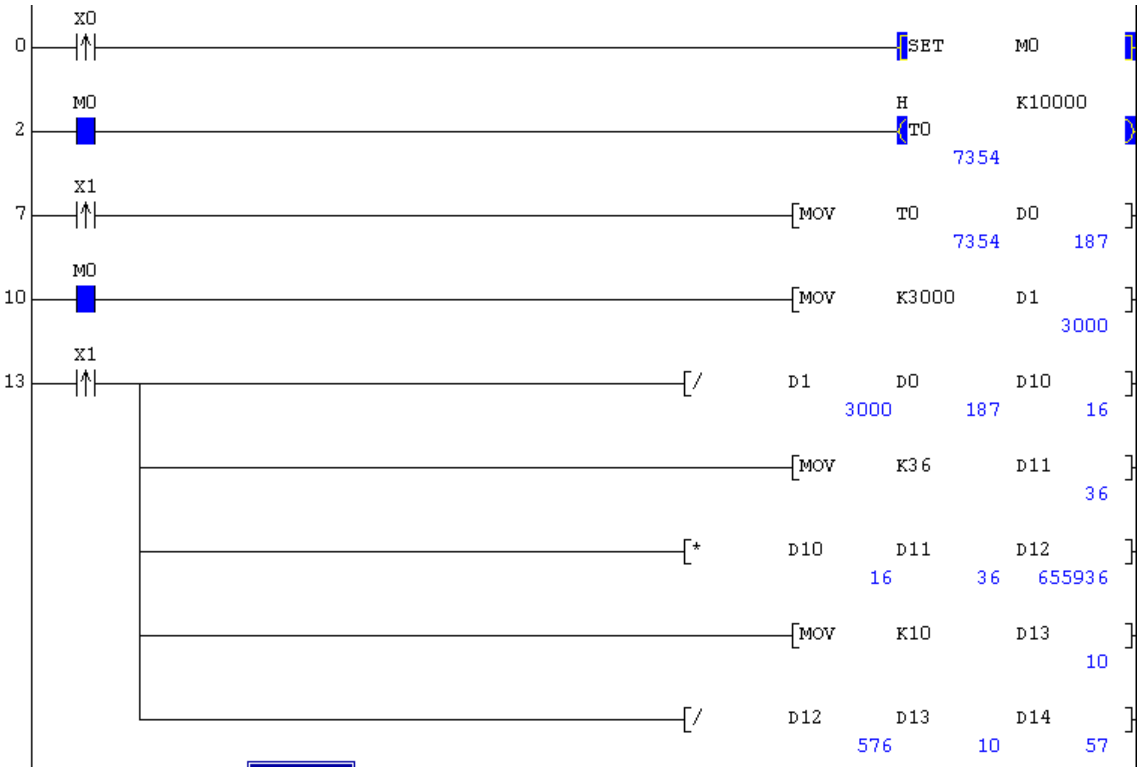


< 뺄셈계산2 >

(2) 산술명령어를 이용한 속도계 시스템 프로그램



< 정확한 계산을 위해 고속 타이머의 설정을 1ms로 설정 >



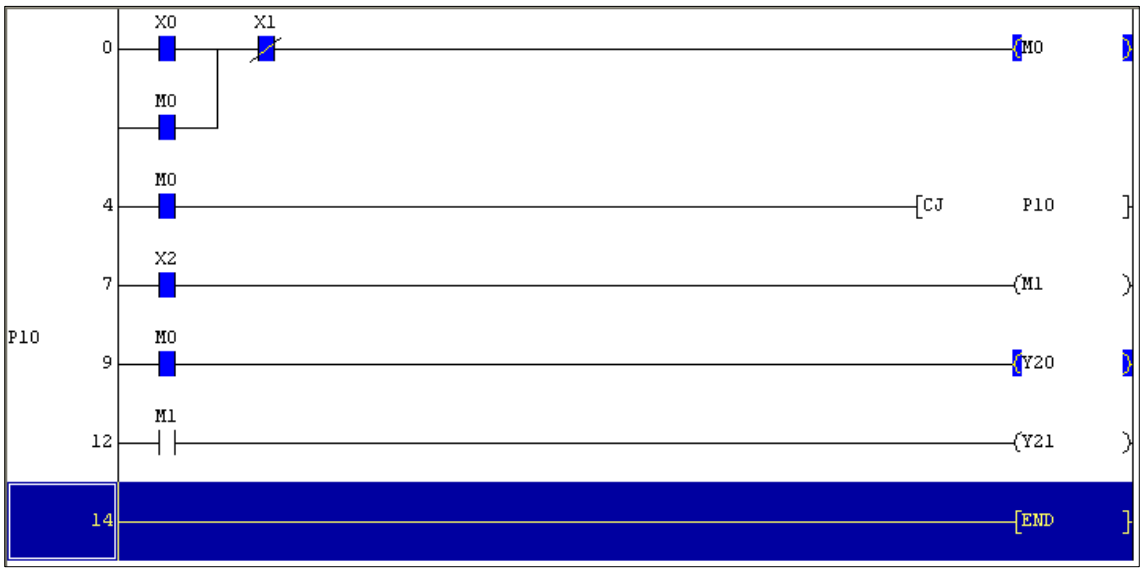
< 속도계 및 단위 변환 >

X0가 센서1, X1이 센서2이며, 센서간의 간격은 3m라고 가정합니다. D1에 3000mm 를 저장해 놓고, 두 센서에 들어온 시간을 D0에 ms 단위로 합니다. 3000mm/187ms의 단위를 km/h로 변환 하는 과정이 아래에 있으며, 그 결과 값이 D14에 저장되었습니다. 즉 57km/h가 됩니다.

3.4.3 분기명령

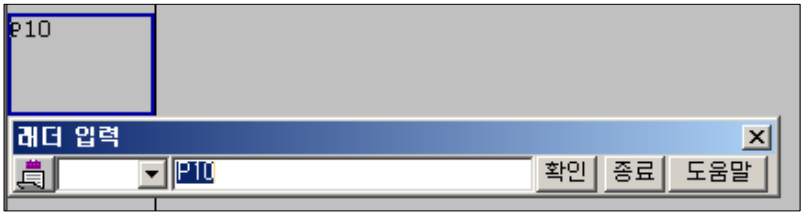
분류	명령 기호	심볼	처리 내용
점프	CJ	CJ Pn	· 입력 조건 성립시에 Pn으로 점프
	SCJ	SCJ Pn	· 입력 조건이 성립한 다음의 스캔에서 Pn으로 점프
	JMP	JMP Pn	· 무조건 Pn으로 점프
	GOEND	GOEND	· 입력 조건 성립시에 END 명령으로 점프

< 프로그램 분기명령 >



< CJ명령 사용 >

베이직 언어, 또는 C언어에서 GOTO 명령문과 같습니다. M0가 동작되어 [CJ P10] 을 실행하게 되면 P10레이블 위단은 연산하지 않습니다. 즉 프로그램 코딩에서 실행되지 않기 때문에 M1에 연결되어 있는 Y21코일은 동작하지 않습니다.



< 레이블 넣기 >

레이블을 넣기 위해서는 래더의 왼쪽 바깥에서 마우스로 더블클릭하여 입력할 수 있습니다. 레이블의 이름은 P**로만 가능합니다.

3.4.4 논리연산 명령

일반적으로 스위치의 입력을 확인할 때 비트마스크(Bit Mask)를 사용합니다. 즉 스위치 또는 센서의 입력 유무를 확인할 때 비트어드레스의 접근이 허용되지 않는다면, 비트마스크를 이용하여 그 비트의 값을 확인할 수 있습니다.

스위치 S1, S2, S3, S4가 있고 이 스위치의 값을 비트 단위로 읽지 못한다면 다음과 같은 수식을 사용하여야 합니다.

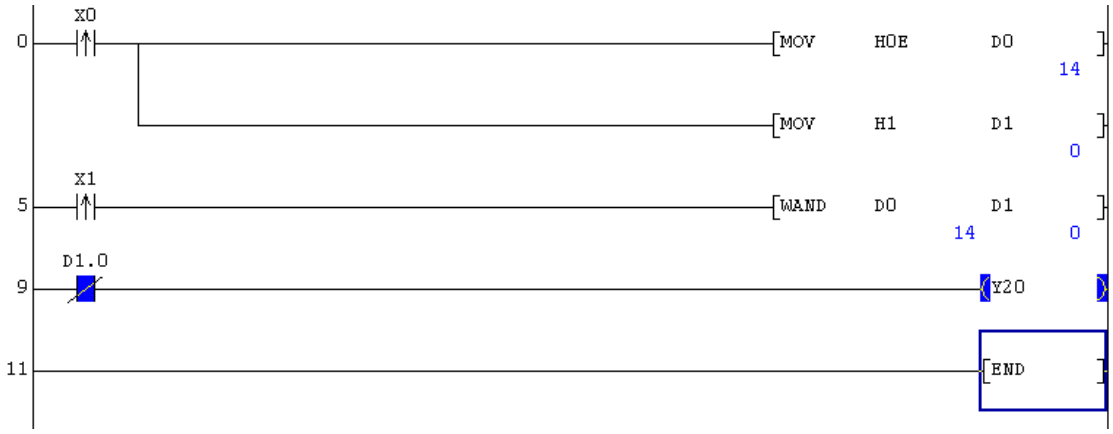
스위치 / 조건	스위치는 모두 'H'이며 눌렸을 때 'L' 임			
	S4	S3	S2	S1
AND	0	0	0	1
S1이 ON일 때	0	0	0	0
S1이 OFF일 때	0	0	0	1

< 'AND' 비트마스크 >

아래 AND처리된 결과 값을 확인하시기 바랍니다.

< X0 만 눌렀을 때 >

X0를 누르면 H0e ('H'는 16진수를 의미함) 와 H1이 각각 D0와 D1레지스터에 저장됩니다.

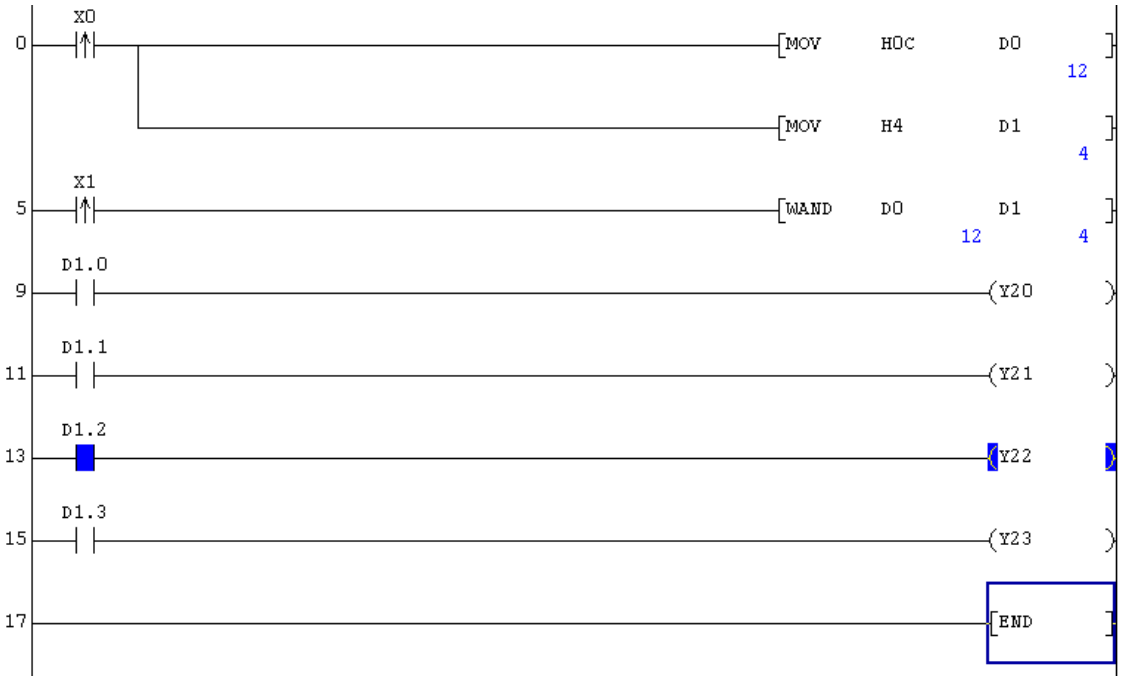


< X1을 눌렀을 때 >

X1을 누르면 D0와 D1이 'AND'연산이 되어 그 결과 값이 D1에 저장됩니다. 즉 D0.0의 값이 '0'임이 확인됩니다.

< H0C와 H04 의 비트 연산1 >

D1.0는 디바이스의 각 비트를 의미합니다. 미쯔비시 PLC에서는 각 디바이스의 비트를 위와 같이 표기합니다.



< H0C와 H04 의 비트 연산2 >

3.4.5 로테이션 명령

< D0에 '1'을 넣고 좌로 로테이션 >

< 1초마다 타이머가 ON/OFF 되고 D0의 값이 쉬프트 >

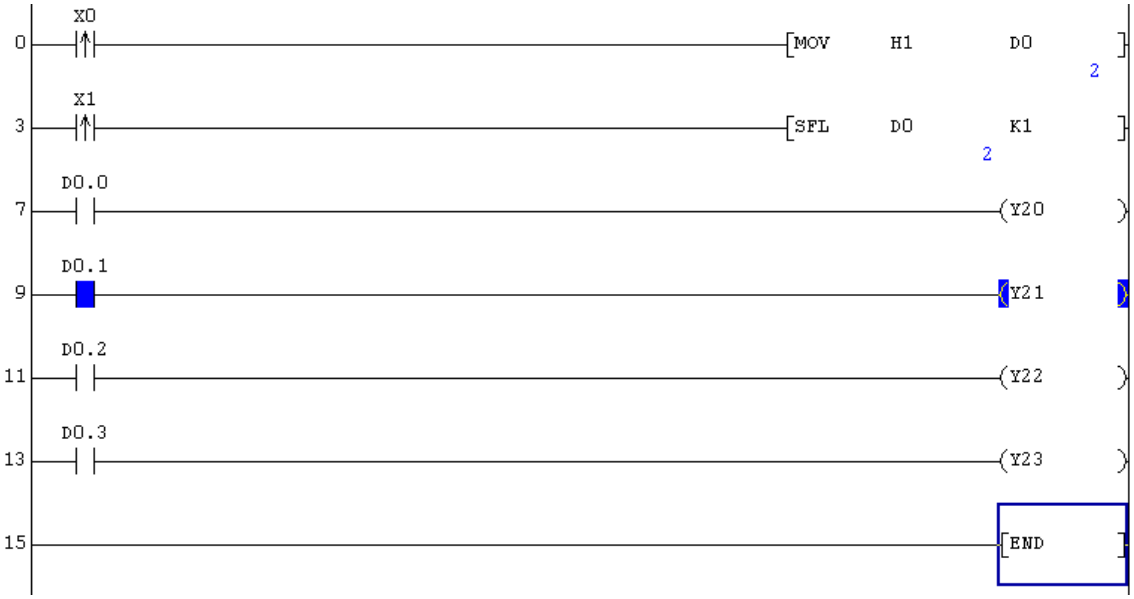
< [ROL D0 K1]에서 K1은 명령 실행시 n비트 이동값 >

분류	명령 기호	심볼	처리 내용	실행 조건	기본스텝수
오른쪽 로테이션	ROR	ROR D n	b15 (D) b0 SM700 오른쪽으로 n비트 로테이션		3
	RORP	RORP D n			
	RCR	RCR D n	b15 (D) b0 SM700 오른쪽으로 n비트 로테이션		3
	RCRP	RCRP D n			
왼쪽 로테이션	ROL	ROL D n	SM700 b15 (D) b0 왼쪽으로 n비트 로테이션		3
	ROLP	ROLP D n			
	RCL	RCL D n	SM700 b15 (D) b0 왼쪽으로 n비트 로테이션		3
	RCLP	RCLP D n			
오른쪽 로테이션	DROR	DROR D n	(D+1) (D) b31 ~ b16 b15 ~ b0 SM700 오른쪽으로 n비트 로테이션		3
	DRORP	DRORP D n			
	DRCR	DRCR D n	(D+1) (D) b31 ~ b16 b15 ~ b0 SM700 오른쪽으로 n비트 로테이션		3
	DRCRP	DRCRP D n			
왼쪽 로테이션	DROL	DROL D n	SM700 (D+1) (D) b31 ~ b16 b15 ~ b0 왼쪽으로 n비트 로테이션		3
	DROLP	DROLP D n			
	DRCL	DRCL D n	SM700 (D+1) (D) b31 ~ b16 b15 ~ b0 왼쪽으로 n비트 로테이션		3
	DRCLP	DRCLP D n			

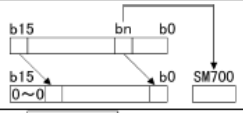
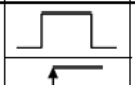
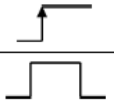
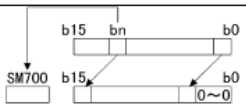
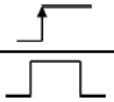
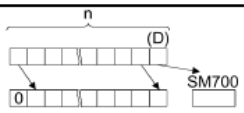
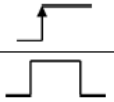
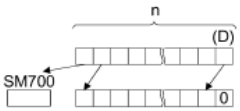
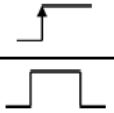
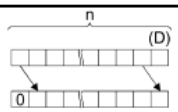
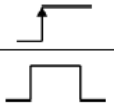
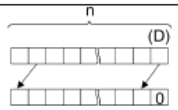
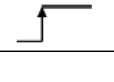
< 그 외 로테이션 명령 >

3.4.6 쉬프트 명령

< D0에 1값을 전송 >

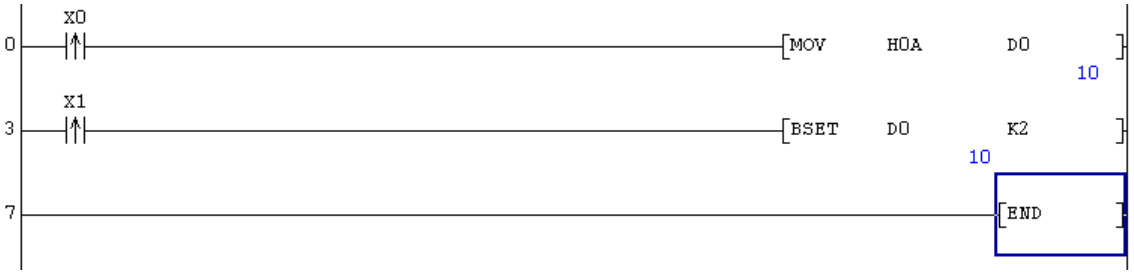


< D0디바이스의 각 비트를 이용하여 출력 >

분류	명령 기호	심볼	처리 내용	실행 조건	기본스텝수
n비트 시프트	SFR	SFR D n			3
	SFRP	SFRP D n			
	SFL	SFL D n			3
	SFLP	SFLP D n			
1비트 시프트	BSFR	BSFR D n			3
	BSFRP	BSFRP D n			
	BSFL	BSFL D n			3
	BSFLP	BSFLP D n			
1워드 시프트	DSFR	DSFR D n			3
	DSFRP	DSFRP D n			
	DSFL	DSFL D n			3
	DSFLP	DSFLP D n			

< 그 외 쉬프트 명령 >

3.4.7 비트처리 명령



< H0A값을 D0에 전송 >

< X1 신호 'ON' 시 1010b → 1110b[2] 로 된 모습>

여기서 K2는 2번째 비트를 의미함

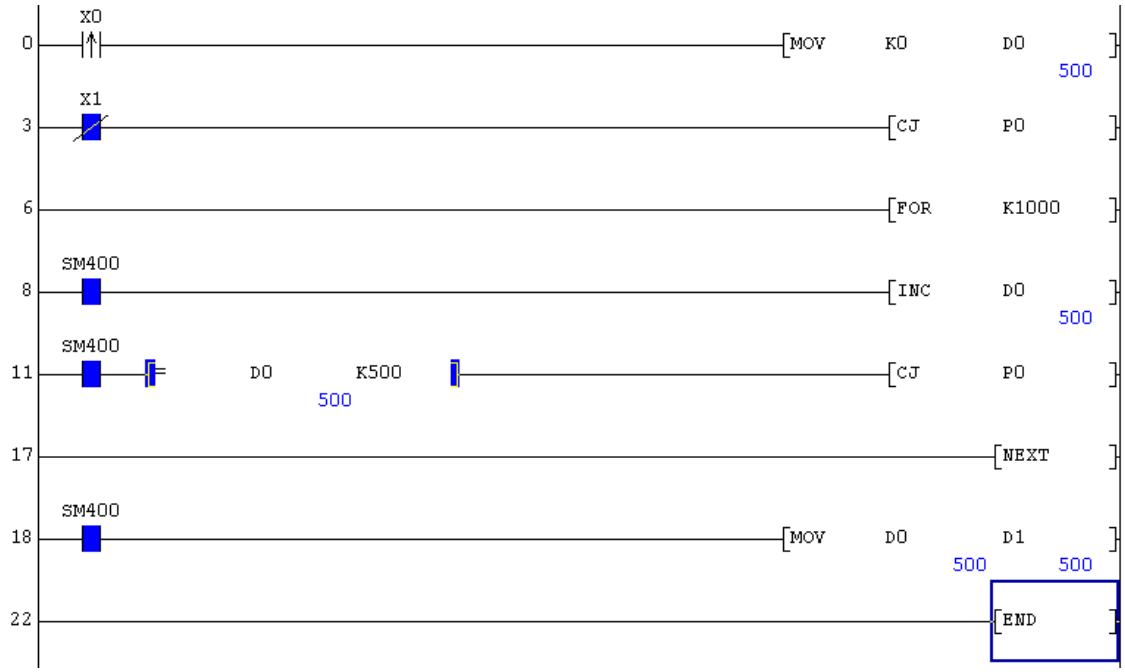
분류	명령 기호	심볼	처리 내용	실행 조건	기본스텝수
비트 세트/ 리세트	BSET	BSET D n	(D) b15 bn b0 1		3
	BSETP	BSETP D n			
	BRST	BRST D n	(D) b15 bn b0 0		3
	BRSTP	BRSTP D n			
비트 테스트	TEST	TEST S1 S2 D	(S1) b15 ~ b0 (D) (S2)로 지정된 비트		4
	TESTP	TESTP S1 S2 D			
	DTEST	DTEST S1 S2 D	(S1) b31 ~ b0 (D) (S2)로 지정된 비트		4
	DTESTP	DTESTP S1 S2 D			
비트 디바이스 일괄 리세트	BKRST	BKRST S n	(S) ON OFF 리세트 (S) OFF OFF n ON ON OFF OFF		3
	BKRSTP	BKRSTP S n			

< 그 외 비트처리 명령 >

2) 1110b에서 b는 2진수를 의미함

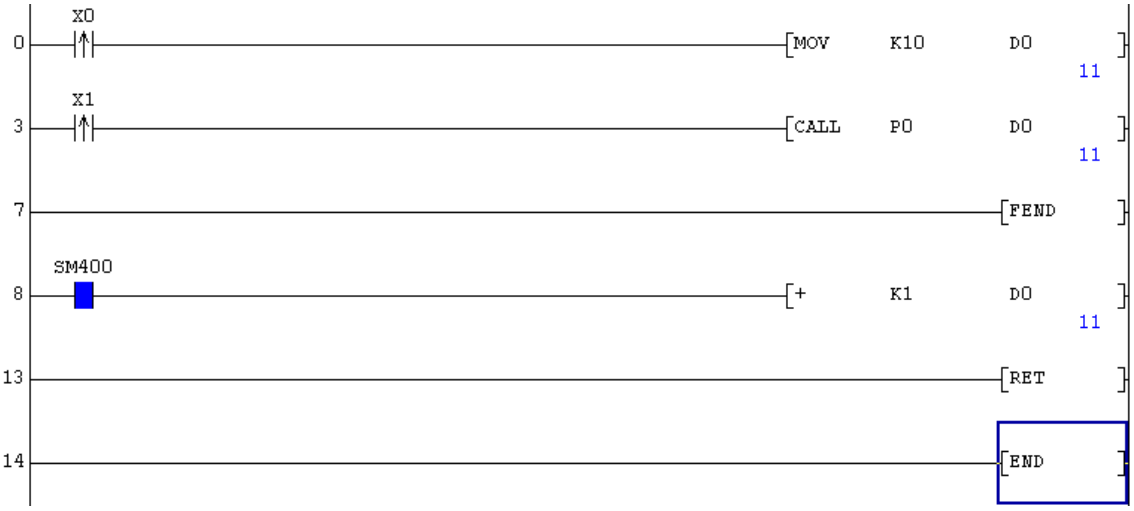
3.4.8 구조화 명령

(1) For Next 명령



< 반복문, 500까지 증가 시킨뒤 점프 >

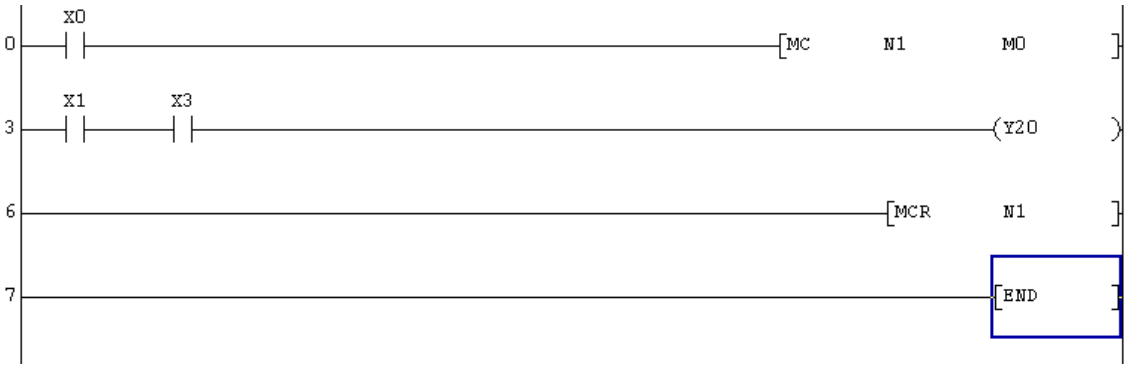
(2) 덧셈 서브 루틴



< 덧셈 서브 루틴 >

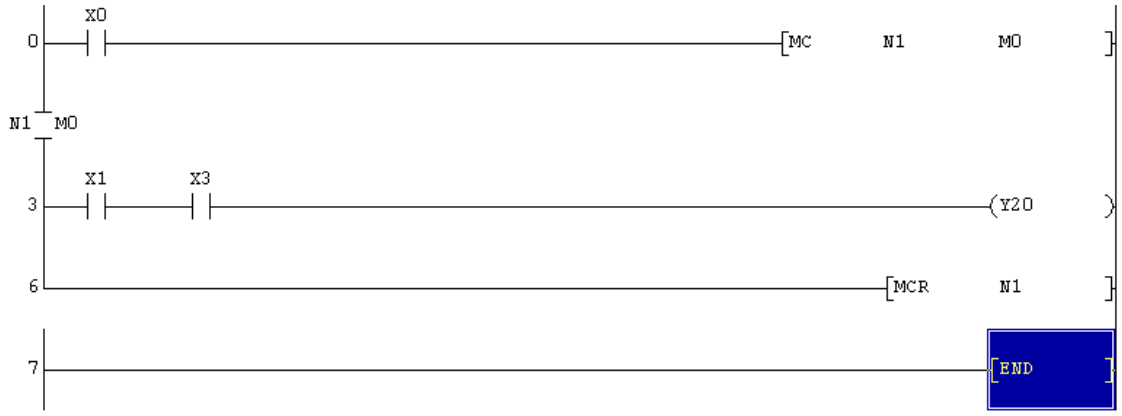
FEND는 메인 프로그램의 종료를 의미함.

(3) 마스터콘트롤



< 마스터 콘트롤 프로그램 >

위와 같이 프로그램을 하고 난뒤 컴파일 하고, 모니터링을 하면 아래와 같이 래더가 변경 됩니다.



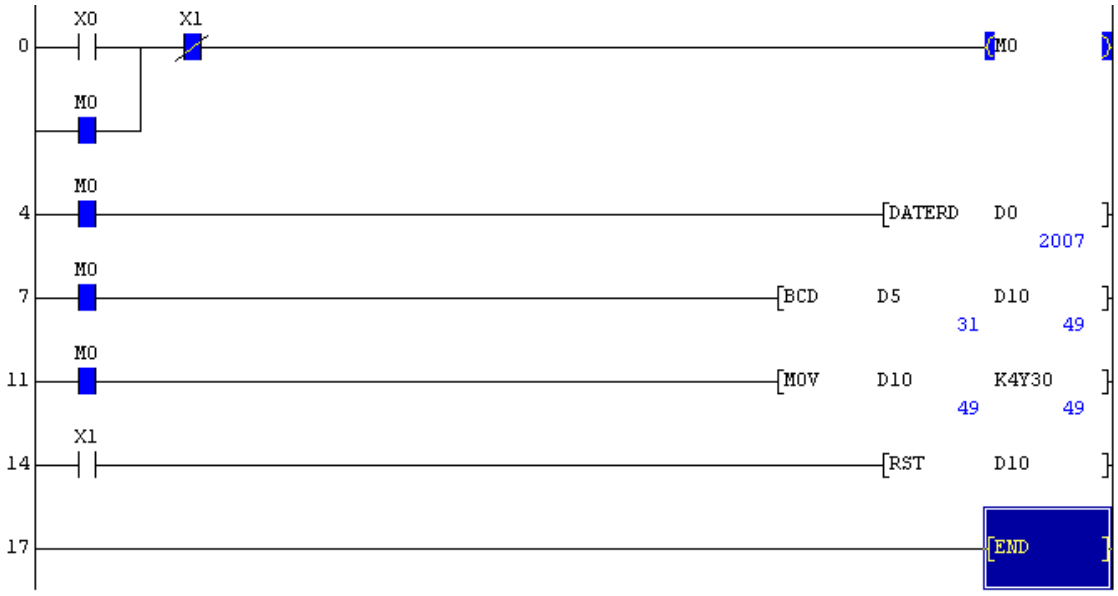
< 마스터 콘트롤 모니터링 >

X0가 ON되어 M0가 ON되면 스텝3라인이 실행 가능합니다. 구조화 프로그램을 작성할 때 유용하게 사용될 수 있습니다.

3.4.9 시계데이터 읽기

분류	명령 기호	심볼	처리 내용	실행 조건	기본스텝수	서브세트	참조 페이지
시계 데이터의 읽기/ 쓰기	DATERD	DATERD D	· (시계 소자) → (D)+0 년, +1 월, +2 일, +3 시, +4 분, +5 초, +6 요일		2		7-271
	DATERDP	DATERDP D					
	DATEWR	DATEWR S	· (D)+0 년, +1 월, +2 일, +3 시, +4 분, +5 초, +6 요일 → (시계 소자)		2		7-275
	DATEWRP	DATEWRP S					
시계 데이터의 가감산	DATE+	DATE+ S1 S2 D	(S1) 시 분 초 + (S2) 시 분 초 → (D) 시 분 초		4		7-279
	DATE+P	DATE+P S1 S2 D					
	DATE-	DATE- S1 S2 D	(S1) 시 분 초 − (S2) 시 분 초 → (D) 시 분 초		4		7-281
	DATE-P	DATE-P S1 S2 D					
시계 데이터의 변환	SECOND	SECOND S D	(S) 시 분 초 → (D) 초 (하위) 초 (상위)		3		7-283
	SECONDP	SECONDP S D					
	HOUR	HOUR S D	(S) 초 (하위) 초 (상위) → (D) 시 분 초				
	HOURP	HOURP S D					

< PLC 내부 RTC 관련 함수 >



< DATERD 및 BCD, MOV 명령 사용>

시스템시간(PLC 내부의 시계, RTC)읽기 명령 DATERD를 읽어 들이면 연, 월,일,시,분,초, 등이 D0데이터부터 입력이 됩니다.

D5데이터가 '초'에 해당되며, D5를 BCD로 변환하여 D10에 저장하고, D10 데이터를 외부 출력 모듈 Y30에 있는 FND(7세그먼트)모듈로 보내 간단한 '초시계'를 만들 수 있습니다.